SRL2004

ICML 2004 Workshop on Statistical Relational Learning and its Connections to Other Fields

Workshop Co-chairs

Tom Dietterich, Oregon State University
Lise Getoor, University of Maryland, College Park
Kevin Murphy, MIT AI Lab

Program Committee

Preface

This workshop is the third in a series of workshops held in conjunction with AAAI and IJCAI. The first workshop was held in July, 2000 at AAAI. Notes from that workshop are available at http://robotics.stanford.edu/srl/. The second workshop was held in July, 2003 at IJCAI. Notes from that workshop are available at http://kdl.cs.umass.edu/srl2003/ There has been a surge of interest in this area. The efforts have been diffused across a wide collection of sub-areas in computer science including machine learning, database management, and theoretical computer science.

The goal of this year's workshop is to reach out to related fields that have not participated in previous workshops. Specifically, we seek to invite researchers in computer vision, spatial statistics, social network analysis, language modeling, and probabilistic inference to attend the workshop and give tutorials on the relational learning problems and techniques developed in their fields. These fields have many years of experience in particular kinds of relational learning, and we hope that bringing these diverse communities together, we can all achieve a better understanding of the range of problems and methods that can be brought to bear on relational learning problems.

We'd like to give a big THANK YOU to the program committee.

Looking forward to a lively and productive workshop in Banff.

Tom Dietterich, Oregon State University

Lise Getoor, University of Maryland, College Park

Kevin Murphy, University of British Columbia

Table of Contents

Invited Speakers

Michael Collins, MIT
Mark Handcock, University of Washington
David Heckerman, Microsoft Research
Daniel Huttenlocher, Cornell University
David Poole, University of British Columbia

Abstracts

Stuctured machine learning problems in natural language processing

Michael Collins, MIT CSAIL/EECS

Many problems in natural language processing involve the mapping from strings to structured objects such as parse trees, underlying state sequences, or segmentations. This leads to an interesting class of learning problems: how to induce classification functions where the output "labels" have meaningful internal structure, and where the number of possible labels may grow exponentially with the size of the input strings. Probabilistic grammars -- for example hidden markov models or probabilistic context-free grammars -- are one common approach to this type of problem. In this talk I will describe recent work on alternatives to HMMs and PCFGs, based on generalizations of binary classification algorithms such as boosting, the perceptron algorithm, or large-margin (SVM) methods.

Statistical Models for Social Networks

Mark Handcock, University of Washington

This talk is an overview of social network analysis from the perspective of a statistician.

The main focus is on the conceptual and methodological contributions of the social network community going back over eighty years. The field is, and has been, broadly multidisciplinary with significant contributions from the social, natural and mathematical sciences. This has lead to a plethora of terminology, and network conceptualizations commensurate with the varied objectives of network analysis. As a primary focus of the social sciences has been the representation of social relations with the objective of understanding social structure, social scientists have been central to this development.

We review statistical exponential family models that recognize the complex dependencies within relational data structures. We consider three issues: the specification of realistic models, the algorithmic difficulties of the inferential methods, and the assessment of the degree to which the graph structure produced by the models matches that of the data. Insight can be gained by considering model degeneracy and inferential degeneracy for commonly used estimators.

Probabilistic Entity-Relationship Models, PRMs, and Plate Models

David Heckerman, Microsoft Research

We introduce a graphical language for relational data called the probabilistic entity-relationship (PER) model. The model is an extension of the entity-relationship model, a common model for the abstract representation of database structure. We concentrate on the directed version of this model---the directed acyclic probabilistic entity-relationship (DAPER) model. The DAPER model is closely related to the plate model and the probabilistic relational model (PRM), existing models for relational data. The DAPER model is more expressive than either existing model, and also helps to demonstrate their similarity. In addition to describing the new language, we discuss important facets of modeling relational data, including the use of restricted relationships, self relationships, and probabilistic relationships.

This is joint work with Christopher Meek and Daphne Koller.

Pictorial Structure Models for Visual Recognition

Dan Huttenlocher, Cornell University

There has been considerable recent work in object recognition on representations that combine both local visual appearance and global spatial constraints. Several such approaches are based on statistical characterizations of the spatial relations between local image patches. In this talk I will give an overview of one such approach, called pictorial structures, which uses spatial relations between pairs of parts. I will focus on the recent development of highly efficient techniques both for learning certain forms of pictorial structure models from examples and for detecting objects using these models.

Relations, generalizations and the reference-class problem: A logic programming / Bayesian perspective

David Poole, Dept of Computer Science, University of British Columbia

Logic programs provide a rich language to specify the interdependence between relations. There has been much success with inductive logic programming finding relationships from data. There has also been considerable success with Bayesian learning. However there is a large conceptual gap in that inductive logic programming does not have any statistics. This talk will explore how to get statistics from data. This problem is known as the reference-class problem. This talk will explore the combination of logic programming and hierarchical Bayesian models as a solution to the reference class problem.

This is joint work with Michael Chiang.

Feature Definition and Discovery in Probabilistic Relational Models

Eric Altendorf ERIC@CLEVERSET.COM
Bruce D'Ambrosio DAMBROSI@CLEVERSET.COM
CleverSet, Inc., 673 Jackson Avenue, Corvallis OR, 97330

Abstract

Feature expression in relational models can be viewed as the construction, for a given relational table, of valid path expressions and transformational operators which navigate the relational structure of the schema, propositionalizing native attributes from related tables via selection and aggregation. We present a language for expressing such features ("synthetic variables") and a method for efficiently searching over this language for definitions of relevant and interesting features for a probabilistic relational model.

1. Introduction

Relational and object models frequently make use of the notion of a *path expression.* A basic path expression, such as Product.maker.name , selects data from a table related to the root table of the original query. The use of path expressions in the context of probabilistic relational models (PRMs) has been discussed by Friedman et al. (1999) and Getoor et al. (2001).

Our first contribution has been the development of a path language with greater expressive power than those currently proposed. Some syntax and constructs were inspired by the PathLog language for OODBs developed by Frohn et al. (1994). However, the application to PRMs differs significantly and required both new features (such as arbitrary function chaining and data filtering), as well as a library of useful aggregators and operators. Our second contribution has been the development of automated tools to allow the modeler to make effective use of the richness of the language.

To motivate our discussion, consider the following examples of features we might wish to define (these examples are taken from a website user behavior model). Note that the first two are expressible in the language used in (Friedman et al., 1999; Getoor et al., 2001), while the rest are not.

- Session.clicks.Count() : For a given session, return the number of clicks that were in that session.
- Session.clicks.url.Mode() : For a session, return the most frequently requested URL.
- Click.session.clicks.pagetype.Uniquify().Count() : For a click, find the number of distinct page types visited in that session.
- Click.session.clicks.Diff(.time,.prev.time).Mean() : For a click, find its session and calculate the average time between clicks in that session.
- Click.Diff(.session.clicks[.GT(.time, $src.time)]
  [.Equals(.pagetype,"Checkout")].sequence_no,
  $src.sequence_no).Min() : For a click, return the number of subsequent clicks in the session before the first request for a Checkout page—that is, the number of clicks until a purchase event.

2. Expression language

2.1. Grammar

Table 1. Synthetic variable abstract grammar

expr	:	rootelem{elem}+
elem	:	.field | .function | .this | variable | selector | constant | universal
rootelem	:	classname | variable | constant
selector	:	[expr]
function	:	funcname({expr}*)
constant	:	`'[0-9]+{.[0-9]+}?'` | `"[^"]+"`
variable	:	$ varname
universal	:	* classname

The grammar is defined in table 1. The basic construct in the language is an *expression*, which is a chain of elements. The first element is a *root element* which types the source datum, and subsequent elements define data mappings. Some elements contain and make use of subexpressions. Generally speaking, expressions may make sense only in specific contexts (for example, as subexpressions). A *synthetic variable* is an expression which is well-defined given no special context other than the source datum on which it is to be evaluated.

A few minor points bear special mention. In the context of a subexpression, as a syntactic sugar we omit the classname element since it is implied by its context. Also, we name two special types of functions: *operators*, such as Equals() , Diff() , or GT() (greater than), which take two arguments, and *aggregators*, such as Mean() or Count() , which take no arguments and perform a many-to-one cardinality mapping (see 2.3). Finally, the .this element is a special no-op construct necessary in certain subexpression constructions.

2.2. Semantics

Expressions may be evaluated on an instance of the class associated with the root element. For example, Session.clicks.Count() may be evaluated on instances from the Session table, and Session.clicks[.Equals(.pagetype,.prev.pagetype)].Count() contains three subexpressions: (i) .Equals(.pagetype,.prev.pagetype) , (ii) .pagetype , and (iii) .prev.pagetype , each of which may be evaluated on an instance from the Click table.

Evaluation of an expression proceeds left to right, each element accepting data from the previous element and producing data for the subsequent element. The value of the expression is the output of the last element in the chain.

2.2.1. Fields

The evaluation of a field element outputs the reference or primitive value(s) contained in the appropriate field on the incoming data object. If the incoming data is a singleton and the field is single-valued, the output is a singleton. Otherwise, the output is multivalued. If the incoming data is multivalued and the field is multivalued, the result will be a flattened set (the bag union of all field values on all instances from the incoming data); that is, we do not support nested collections.

2.2.2. Selectors and functions

The evaluation of a selector returns a subset of the incoming data—specifically, the collection containing each datum in the incoming data on which the provided subexpression evaluates to true.

The evaluation of a function returns a value based on the incoming data (and, if applicable, the values produced by evaluating the subexpressions on the incoming datum). For example, two-argument boolean operators such as Equals() or GT() take the incoming datum, evaluate each subexpression on that datum, and return a boolean value based on a comparison of those results.

Operators provide an implicit "map" behavior so that when applied to multivalued incoming data, they apply themselves to each datum in turn. Thus, one can construct variables such as Click.session.clicks .Diff(.timestamp,.prev.timestamp).Mean() .

2.2.3. Intra-expression variables

One seemingly expressive language construct which has turned out to be less useful than we predicted is arbitrary variable binding. The only important application we have found occurs when subexpressions refer to the original source datum on which the synthetic variable is being evaluated. We support this by implicitly binding the source datum to the $src variable, but (currently) do not offer a mechanism for arbitrary binding of variables (see 5.3).

2.2.4. Universal selection

In some datasets, we do not have explicit relations. Consider a spatiotemporal dataset, in which we would like to aggregate over spatially or temporally proximal events, but in which we lack fields (such as adjacent) to navigate the data. In this case, we must select *all* instances of a class, and then filter by our temporal or spatial conditions. We use the *universal selection* element, denoted by *classname , for this. For example, in the West Nile Virus domain (see 3.3.1), we have events indicating occurrences of the disease in humans and birds. To count the number of prior bird cases for a given human case, we can write: HumanCase*BirdCase[.GT($src.date,.date)].Count() This functionality gives us the power to perform two-table joins on arbitrary conditions.

2.3. Typing

To facilitate type-checking, each element defines a required input and guaranteed output type. These types will depend on the relational schema, but not the data, meaning that expressions can be statically typechecked given the schema.

Datatypes are either reference (one type per table in the schema, with polymorphic subtyping allowed) or primitive[1]. Types also specify the cardinality of the data: singleton or multivalued (0 to n).

For an expression to be correctly typed, each element

[1]Currently string, numeric, or boolean, though we believe a type system based on measurement types such as nominal, ordinal, and ratio, might be more useful.

must accept the type of data produced by the preceding element. For example, field elements require input of the reference type (or subtype) on which that field is defined, aggregators require (possibly) multivalued inputs, single-input functions require singleton incoming data, and numeric aggregators like `Mean()` or `Sum()` require numeric input.

We require expressions to produce singleton data. This guarantees that synthetic variables can be evaluated to a single value, and that selectors and operators may evaluate their subexpressions to a single value.

2.4. Domain specific extensions

We have also built special purpose extensions within the grammar for particular problem domains. This capability is essential for doing useful applied modeling. Space constraints prohibit discussion, but some examples from spatiotemporal domains include: (i) functions for reifying and operating on temporal data, (ii) specialized selectors which allow rapid parameter adjustments for exploring scale effects, and (iii) density aggregators which normalize spatial data counts by the area of the sampled region.

3. Search

With a definition of the syntax and semantics of the language we can automatically enumerate synthetic variables. We view this problem as a search (for useful variables) in a very large search space (the space of all grammatically correct variables). An equivalent view is as a search over the space of possible database queries (Popescul & Ungar, 2003).

Our basic search is breadth-first, using a variant of the traditional cost ranking of path expressions by their *path length* (the number of field traversals they define).[2] Specifically, classname elements, which are in some sense artifacts of our grammar, cost nothing, and selectors and functions cost only what their their subexpressions cost.

3.1. Problem description

The space of possible synthetic variables without subexpressions is exponential in its length. (The base of the exponent is of course dependent on the relational schema and the aggregators enabled in the search.) Subexpressions introduce exponential branching—for instance, the number of selectors that may be applied at a given point will be exponential in the allowable size of subexpressions—and we therefore generally have superexponential overall growth.

As a very rough guide, in a simple schema, using no search optimizations or heuristics, search up to complexity (depth) 4 or 5 is generally reasonably tractable, but many interesting variables occur at complexities ranging from 8 to 12. In recent work with a major e-retailer, we have found valuable variables at complexities of 20 to 22. It is clear that such variables cannot be found by brute breadth-first enumeration, and so we introduce heuristics, synthetic variable filtering rules, and an interactive search process.

3.2. Synthetic variable filters

Besides restricting synthetic variables for grammatical and type correctness, we implement a number of filtering rules to prune the search space.

Field loop suppression: Traversals of one-to-many fields followed by their many-to-one inverse are generally useless, and so we suppress their generation. For example, while `Product.maker.products` makes sense (return all products produced by the same maker), `Product.maker.products.maker` is identical (up to repetition of data) as `Product .maker`. Loops of this nature may also span subexpression boundaries.

Repeated fields: The modeler may wish to limit chains of repeats of a transitive field, such as `Click .prev`, which could be arbitrarily extended to `Click .prev.prev.prev...`. The modeler can limit such repetitions on a per-field basis.

Field costs: The modeler may also wish to include certain fields more often or less often than others (due to an interest in a certain relationship, or to artifacts of the data's relational encoding). We allow the modeler to override the default cost of 1 on a per-field basis, to any positive value. Values greater than 1.0 shrink the search space, while values less than 1.0 increase it (but simultaneously cause variables including that field to appear at lesser search depths).

Field filtering: In some of our models data are related both spatially and temporally. However, our grammar, which is primarily navigational, does not provide a mechanism for multi-column joins. Therefore, we use a field traversal in combination with a filter, e.g.: `Posbirds.geocell.posMosqPool [.Equals(.month,$src.month)]`. By requiring that, say, temporal fields like `.month` are *only* used within selectors, we can generate expressions like the one above yet rule out ones like `Posbirds.geocell`

[2] We do offer other queue prioritizations—see 5.1.

.posMosqPool.month.posBirds . Although this may be a suboptimal solution, it has worked quite well in practice thus far.

Operator arguments: Arguments to a binary operator such as Equals() or GT() must be of the same type, but this does not prevent semantically meaningless combinations—product weight and maker's annual sales may both be numeric, but we shouldn't compare the two. We adopt the conservative rule that, for primitive types, comparisons are formed only between values from the same field on the same table.

Aggregator selection: It is very easy for the search routine to create vast numbers of variables merely by appending aggregators on the end of expressions, most of which are uninteresting. Therefore, we require that the modeler specifically enable aggregators in the search on a per-field basis.

Trivial domains: Although we focused on non-data-driven feature selection so we could operate in data-limited situations, we do implement a basic filtering rule for variables with trivial domains (i.e., constant). Variables may be constant because they aggregate large amounts of missing or constant data, or they may be provably axiomatic. While such variables may (occasionally) reveal something interesting about the domain or the data, they are not useful as random variables in a probabilistic model.

3.3. Empirical evaluation

The general quality of our results is encouraging. Recall (the proportion of desired variables which were found) appears to be quite good, and precision (proportion of variables returned which are interesting) is acceptable. For instance, at depth 9 in our West Nile Virus schema (see 3.3.1), we generate approximately 100 synthetic variables for the PosBirds table, compared to roughly 100,000 possible grammatically correct variables. The generated set includes all variables we had previously determined as essential to the model, and roughly 25% of the variables generated appear relevant to the model being constructed. Finally, the search is tractable; it takes between 30 seconds and several minutes on a desktop workstation.

We here present more detailed results of experiments we conducted to evaluate our pruning heuristics. Note that designing such experiments is difficult, because performance numbers and results vary highly. A slight change in the branching factor of the schema or the set of enabled aggregators could easily change the results for a given search by an order of magnitude. Additionally, there are a large number of free parameters, including (i) schema and set of enabled aggregators, (ii) amount of data populating the schema, (iii) depth of search, (iv) depth of subexpression search, and (v) filters applied.

3.3.1. Schemas used

We use two schemas, a weblog click-stream schema from an online retailer, and a geospatial epidemiological schema with data on the West Nile Virus in Maryland. In tables 2 and 3 we summarize the database tables, the number of instances (rows) in each, the numbers of each type of column, and the total number of aggregators enabled on various columns therein. Multivalued ("$*$-val") reference columns are constructed as implicit inverses.

Table 2. Clickstream schema

Table	# of rows	Prim. cols	Ref. cols 1-val	Ref. cols $*$-val	# of agg.
Click	3789	7	4	0	0
Page	4823	8	0	0	0
Session	802	4	1	1	1
Visitor	563	8	0	0	0

Table 3. West Nile Virus schema

Table	# of rows	Prim. cols	Ref. cols 1-val	Ref. cols $*$-val	# of agg.
PosBirds	77	1	2	0	0
GeoCell	1248	0	0	7	1
Adjacent	9388	0	2	0	0
Month	13	1	1	4	0
PosMosqPl	10	1	2	0	1
NegMosqTr	73	1	2	0	1
Horse	5	1	2	0	1
Human	3	1	2	0	1
License	2116	1	1	0	0

3.3.2. Effectiveness of heuristics

Given a fixed schema, dataset, aggregator selection, and maximum subexpression complexity, we begin by searching with no heuristics, then enable them, one by one, and measure for each search depth: (i) the elapsed time, and (ii) the number of variables generated. The results are shown in figures 1, 2, 3, and 4. Missing data points indicate that that particular search either ran out of memory or time.[3] Almost all searches ran out of time or memory at depth 10. Space was generally a problem when operating with fewer heuristics, as the

[3]The memory limit was a 512MB Java heap size limit, and the time limit was 30 minutes. This of course is a research prototype, and much more efficient implementations are possible.

search created too many variables, while time was a problem when operating with more heuristics, as the search spent too long filtering (particularly in evaluating and removing variables with trivial domains).

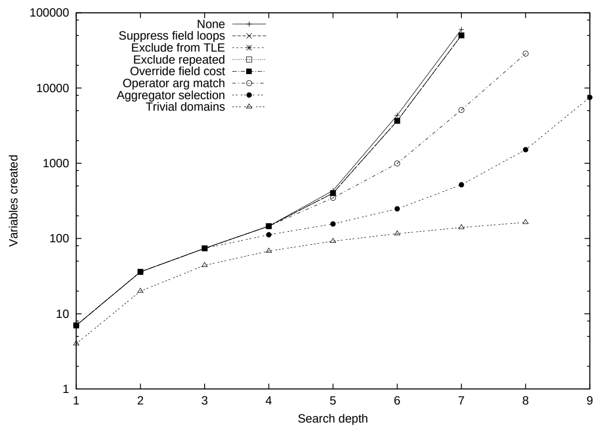


Figure 1. Number of variables created, clickstream schema

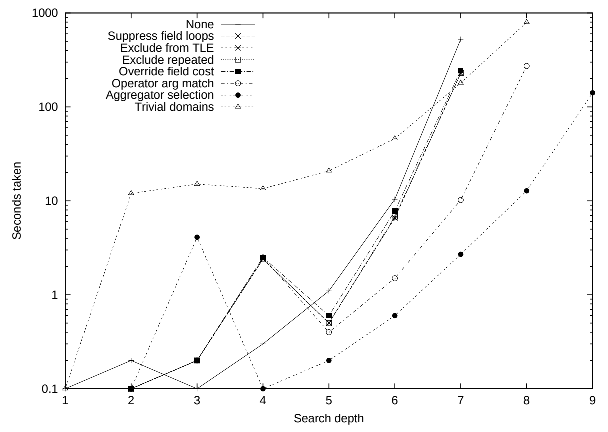


Figure 2. Time taken, clickstream schema

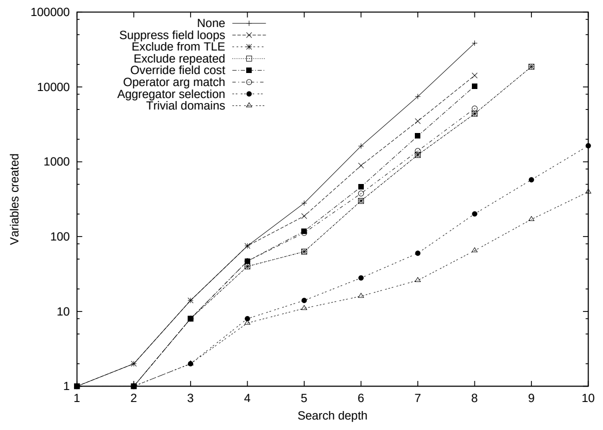


Figure 3. Number of variables created, WNV schema

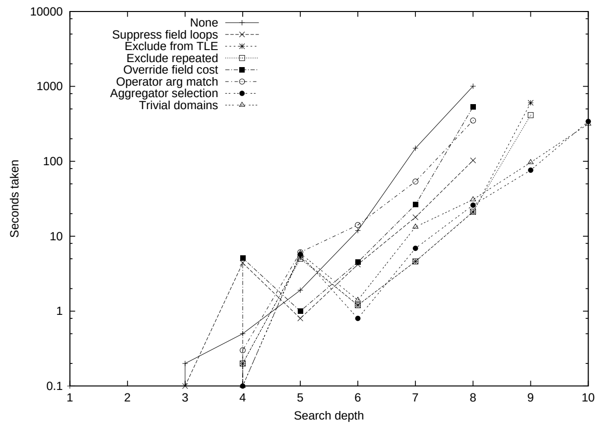


Figure 4. Time taken, WNV schema

4. Interactive search process

Even by eliminating large regions of the search space the number of variables generated through the fully automatic search is generally too large for a PRM. We therefore designed the system with various features to make the feature selection process interactive.

4.1. Search results review

First, we separated feature search from model building. The modeler may perform an explicit synthetic variable search, and then review the results. Each synthetic variable in the results list may be evaluated on the loaded instance data, to produce either raw data or histograms, and can be deleted from the search results list or added to the PRM (for inclusion in later structure discovery).

4.2. Partial expressions

Second, we allow the creation of *partial expressions.* Partial expressions can be extended in a manual depth-first search, where possible next elements are automatically found and presented to the modeler for selection, or provided as a "seed" to the search algorithm, to automatically search for possible completions. In this way, with assistance from the modeler, the search algorithm can be used at arbitrary depths.

4.3. Complete manual specification

We also provide a parser which allows the modeler to input free-form synthetic variable definitions. We have found a number of interesting variables in a few domains which can only be created with the parser.

5. Future work

5.1. Search queue prioritizations

We search by producing continuations of partial expressions in a priority queue, normally prioritized by lowest expression complexity (as discussed). We also have two other scoring options, based on metrics calculated from complete expressions:[4]

Expression entropy: This lets us prioritize high-entropy variables and deprioritize nearly-constant variables. It is a generalization of the trivial domain filtering rule, which immediately removes from the search queue variables with zero entropy.

Log-likelihood: We prioritize by predictive power of the variable, measured by the log-likelihood of the data for modeler-selected "target" variables, given a model including the variable in question.

These techniques have not been empirically tested for performance, however, and deserve further study.

5.2. Extending the search space

There are many potentially interesting spaces over which we cannot automatically search, such as lengthy subexpressions, and values of constants passed as arguments to operators.

One approach is to develop new heuristics and search strategies. Another is to improve expression evaluation efficiency, since trivial domain testing (which requires evaluation of candidate synthetic variables) consumes the majority of time during search when the underlying dataset is large. Future research should address scalability issues, streamline evaluation, and/or develop statistically sound sampling techniques.

5.3. Intra-expression variables

Our original design called for general intra-expression variables which could be bound within an expression and used in other parts of the expression. We also considered allowing a variable bound multiple times to define an implicit join condition on the data, as it does in PathLog. Further research is needed to determine what, if any, expressive power such variables would add to the language (as we have not yet found a need for them).

[4]Partial expressions cannot be evaluated or scored on their own, so we prioritize them by an average of the scores of known complete expressions, weighted by a syntactic similarity metric.

5.4. Latent synthetic variables

One simplifying assumption we make in much of our work is that we do not need to do inference over data missing from a path expression. That is, we assume that the evaluation of a path expression is deterministic, or that path expressions are never latent variables of the model. Relaxing this assumption of course requires more advanced unrolling techniques to implement language elements as Bayes net fragments.

6. Conclusion

We have presented an expressive and efficiently evaluable language for defining derived attributes in relational models. Our experience has shown this language sufficiently expressive for a very wide range of real-world models. We have presented a set of heuristics for constraining searches over the space of possible expressions, and validated these heuristics empirically with performance tests, showing that they make the search space significantly more tractable. We have also discussed an interactive search process for going beyond the limits of fully automatic search. Finally, we discussed several important directions for valuable future research.

Note: parts of the described language and search capabilities are patent-pending.

References

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99-Vol2)* (pp. 1300–1309). S.F.: Morgan Kaufmann Publishers.

Frohn, J., Lausen, G., & Uphoff, H. (1994). Access to objects by path expressions and rules. *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, Sep. 12-15, 1994, Santiago de Chile, Chile* (pp. 273–284). Morgan Kaufmann.

Getoor, L., Friedman, N., Koller, D., & Taskar, B. (2001). Learning probabilistic models of relational structure. *Proceedings of the 18th International Conference on Machine Learning* (pp. 170–177). Morgan Kaufmann, San Francisco, CA.

Popescul, A., & Ungar, L. H. (2003). Statistical relational learning for link prediction. *Proceedings of the IJCAI 2003 Workshop on Learning Statistical Models from Relational Data.*

A Comparison of Inference Techniques for Semi-supervised Clustering with Hidden Markov Random Fields

Mikhail Bilenko MBILENKO@CS.UTEXAS.EDU
Sugato Basu SUGATO@CS.UTEXAS.EDU
Department of Computer Sciences, University of Texas at Austin, 1 University Station C0500, Austin, TX 78712 USA

Abstract

Recently, a number of methods have been proposed for semi-supervised clustering that employ supervision in the form of pairwise constraints. We describe a probabilistic model for semi-supervised clustering based on Hidden Markov Random Fields (HMRFs) that incorporates relational supervision. The model leads to an EM-style clustering algorithm, the E-step of which requires collective assignment of instances to cluster centroids under the constraints. We evaluate three known techniques for such collective assignment: belief propagation, linear programming relaxation, and iterated conditional modes (ICM). The first two methods attempt to globally approximate the optimal assignment, while ICM is a greedy method. Experimental results indicate that global methods outperform the greedy approach when relational supervision is limited, while their benefits diminish as more pairwise constraints are provided.

1. Introduction

There has been significant recent interest in *semi-supervised clustering*, where the goal is to improve the performance of unsupervised clustering algorithms with limited amounts of supervision in the form of labels or pairwise constraints on the data points (Wagstaff et al., 2001; Klein et al., 2002; Xing et al., 2003; Bilenko et al., 2004). In this paper, we present a probabilistic model for semi-supervised clustering with pairwise relations and compare the performance of several inference methods for cluster assignment in the context of an EM-based algorithm.

Existing methods for semi-supervised clustering fall into two general categories that we call *constraint-based* and *distance-based*. Constraint-based methods rely on user-provided labels or relational constraints to guide the algorithm towards a more appropriate data partitioning (Wagstaff et al., 2001). In distance-based approaches, an existing clustering algorithm that uses a particular clustering distortion measure is employed, but the measure is trained to satisfy the labels or constraints in the given supervised data (Klein et al., 2002; Xing et al., 2003; Cohn et al., 2003; Bar-Hillel et al., 2003). In Bilenko et al. (2004), we have proposed an integrated framework for semi-supervised clustering that combines the constraint-based and distance-based approaches in a unified probabilistic model.

We have recently shown that this semi-supervised clustering framework has an underlying probabilistic model – a Hidden Markov Random Field (HMRF) (Basu et al., 2004). Then, minimizing the integrated clustering objective function becomes equivalent to finding the maximum *a posteriori* probability (MAP) configuration of the HMRF (Zhang et al., 2001). It can be shown that the HMRF clustering model is able to incorporate any Bregman divergence (Banerjee et al., 2004) as the clustering distortion measure, which allows using the framework with such common distortion measures as KL-divergence, I-divergence, and parameterized squared Mahalanobis distance. Additionally, cosine similarity can also be used as the clustering distortion measure in the framework, which makes it useful for directional datasets.

The HMRF semi-supervised clustering model suggests an EM-based algorithm that minimizes the integrated clustering objective function to obtain a partitioning of the dataset. The E-step of the algorithm can be mapped to an inference step of a probabilistic relational model. In prior work, we used a fast greedy iterated conditional modes (ICM) inference technique in the E-step (Bilenko et al., 2004; Basu et al., 2004). Here, we compare ICM to two global approximate inference techniques for relational models: belief propagation and linear programming (LP) relaxation. Our experiments reveal that, when provided with sufficient relational supervision, ICM produces results comparable to belief propagation and LP relaxation on the constraint-based semi-supervised clustering task at a fraction of computational cost.

2. Background

2.1. The HMRF Clustering Framework

Given a set of data points $\mathcal{X}$, sets of pairwise must-link constraints $\mathcal{M}$ and cannot-link constraints $\mathcal{C}$ with sets of asso-

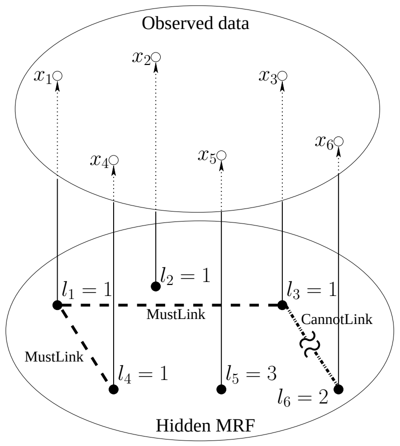


Figure 1. A Hidden Markov Random Field

ciated violation costs $\mathcal{W}$ and $\overline{\mathcal{W}}$, and a distortion measure D between the points, the semi-supervised clustering task is to optimally partition $\mathcal{X}$ into K clusters.[1] An optimal partitioning is that which minimizes the total distortion between the points and their cluster representatives according to D, while keeping constraint violations to a minimum.

This problem can be formalized as the task of label assignment in a Hidden Markov Random Field (HMRF) (Basu et al., 2004). The HMRF model consists of (1) a *hidden* field $\mathcal{L} = \{l_i\}_{i=1}^N$ of random variables that encode cluster assignments of data points; and (2) an *observable* set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ of random variables that correspond to observed data points. Every data point $\mathbf{x}_i$ is assumed to be generated from a conditional probability distribution $\mathbf{Pr}(\mathbf{x}_i|l_i)$ determined by the value of the corresponding hidden variable l_i. Random variables $\mathcal{X}$ are conditionally independent given the hidden variables $\mathcal{L}$, i.e., $\mathbf{Pr}(\mathcal{X}|\mathcal{L}) = \prod_{\mathbf{x}_i \in \mathcal{X}} \mathbf{Pr}(\mathbf{x}_i|l_i)$.

Note that a relational model similar to HMRFs has been proposed by Segal et al. (2003) for semi-supervised clustering with constraints, except that it only utilized must-link constraints and did not incorporate learnable distortion measures.

Fig. 1 shows a simple example of an HMRF. The observed dataset $\mathcal{X}$ consists of six points $\{\mathbf{x}_1 \ldots \mathbf{x}_6\}$ with corresponding cluster label variables $\{l_1 \ldots l_6\}$. Two must-link constraints are provided between (l_1,l_3) and (l_1,l_4), and one cannot-link constraint is provided between (l_3,l_6). The task is to partition the six points into three clusters. One clustering configuration is shown in Fig. 1, where the must-linked points $\mathbf{x}_1, \mathbf{x}_3$ and $\mathbf{x}_4$ are put in cluster 1; the point $\mathbf{x}_6$, which is cannot-linked to $\mathbf{x}_3$, is assigned to cluster 2; $\mathbf{x}_2$ and $\mathbf{x}_5$, which are not involved in any constraints, are put in clusters 1 and 3 respectively.

[1]Must-link and cannot-link constraints specify pairs of points that should be in same and different clusters respectively.

Every hidden random variable l_i has an associated set of neighbors $\mathcal{N}_i$. The must-link constraints $\mathcal{M}$ and cannot-link constraints $\mathcal{C}$ define the neighborhood over each hidden variable to be all the points that are must-linked or cannot-linked to the corresponding data point. A Markov Random Field is then defined over the hidden variables.

Let us consider a particular cluster label configuration $\mathcal{L}$ to be the joint event $\mathcal{L} = \{l_i\}_{i=1}^N$, which corresponds to a specific assignment of data points to clusters. By the Hammersley-Clifford theorem, the probability of a label configuration in the Markov Random Field can be expressed as a Gibbs distribution (Geman & Geman, 1984):

$$\mathbf{Pr}(\mathcal{L}) = \frac{1}{Z} e^{-V(\mathcal{L})} = \frac{1}{Z} e^{-\sum_{\mathcal{N}_i \in \mathcal{N}} V_{\mathcal{N}_i}(\mathcal{L})} \qquad (1)$$

where $\mathcal{N}$ is the set of all neighborhoods, Z is a normalizing constant, and $V(\mathcal{L})$ is the overall label configuration potential function, which can be decomposed into the functions $V_{\mathcal{N}_i}(\mathcal{L})$ denoting the potential for every neighborhood $\mathcal{N}_i$.

Since we are provided with pairwise constraints over the class labels, we restrict the MRFs over the hidden variables to have pairwise potentials. The prior probability of a configuration of cluster labels $\mathcal{L}$ then becomes $\mathbf{Pr}(\mathcal{L}) = \frac{1}{Z}\exp(-\sum_i \sum_j V(i,j))$, where

$$V(i,j) = \begin{cases} f_M(\mathbf{x}_i, \mathbf{x}_j) & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \\ f_C(\mathbf{x}_i, \mathbf{x}_j) & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

Here, $f_M(\mathbf{x}_i, \mathbf{x}_j)$ is a non-negative function that penalizes the violation of a must-link constraint, and $f_C(\mathbf{x}_i, \mathbf{x}_j)$ is the corresponding penalty function for cannot-links. Intuitively, this form of $\mathbf{Pr}(\mathcal{L})$ gives higher probabilities to label configurations that attempt to satisfy the must-link constraints $\mathcal{M}$ and cannot-link constraints $\mathcal{C}$.

It is possible to use a learnable distortion measure D that adapts distance computations to respect the user-provided constraints. To facilitate learning of distortion measure parameters, the penalty for violating a must-link constraint between *distant* points should be higher than that between *nearby* points. This reflects the fact that if two must-linked points are far apart according to the current distortion measure, the distance measure parameters need to be modified to bring those points closer together. Inversely, the penalty for violating a cannot-link constraint between two points that are *nearby* according to the current distance measure should be higher than for two *distant* points. To reflect this reasoning, the penalty functions are chosen as follows:

$$f_M(\mathbf{x}_i, \mathbf{x}_j) = w_{ij}\varphi_D(\mathbf{x}_i, \mathbf{x}_j)\mathbb{1}[l_i \neq l_j] \qquad (3)$$

$$f_C(\mathbf{x}_i, \mathbf{x}_j) = \overline{w}_{ij}\big(\varphi_{D\max} - \varphi_D(\mathbf{x}_i, \mathbf{x}_j)\big)\mathbb{1}[l_i = l_j] \qquad (4)$$

where φ_D is a monotonically increasing *penalty scaling* function of the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ (typically equivalent to the distortion measure D), and $\varphi_{D\max}$ is the maximum value of φ_D for the dataset. This form of f_C ensures that the penalty for violating a cannot-link constraint remains non-negative. Note that the resulting potential function V corresponds to a metric version of previously described generalized Potts potential (Kleinberg & Tardos, 1999).

The overall posterior probability of a cluster label configuration $\mathcal{L}$ is $\mathbf{Pr}(\mathcal{L}|\mathcal{X}) \propto \mathbf{Pr}(\mathcal{L})\mathbf{Pr}(\mathcal{X}|\mathcal{L})$, assuming $\mathbf{Pr}(\mathcal{X})$ to be constant. We consider $\mathbf{Pr}(\mathcal{X}|\mathcal{L})$ to have an exponential form, which encompasses most commonly used distortion measures such as squared Euclidean distance, KL divergence, and cosine similarity (Basu et al., 2004). Finding the maximum *a posteriori* (MAP) configuration of the HMRF then becomes equivalent to maximizing the posterior probability:

$$\mathbf{Pr}(\mathcal{L}|\mathcal{X}) = \frac{1}{Z} e^{-\sum_i \sum_j V(i,j)} \cdot e^{-\sum_{x_i \in \mathcal{X}} D(\mathbf{x}_i, \mu_{l_i})} \quad (5)$$

where Z is a normalizing constant. Henceforth, we will refer to the first exponential factor of $\mathbf{Pr}(\mathcal{L}|\mathcal{X})$ as the *constraint potential*, the second factor as the *distance potential*, and the negative logarithm of $\mathbf{Pr}(\mathcal{L}|\mathcal{X})$ as the *posterior energy*. Note that MAP estimation would reduce to maximum likelihood (ML) estimation of $\mathbf{Pr}(\mathcal{X}|\mathcal{L})$ if $\mathbf{Pr}(\mathcal{L})$ is constant. However, because our model accounts for dependencies between the cluster labels and $\mathbf{Pr}(\mathcal{L})$ is not constant, full MAP estimation of $\mathbf{Pr}(\mathcal{L}|\mathcal{X})$ is required.

Taking logarithms of Eqn.(5) gives the following cluster objective function, minimizing which is equivalent to maximizing the MAP probability in Eqn.(5), or, equivalently, minimizing the posterior energy of the HMRF:

$$\begin{aligned}\mathcal{J}_{\text{obj}} = &\sum_{x_i \in \mathcal{X}} D(\mathbf{x}_i, \mu_{l_i}) + \sum_{(x_i,x_j)\in\mathcal{M}} w_{ij}\varphi_D(\mathbf{x}_i, \mathbf{x}_j)\mathbb{1}[l_i \neq l_j] \\ &+ \sum_{(x_i,x_j)\in\mathcal{C}} \overline{w}_{ij}\big(\varphi_{D\max} - \varphi_D(\mathbf{x}_i, \mathbf{x}_j)\big)\mathbb{1}[l_i = l_j] + \log Z \quad (6)\end{aligned}$$

where Z is a normalizing constant. Thus, the task is to minimize $\mathcal{J}_{\text{obj}}$ over $\{\mu_h\}_{h=1}^K$, $\mathcal{L}$, and D (if the latter is parameterized). For computational efficiency, we consider $\log Z$ to be constant during the clustering iterations.

The *Expectation-Maximization* (EM) algorithm is a popular solution to such "incomplete data" problems (Dempster et al., 1977). It is well-known that K-Means is equivalent to an EM algorithm with hard clustering assignments (Kearns et al., 1997; Banerjee et al., 2004). Thus, we can use a K-Means-type iterative clustering algorithm, HMRF-KMEANS, to find a (local) maximum of the above function.

2.2. The HMRF-KMEANS Algorithm

The outline of the HMRF-KMEANS algorithm is presented in Figure 2. Initialization is performed using neighborhoods inferred from the constraints as described in (Bilenko et al., 2004). The basic idea of HMRF-KMEANS is as follows: in the E-step, given the current cluster representatives, all data points are collectively re-assigned to clusters to minimize $\mathcal{J}_{\text{obj}}$. In the M-step, the cluster representatives $\{\mu_h\}_{h=1}^K$ are re-estimated from the cluster assignments to minimize $\mathcal{J}_{\text{obj}}$ for the current assignment. The clustering distortion measure D is updated in the M-step to reduce the objective function simultaneously by transforming the space in which data lies, thus performing metric learning.

Note that this procedure is an instantiation of the generalized EM algorithm (Dempster et al., 1977; Neal & Hinton, 1998), where the objective function is reduced but not necessarily minimized in the M-step. Effectively, the E-step minimizes $\mathcal{J}_{\text{obj}}$ over cluster assignments $\mathcal{L}$, the M-step (A) minimizes $\mathcal{J}_{\text{obj}}$ over cluster representatives $\{\mu_h\}_{h=1}^K$, and the M-step (B) minimizes $\mathcal{J}_{\text{obj}}$ over the parameters of the distortion measure D. The E-step and the M-step are repeated till a specified convergence criterion is reached.

```
Algorithm: HMRF-KMeans
Input: Set of data points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, number of clusters $K$,
    set of must-link constraints $\mathcal{M} = \{(\mathbf{x}_i, \mathbf{x}_j)\}$,
    set of cannot-link constraints $\mathcal{C} = \{(\mathbf{x}_i, \mathbf{x}_j)\}$,
    distance measure $D$, constraint violation costs $\mathcal{W}$ and $\overline{\mathcal{W}}$.
Output: Disjoint $K$-partitioning $\{\mathcal{X}_h\}_{h=1}^K$ of $\mathcal{X}$ such that
    objective function $\mathcal{J}_{\text{obj}}$ in Eqn.(6) is (locally) minimized.
Method:
1. Initialize the $K$ clusters centroids $\{\mu_h^{(0)}\}_{h=1}^K$, set t ← 0
2. Repeat until convergence
2a.  E-step: Given $\{\mu_h^{(t)}\}_{h=1}^K$, re-assign cluster labels
       $\{l_i^{(t+1)}\}_{i=1}^N$ on the points $\{\mathbf{x}_i\}_{i=1}^N$ to minimize $\mathcal{J}_{\text{obj}}$.
2b.  M-step(A): Given cluster labels $\{l_i^{(t+1)}\}_{i=1}^N$, re-calculate
       cluster centroids $\{\mu_h^{(t+1)}\}_{h=1}^K$ to minimize $\mathcal{J}_{\text{obj}}$.
2c.  M-step(B): Re-estimate distance measure $D$ to reduce $\mathcal{J}_{\text{obj}}$.
2d.  t ← t+1
```

Figure 2. The HMRF-KMEANS algorithm

The relational nature of the supervision is significant in the E-step, where the task is to assign data points to clusters using the current estimates of the cluster representatives. In simple K-Means there is no interaction between the cluster labels, and the E-step is a simple assignment of every point to the cluster representative that is nearest to it according to the clustering distortion measure. In contrast, the HMRF model incorporates interaction between the cluster labels defined by the random field over the hidden variables. Thus, optimal assignment in the E-step of HMRF-KMEANS is a relational inference problem.

3. Inference Techniques

Below we describe three inference methods for collective assignment of data points to clusters in the E-step of HMRF-KMEANS.

3.1. The Belief Propagation Approach

A global joint assignment of the points to clusters that (locally) minimizes the objective function $\mathcal{J}_{\text{obj}}$ can be found

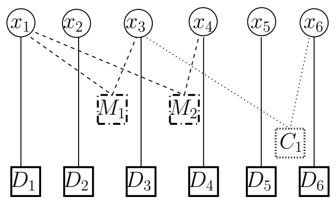


Figure 3. The Factor Graph for the HMRF in Figure 1

by performing approximate inference on the HMRF using belief propagation (Pearl, 1988). This approach is similar to the technique used by Segal et al. (2003).

To implement the message passing algorithm for approximate inference on the HMRF, we represent the HMRF as a factor graph model (Kschischang et al., 2001). The sum-product/max-product algorithm on the factor graph model has been shown to be a generalization of several well known inference algorithms on graphical models. Interpreting the HMRF model as a factor graph enables us to perform belief propagation on the HMRF using the max-product message passing algorithm on the corresponding factor graph.

The factor graph corresponding to the example HMRF in Figure 1 is shown in Figure 3. The factor graph has the following components:

(1) N variable nodes $\{\mathbf{x}_i\}_{i=1}^N$ representing the data points.

(2) N factor nodes $\{D_i\}_{i=1}^N$ that encode the distance potential components of the objective function. Each distance factor node D_i has an edge connecting it to the corresponding variable node $\mathbf{x}_i$, and a table containing different values of the distance potential function. This table has an entry for each possible cluster assignment of the variable; the j^{th} entry is $\exp(-d)$, where d is the distance from the i^{th} point to the j^{th} cluster.

(3) $|\mathcal{M}|$ factor nodes $\{M_i\}_{i=1}^{|\mathcal{M}|}$ and $|\mathcal{C}|$ factor nodes $\{C_i\}_{i=1}^{|\mathcal{C}|}$, which encode the cost of violating the must-link and cannot-link constraints, respectively. There is one factor node for each constraint, which is linked by edges to the 2 variable nodes involved in that constraint.

The constraint potential table associated with each constraint factor node maps a set of K^2 value-pairs (corresponding to possible cluster assignments to the pair of points in the constraint) to potential values. For the factor node encoding the must-link constraint $(\mathbf{x}_i, \mathbf{x}_j)$, the potential value for the entry (l_i, l_j) in the constraint potential table is 1 if $l_i = l_j$, i.e., $\mathbf{x}_i$ and $\mathbf{x}_j$ have the same cluster assignments. If $l_i \neq l_j$, the potential value is $\exp(-d(\mathbf{x}_i, \mathbf{x}_j)w_{ij})$, where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the distance between the points $\mathbf{x}_i$ and $\mathbf{x}_j$ according to the current estimate of the distortion measure D, and w_{ij} is the weight of the constraint.

Similarly, for the cannot-link factor nodes, the potential tables have values of 1 for the entry (l_i, l_j) where $l_i \neq l_j$, and $\exp(-(d_{\max}(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{x}_i, \mathbf{x}_j))\overline{w}_{ij})$ if $l_i = l_j$. The potential values of the constraint factor nodes correspond to the metric version of the Potts potential function, as explained in Section 2.1.

Finding the collective assignment of points to minimize $\mathcal{J}_{\text{obj}}$ in the E-step corresponds to running the max-product message-passing algorithm on the factor graph (Kschischang et al., 2001). Once the message-passing algorithm converges, the cluster assignment for each data point is obtained from the value in the corresponding variable node.

3.2. The Iterated Conditional Modes Approach

The Iterated Conditional Modes (ICM) inference technique (Besag, 1986) is a greedy strategy to sequentially update the cluster assignment of each point, keeping the assignments for the other points fixed. Based on a preselected random ordering, each point $\mathbf{x}_i$ is sequentially assigned to the cluster representative μ_h that minimizes the point's contribution to the objective function $\mathcal{J}_{obj}(\mathbf{x}_i, \mu_h)$:

$$\mathcal{J}_{\text{obj}}(\mathbf{x}_i, \mu_h) = D(\mathbf{x}_i, \mu_h) + \sum_{(x_i,x_j)\in\mathcal{M}} w_{ij}\varphi_D(\mathbf{x}_i, \mathbf{x}_j)\mathbb{1}[h \neq l_j] + \sum_{(x_i,x_j)\in\mathcal{C}} \overline{w}_{ij}\big(\varphi_{D\max} - \varphi_D(\mathbf{x}_i, \mathbf{x}_j)\big)\mathbb{1}[h = l_j] \quad (7)$$

Optimal assignment for every point is that which minimizes the distortion between the point and its cluster representative (first term of $\mathcal{J}_{\text{obj}}$) along with incurring a minimal penalty for constraint violations caused by this assignment (second and third terms of $\mathcal{J}_{\text{obj}}$). Once a point $\mathbf{x}$ is assigned to a cluster, the subsequent points in the sequence determined by the ordering use the current cluster assignment of $\mathbf{x}$ to calculate possible constraint violations.

After all points are assigned, the assignment process is repeated according to a new random ordering. This process proceeds until no point changes its cluster assignment between two successive iterations. ICM is guaranteed to reduce $\mathcal{J}_{\text{obj}}$ or keep it unchanged (if $\mathcal{J}_{\text{obj}}$ is already at a local minimum) in the E-step (Besag, 1986).

3.3. The LP Relaxation Approach

The task of finding an assignment of instances to clusters to minimize the objective function can be posed as an integer programming problem. Such a formulation has been proposed by Kleinberg and Tardos in the context of the general *metric labeling* problem, where they considered the cost of assigning labels to instances while attempting to satisfy a set of must-link pairwise constraints (Kleinberg & Tardos, 1999). We extend this formulation to include cannot-link constraints, which allows using it for assigning instances to clusters in the E-step of HMRF-KMEANS.

Let $\mathcal{Y} = \{y_{il}\}$, $i = 1..N$, $l = 1..K$, be a set of nonnegative binary variables that encode membership of instances in clusters: $y_{il} = 1$ signifies that the i^{th} instance belongs to the l^{th} cluster. Sets of nonnegative binary variables $\mathcal{Y}^{(\mathcal{M})} = \{y_i^{(M)}\}_{i=1}^{|\mathcal{M}|}$ and $\mathcal{Y}^{(\mathcal{C})} = \{y_i^{(C)}\}_{i=1}^{|\mathcal{C}|}$ encode violations of must-link and cannot-link pairwise constraints respectively. Each $y_k^{(M)} = 1$ signifies that the k^{th} must-link pairwise constraint $e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2})$ is violated, while $y_k^{(C)} = 1$ signifies that the k^{th} cannot-link pairwise constraint $e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2})$ is violated. The objective function to be optimized in the E-step of HMRF-KMEANS then becomes:

$$\mathcal{J}_{\text{obj}} = \sum_{\mathbf{x}_i \in \mathcal{X}} \sum_{l \in \mathcal{L}} D(\mathbf{x}_i, \mu_l)\, y_{il} + \sum_{(\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) \in \mathcal{M}} w_k f_M(\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) y_k^{(M)} + \sum_{(\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) \in \mathcal{C}} \overline{w}_k f_C(\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) y_k^{(C)} \tag{8}$$

Assigning each instance to only one cluster imposes the following linear constraint on variables in $\mathcal{Y}$:

$$\sum_{l \in \mathcal{L}} y_{il} = 1, \quad \mathbf{x}_i \in \mathcal{X} \tag{9}$$

Also, consistency of pairwise constraint violation variables in $\mathcal{Y}^{(\mathcal{M})}$ and $\mathcal{Y}^{(\mathcal{C})}$ with the assignment variables in $\mathcal{Y}$ requires satisfaction of the following linear constraints:

$$\begin{aligned} y_k^{(M)} &= \frac{1}{2} \sum_{l \in \mathcal{L}} |y_{k_1 l} - y_{k_2 l}|, & e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) \in \mathcal{M}; \\ y_k^{(C)} &= 1 - \frac{1}{2} \sum_{l \in \mathcal{L}} |y_{k_1 l} - y_{k_2 l}|, & e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) \in \mathcal{C} \end{aligned} \tag{10}$$

These constraints can be expressed in a linear program by replacing variables $\mathcal{Y}^{(\mathcal{M})}$ and $\mathcal{Y}^{(\mathcal{C})}$ with corresponding sets of auxiliary variables $\mathcal{Z}^{(\mathcal{M})}$ and $\mathcal{Z}^{(\mathcal{C})}$, where $z_{kl}^{(M)} = 1$ iff the k^{th} must-link pair $e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2})$ is violated and either $\mathbf{x}_{k_1}$ or $\mathbf{x}_{k_2}$ is assigned to l^{th} cluster. Semantics of $z_{kl}^{(C)}$ are similar: $z_{kl}^{(C)} = 1$ iff k^{th} cannot-link pair $e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2})$ is violated and both $\mathbf{x}_{k_1}$ and $\mathbf{x}_{k_2}$ are assigned to l^{th} cluster. Variables in $\mathcal{Y}^{(\mathcal{M})}$ and $\mathcal{Y}^{(\mathcal{C})}$ can be expressed via variables in $\mathcal{Z}^{(\mathcal{M})}$ and $\mathcal{Z}^{(\mathcal{C})}$ as follows:

$$\begin{aligned} y_k^{(M)} &= \frac{1}{2} \sum_{l \in \mathcal{L}} z_{kl}^{(M)}, & e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) \in \mathcal{M} \\ y_k^{(C)} &= \sum_{l \in \mathcal{L}} z_{kl}^{(C)}, & e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) \in \mathcal{C} \end{aligned} \tag{11}$$

Consistency of assignment variables in $\mathcal{Y}$ with pairwise constraint violation variables in $\mathcal{Z}^{(\mathcal{M})}$ and $\mathcal{Z}^{(\mathcal{C})}$ can then be achieved by introducing the following linear constraints:

$$z_{kl}^{(M)} \geq y_{k_1 l} - y_{k_2 l}, \quad e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) \in \mathcal{M} \tag{12}$$

$$z_{kl}^{(M)} \geq y_{k_2 l} - y_{k_1 l}, \quad e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) \in \mathcal{M} \tag{13}$$

$$z_{kl}^{(C)} \leq y_{k_1 l} + y_{k_2 l}, \quad e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) \in \mathcal{C} \tag{14}$$

$$z_{kl}^{(C)} \geq y_{k_1 l} + y_{k_2 l} - 1, \quad e_k = (\mathbf{x}_{k_1}, \mathbf{x}_{k_2}) \in \mathcal{C} \tag{15}$$

Minimization of objective function (8) under constraints (9) and (12)-(15) to solve for binary variables $\mathcal{Y}$, $\mathcal{Z}^{(\mathcal{M})}$, and $\mathcal{Z}^{(\mathcal{C})}$ is NP-hard. Kleinberg and Tardos proposed a linear programming relaxation of this integer programming problem by allowing $\mathcal{Y}$, $\mathcal{Z}^{(\mathcal{M})}$, and $\mathcal{Z}^{(\mathcal{C})}$ to be nonnegative real numbers, and provided a randomized method for rounding the real solution to the linear program to integers (Kleinberg & Tardos, 1999). We follow their approach, which allows us to perform collective assignment of all instances in $\mathcal{X}$ to cluster centroids.

4. Experiments

4.1. Methodology and Datasets

Experiments were conducted on three datasets: *Iris* from the UCI repository, the *Protein* dataset used by Xing et al. (2003) and Bar-Hillel et al. (2003), and a randomly sampled subset from the *Letters* handwritten character recognition dataset. For *Letters*, we chose three classes: $\{$**I, J, L**$\}$, sampling 10% of the data points from the original dataset randomly. We used parameterized squared Euclidean distance as the clustering distortion measure D for these experiments.

We used pairwise F-Measure to evaluate the clustering results based on the underlying classes. F-Measure relies on the traditional information retrieval measures, adapted for evaluating clustering by considering same-cluster pairs:

$$Precision = \frac{\#PairsCorrectlyPredictedInSameCluster}{\#TotalPairsPredictedInSameCluster}$$

$$Recall = \frac{\#PairsCorrectlyPredictedInSameCluster}{\#TotalPairsInSameCluster}$$

$$F\text{–}Measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

We generated learning curves with 2-fold cross-validation for each dataset. Each point on the learning curve represents a particular number of randomly selected pairwise constraints given as input to the algorithm. Unit constraint costs were used for all constraints, since the datasets did not provide individual weights for the constraints. The clustering algorithm was run on the whole dataset, but the pairwise F-Measure was calculated only on the test set. Results were averaged over 10 runs of 2 folds. For each trial, cluster initialization was performed using neighborhoods inferred from the provided constraints (Bilenko et al., 2004), and then the HMRF-KMEANS algorithm was run with a particular inference technique in the E-step, and metric learning for Euclidean distance in the M-step.

4.2. Results and Discussion

We compared the three methods described in Section 3 for collective assignment of instances to clusters. Figures 4-6 show learning curves for the three datasets. For each

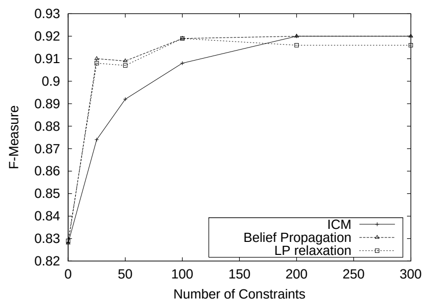


Figure 4. Iris results

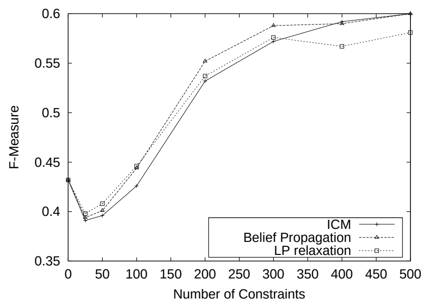


Figure 5. Protein results

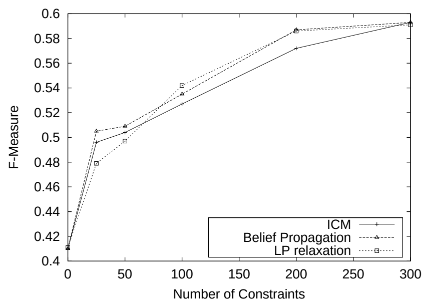


Figure 6. Letters results

dataset, ICM was faster than belief propagation and LP relaxation by at least an order of magnitude, which agrees with the relative computational complexities of these algorithms.

As the results demonstrate, global relational methods such as belief propagation and LP relaxation outperform the greedy approaches when a limited number of pairwise constraints is provided. However, as the number of provided constraints increases, returns from these computationally expensive methods diminish, and for every dataset there exists a number of constraints beyond which ICM performs no worse than the global approximate inference methods.

5. Conclusions

We have compared two methods for global approximate inference (belief propagation and LP relaxation) with a greedy approximate inference algorithm (ICM) in the context of collective assignment of data points to clusters in semi-supervised clustering with pairwise relational constraints. Our results indicate that belief propagation and LP relaxation outperform ICM when a limited number of pairwise constraints is provided. However, given a sufficiently large amount of relational supervision, the greedy algorithm for approximate inference performs on par with global methods. Thus, greedy inference techniques should be considered for scaling up semi-supervised clustering to large datasets due to their low computational cost.

6. Acknowledgments

We would like to thank Razvan Bunescu for helpful discussions. This research was supported in part by NSF grants IIS-0325116 and IIS-0117308.

References

Banerjee, A., Merugu, S., Dhillon, I. S., & Ghosh, J. (2004). Clustering with Bregman divergences. *Proceedings of SDM-2004*.

Bar-Hillel, A., Hertz, T., Shental, N., & Weinshall, D. (2003). Learning distance functions using equivalence relations. *Proceedings of ICML-2003*, (pp. 11–18).

Basu, S., Bilenko, M., & Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. In submission, available at `http://www.cs.utexas.edu/~ml/publication`.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B (Methodological)*, *48*, 259–302.

Bilenko, M., Basu, S., & Mooney, R. J. (2004). Integrating constraints and metric learning in semi-supervised clustering. To appear in *Proceedings of ICML-2004*.

Cohn, D., Caruana, R., & McCallum, A. (2003). *Semi-supervised clustering with user feedback* (Technical Report TR2003-1892). Cornell University.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–38.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE PAMI*, *6*, 721–742.

Kearns, M., Mansour, Y., & Ng, A. Y. (1997). An information-theoretic analysis of hard and soft assignment methods for clustering. *Proceedings of UAI-1997* (pp. 282–293).

Klein, D., Kamvar, S. D., & Manning, C. (2002). From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. *Proceedings of ICML-2002* (pp. 307–314).

Kleinberg, J., & Tardos, E. (1999). Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *Proceedings of FOCS-1999* (pp. 14–23).

Kschischang, F. R., Frey, B., & Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*, 498–519.

Neal, R. M., & Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models* (pp. 355–368). MIT Press.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Segal, E., Wang, H., & Koller, D. (2003). Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics*, *19*, i264–i272.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained K-Means clustering with background knowledge. *Proceedings of ICML-2001* (pp. 577–584).

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. *NIPS 15* (pp. 505–512).

Zhang, Y., Brady, M., & Smith, S. (2001). Hidden Markov random field model and segmentation of brain MR images. *IEEE Transactions on Medical Imaging*, *20*, 45–57.

Using neural networks for relational learning

Hendrik Blockeel HENDRIK.BLOCKEEL@CS.KULEUVEN.AC.BE
Werner Uwents
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract

Relational learners need to be able to handle the information contained in a set of related tuples. Most current relational learners are biased either towards the use of aggregate functions that summarize that set, or towards checking the existence of specific kinds of elements in that set. Learning patterns that contain a combination of both is a challenging task. In this paper we introduce a neural networks based approach to relational learning, where the neural net that is learned can actually represent such a combination. This capacity is illustrated on toy problems, but several questions are open with respect to learnability of more complicated concepts.

1. Introduction

Non-relational ("propositional") learners can be said to learn a function $f(x)$ where the domain of x is a cartesian product of different domains. That is, x is described by a fixed number of attributes, for each of which a single value is given; or, in yet other words, x is a single tuple.

In relational learning, a single instance is described by a tuple together with a set of other tuples related (linked) to it. We will adopt the terminology from relational databases here. A database contains a set of relations, each of which is a set of tuples. Tuples may be linked to each other through foreign key relationships. Thus, if we have relations R and S in a relational database, a tuple r of R forms a propositional description of some entity, whereas r together the set of all $s \in S$ that are linked to r, forms a relational description of the same entity.

Assuming we disallow set-valued attributes, the information contained in a set of related tuples can in practice not be represented accurately as a single tuple (for instance, because its size is unbounded), it can only be approximated. We can then distinguish two approaches: those that describe the set with a number of features that is chosen in advance (e.g., its cardinality) and then learn from a propositional representation (these are also called "propositionalization approaches"), and those that search a potentially infinite space of features for the most relevant ones; this feature construction is then part of the learning process.

In relational algebra terminology, the features that are used to describe sets are usually of the general form $\mathcal{F}(\sigma_C(S))$ where $\mathcal{F}$ is some kind of aggregation function, C is a condition that elements of S may or may not fulfill, and $\sigma_C(S)$ is the set of all elements of S that fulfill C. The set S, when classifying a tuple t, is obtained by joining t with any other tuples to which it is linked, joining those tuples again with linked tuples, and so on, up to a certain maximum number of consecutive joins.

Now it is instructive to look at how relational learners fill in $\mathcal{F}$ and C. In propositionalization approaches, the feature itself, i.e., the combination of $\mathcal{F}$ and C, is defined in advance. Any combination can be identified by the user as potentially relevant. There is no automatic feature construction.

Other approaches construct features using some structured search. These typically focus their search on either $\sigma_C(S)$ or $\mathcal{F}$. Inductive logic programming (ILP) (Muggleton & De Raedt, 1994) algorithms typically construct a complex $\sigma_C(S)$, where S is several levels deep and C may be a complex conjunction of conditions; but $\mathcal{F}$ reduces to testing the emptiness of $\sigma_C(S)$ (logical clauses, as learned in ILP, are existentially quantified), in other words, $\mathcal{F}(X) = [count(X) > 0]$.

Yet other approaches, typically described in a relational databases framework, allow several predefined aggregation functions $\mathcal{F}$, but apply them only to S itself, not to a selection of S. Knobbe et al. (2002) are, to our knowledge, the first to present a method that performs a systematic search in a hypothesis space (in this case, that of "selection graphs") where hypotheses combine aggregation and selection. Their approach is however limited to monotone aggregation functions ($\mathcal{F}$ is monotone if $S \subseteq S' \Rightarrow \mathcal{F}(S) < \mathcal{F}(S')$), which limits its applicability somewhat (for instance, Sum and Average are not monotone), and to selecting aggregate functions from a limited set given by the user.

Finally, some approaches may learn any aggregation function, not only predefined ones; an example of these is presented by Perlich and Provost (2003), who classify tuples based on the distribution of linked tuples. These authors also present a classification of relational data mining tasks, stating that most current relational systems handle tasks in categories 1 or 2 (involving no aggregation or uni-dimensional aggregation). Categories 3 and 4 involve multi-dimensional aggregation (aggregating over several variables, of the same relation for category 3, of different relations for category 4). (They also have a category 5, not relevant for this discussion). ILP-like methods are the only ones that consider tasks up to level 4, albeit with strong limitations with respect to the aggregate functions that they allow. Knobbe et al. (2002) partially lift this limitation, by allowing any aggregate function that is a member of a user-defined set of monotone functions. To our knowledge, no approaches currently combine a complicated selection with less limited kinds of aggregation.

In this paper, we explore the use of neural networks for relational learning. The approach we propose, does combine complicated, non-predefined, selections and aggregations, and hence can be positioned in category 4. It can be considered a supervised learning approach, where aggregations and selections are learned in parallel, tuned to each other in order to provide maximal information with respect to the tuple to be classified. This paper presents preliminary work, showing the promise of the approach, but leaving many questions open. We define relational neural networks in Section 2, present some experiments in Section 3, and conclude in Section 4.

2. Relational Neural Nets

A standard feedforward neural network has a fixed number of inputs, all with a well-distinguished effect on the output; in other words, it represents a function $f(x)$ where x is a single tuple. In order to use a neural network for relational learning, we need to extend it with a capacity for handling sets. More specifically, for a given instance, we need to be able to feed a set of tuples into the network for any one-to-many relationship that the instance participates in.

We first discuss how we can handle a set of inputs using a recurrent neural network; then we show how a relational neural network structure can be derived from a relational database schema. Finally we compare this approach to some other approaches that use neural networks for relational learning.

2.1. Handling set inputs using recurrent neural networks

A recurrent neural network is a standard feedforward neural network in which feedback loops are introduced; that is, the outputs of certain neurons are fed back into their inputs (either directly or indirectly).

For 2-layer recurrent neural networks, three architectures are typically distinguished:

- Elman: the output of each layer 1 (hidden layer) node is fed back into each layer 1 node
- Jordan: the output of each layer 2 node is fed back into each layer 1 node
- Willams-Zipfer: all layer 1 and layer 2 outputs are fed back into all layer 1 nodes (thus combining the Elman and Jordan architectures).

The feedback loops give the network a kind of memory, which allows one to feed multiple input vectors into the network for a given training example. Recurrent networks are typically used for time series, where the prediction at a given time point may depend not only on inputs for that time point but also on previous inputs. The input vectors at times $t-k$, $t-k+1$, …, $t-1$, t are then typically fed into the network one after another and the output produced by the network after seeing these inputs is compared with the desired output y_t at time t. Reduction of the difference between y_t and the network output can be done using a gradient descent procedure that can be implemented using a variant of the standard backpropagation algorithm. A typical method used is backpropagation through time, where the recurrent network is unfolded into a multilayer feedforward neural network to which standard backpropagation is applied.

While recurrent neural networks are often used for time series prediction, they appear to be less commonly used for handling sets of input vectors. The main difference between these two is that in time series prediction, the input for a given instance is an ordered set of vectors, rather than an unordered set (which is what we need here). Obviously, any function of an unordered set can be represented as a function of an ordered set, but the converse does not hold. The fact that we want to learn an order-invariant function could in principle be used to constrain the neural net's parameters so that only order-invariant functions can be learnt. At this stage we have not investigated exactly how this could be done.

2.2. Relational neural networks

Assume we have a relational database schema where R_T is the target relation, and $R_1, \ldots, R_n$ are other relations in the database. We denote the attribute sets of R_i by U_i. In the following we will assume a classification setting for ease of discussion, but everything applies equally well to any predictive setting.

Given any relation R, we define

- $S_1(R)$: $R_i \in S_1(R)$ iff each tuple $t \in R$ is connected to exactly one tuple in R_i (i.e., there is a one-to-one or many-to-one relationship between R and R_i, in which R participates completely)

- $S_{01}(R)$: $R_i \in S_{01}(R)$ iff each tuple $t \in R$ is connected to at most one tuple in R_i (again one-to-one or many-to-one, but partial participation)
- $S_n(R)$: $R_i \in S_n(R)$ iff each tuple $t \in R$ is connected to zero, one, or more tuples in R_i (partial or complete participation of R in a one-to-many or many-to-many relationship with R_i)
- $S_u(R)$: $R_i \in S_u$ iff R_i is a relation of the relational database not in $S_1(R)$, $S_{01}(R)$ or $S_n(R)$ (i.e., it is not directly connected to R)

Given a tuple t in the target relation R_T, we want to classify it based on the information contained in the tuple and in any tuples linked to this tuple. For a relation R_i we use U_i to denote the original attribute set of that relation and I_i to denote the attribute set actually used as input to our classifier; one might expect $I_i = U_i$ but there will be some small differences. For the target table, $I_T = U_T - \{C\}$ where C is the class attribute.

For any tuples $t_i \in R_i$ where $R_i \in S_1(R_T)$, the information in t_i can be added to t by joining R_i with $\{t\}$: a single tuple is thus obtained. $I_i = U_i$ for these tuples.

For any $R_i \in S_{01}(R_T)$, an outer join with $\{t\}$ also yields exactly one tuple, which may however contain null values for the attributes of R_i. As neural networks do not have a distinguished encoding for null values, we will use an extra attribute E_i that indicates whether the link to R_i yielded a joining tuple or not. $I_i = U_i \cup \{E_i\}$.

For $R_i \in S_n(R_T)$, an indefinite number of tuples t_{i1}, $\ldots$, t_{in} may be linked to a single $t \in R_T$. In order to allow this set of tuples to influence the classification of t, a recurrent neural network will be constructed. The tuples t_{ij} will be represented using their original attribute values plus an extra attribute E_i which is always true for these tuples. The fact that no tuples t_{ij} exist will be indicated using a single input tuple where $E_i = false$.

Tuples in $R_i \in S_u(R_T)$ are not directly linked to tuples in R_T, but may be linked indirectly. For now we ignore these.

Based on the above, we can propose several options for constructing a relational neural network that classifies $t \in R_T$ based on its own attribute values as well as those of (directly) related tuples. Two options that we will explore here, are:

Option 1: For each relation involved, including the target relation, a different neural network (called a "component") is constructed. The outputs of these networks are combined by a single perceptron. The components are standard feedforward neural networks for the target relation R_T and for any relations in $S_1(R_T)$ and $S_{01}(R_T)$, with as inputs the I_i as defined above. The components for $S_n(R_T)$ are recurrent neural networks.

Option 2: For each t, we can construct a tuple t' with attributes

$$I_T \cup (\bigcup_{i:R_i \in S_1(R_T)} I_i) \cup (\bigcup_{i:R_i \in S_{01}(R_T)} I_i) \cup (\bigcup_{i:R_i \in S_n(R_T)} O_i)$$

with I_i as defined above, and O_i a set of attributes the values of which are the outputs of a 2-layer recurrent neural network with as inputs U_i.

Of course, other options are possible as well, but we restrict ourselves in this paper to these two. Of these two, Option 1 is the more restricted one: a simpler network is constructed in which the results of different components are combined by one perceptron. Option 2 creates a larger and more expressive network; it also has an extra structural parameter, namely the number of outputs $|O_i|$ of the recurrent networks (which we assume here to be the same for all recurrent networks).

Note that the O_i attributes in Option 2 can be seen as aggregate functions that summarize the set of R_i-tuples related to t. Thus, this approach is closer to the propositionalization approach, where a propositional learner learns from a fixed set of attributes, some of which summarize set information. It differs from classical propositionalization approaches in that these aggregates are now learned, instead of predefined.

Note that with Option 2, the technique of adding to t the O_i attributes that summarize related tuples, can be repeated for those related tuples, thus also incorporating information in indirectly linked tuples (from relations in $S_u(R_T)$).

2.3. Related work

The existing work that is probably closest to our approach, is a line of work in the neural networks community on learning from structured data using recursive neural networks or folding architecture networks (Goller & Küchler, 1996; Sperduti & Starita, 1997; Frasconi et al., 1998). These authors describe how to learn from structured data (e.g., logical terms, trees, graphs), and discuss tasks like the identification of substructures, but they do not aim at learning aggregation functions. Those tasks relate to the tasks we consider, more or less in the same way as ILP relates to our approach. Our approach however is not essentially different, and many existing results on learnability of recursive neural networks may carry over to our setting.

A number of neural network based approaches have been defined in the ILP setting (Botta et al., 1997; Basilio et al., 2001); the neural networks in these ap-

ID	Gender	Father	Mother
albert	M	bob	alice
peter	M	bob	alice
alice	F	jack	mary
bob	M	james	anne
...	...	...	...

Table 1. An example instance of the family database.

proaches typically mimic logical inference as it would be made by logic programs or implement numerical computations in them. Again, they do not learn aggregate functions as we do. Ramon and De Raedt (2000) have defined multi-instance neural networks; these are a subset of our relational neural networks designed specifically for the multi-instance case. Ramon et al.'s neural logic programs (Ramon et al., 2002) are somewhat similar to our relational neural networks, with as main differences that they are described in a first order logic framework and that, just like for multi-instance neural networks, specific aggregate functions are encoded in advance by the user, instead of learned (and typically they represent logical conjunctions and disjunctions). We believe that from the point of view of relational learning, the ability to learn aggregate functions is a crucial advantage of our approach.

3. Experiments

To evaluate the potential of this approach, we have performed a number of experiments, varying parameters along a number of dimensions: Option 1 versus Option 2, as discussed above; different architectures for the recurrent components (Elman, Jordan, Williams-Zipfer); different learning approaches (backpropagation through time, evolutionary learning), different parameter settings for these approaches.

3.1. A family database

In this artificially generated toy database, there is a single relation with attributes ID, Gender, Father, Mother; it describes a number of persons and parentship relations between them (Father and Mother are foreign keys to ID). We show an example relation in Table 1. Note that the ID, Father and Mother attributes are present only to identify tuples and to link them to each other; they are not descriptive attributes of which the value will be tested by a hypothesis.

For this toy database, we define a few simple concepts. Given a tuple t, let $S(t)$ be the set of tuples s for which $s.Father = t.ID$ or $s.Mother = t.ID$.

- C1: has at least one son;
  $C1(t) \Leftrightarrow count(\sigma_{Gender=M}(S(t))) > 0$
- C2: has 2 children; $C2(t) \Leftrightarrow count(S(t)) = 2$
- C3: has 2 sons;
  $C3(t) \Leftrightarrow count(\sigma_{Gender=M}(S(t))) = 2$
- C4: has one son and one daughter;
  $C4(t) \Leftrightarrow (count(\sigma_{Gender=M}(S(t))) = 1 \wedge count(\sigma_{Gender=F}(S(t))) = 1)$
- C5: has one son or two daughters;
  $C5(t) \Leftrightarrow count(\sigma_{Gender=F}(S(t))) = 1 \vee count(\sigma_{Gender=M}(S(t))) = 2)$
- C6: has a grandson; $C6(t) \Leftrightarrow count(\{s \in S(t) | count\sigma_{Gender=F}(S(s))) > 0\}) > 0$

Note that C1 is a concept that could be learned by any ILP system: it involves some selection criterion but a trivial aggregation function. C2 is a concept that could easily be learned by any system where aggregates over sets of related tuples are predefined. Concept C3 is a more complicated concept that involves an aggregate over a selection of related tuples. C4 is again an ILP-like concept, but a slightly more complicated one than C1. Similarly, C5 is a slightly more complicated concept than C3, combining two functions that themselves combine aggregation and selection. C6 is an example of a concept that takes into account information "two steps away" from the tuple to be classified.

Most existing systems can learn these only with a relatively strong bias; e.g., in an ILP system typically the allowed aggregate function as well as any conditions that are allowed in the argument of this aggregate function would have to be given in advance.

The experimental setting is as follows: each combination of relational (Option 1, Option 2) and recurrent (Elman, Jordan, Williams-Zipfer) architecture is tried on all concepts C1-C6. For each architecture-concept combination, five runs with different random initializations of the network weights are made. Each run uses the same training set, which consists of 2/3 of the dataset. Of these five randomized runs, the one with highest training accuracy is chosen and evaluated on the remaining, unseen, examples (1/3 of the dataset). (Using a separate validation set would allow to separate the overfitting risk of an architecture from its true generalization power; for these initial exploratory experiments we found it unnecessary to have this separation, but it would be desirable for more sophisticated experiments.)

It turned out that for both relational architectures, using backpropagation through time, the network is able to learn correctly (with 100% test set accuracy) all the concepts. In general, some tuning of the network parameters was necessary for this, which is not unexpected; and usually not all random initializations of the network allowed it to converge to the correct concept. The results for evolutionary learning were similar.

3.2. Other Datasets

We have also attempted to evaluate the proposed approach on two benchmark datasets: the Musk dataset (Dietterich et al., 1997), a multi-instance problem on which several relational learning approaches have been compared, and the Financial dataset (Berka, 2000) for which also Knobbe et al. (2002) have reported results.

Due to inefficiencies in our current implementation, we have been able to evaluate the approach only on relatively small data sets up till now. For the Musk experiments, we have used Musk-1, the smaller of the two available data sets. For the Financial data set, we have not been able to include the Transaction relation in the training data, because it is too large. This means that for this data set our results are not comparable to results published elsewhere.

For the Musk-1 data set, we performed a tenfold cross-validation with the same folds that were used elsewhere. Four different combinations of parameters and architectures were tried, and predictive accuracies on unseen data varied between 79.3% (19 errors) and 85.9% (13 errors). This is to be compared with 88% (11 errors) reported for "multi-instance neural networks" (MINNs) (Ramon & De Raedt, 2000) approach, which are a special case of our relational neural networks.

Relational neural networks cannot outperform MINNs on this type of problems, since the hypothesis space searched by the former, H_{RNN}, is a superset of the hypothesis space H_{MINN} searched by the latter, and any hypothesis in $H_{RNN} - H_{MINN}$ is necessarily meaningless because it violates the multi-instance assumptions. In other words, the hypothesis in H_{RNN} that best approximates the target hypothesis, must also be in H_{MINN}. The best we can hope for, is that our approach performs as well as multi-instance neural networks on multi-instance problems (while performing better outside that class). From that point of view, the obtained results are promising, but more experimentation is needed to obtain more insight into, e.g., the probability of obtaining with our more general approach an accuracy that is comparable to that of the more specialized approach.

For the Financial dataset, 2/3 of the data were used for training and 1/3 for testing, and six different configurations were tried. Obtained accuracies were between 74% and 85%. This data set has a highly skewed class distribution: the frequency of the majority class is 85%. Thus, accuracy-wise, these results are quite bad. A ROC analysis, however, shows that on average the method does perform better than random guessing, in the best case combining a true positive rate of 0.95 with a false positive rate of 0.75. As mentioned, these results are not comparable with published results because only a subset of the available information was used.

4. Conclusions

We have presented a novel, neural network based, approach to relational learning. The approach consists of constructing a neural network based on the relational database schema, where recurrent components are introduced for one-to-many relationships. The power of this approach is yet to be determined, but this initial exploration reveals that, in Perlich and Provost's terminology (Perlich & Provost, 2003), simple category 3 and category 4 concepts (as defined for our toy database) can be accurately represented and learned. An experiment on a multi-instance benchmark further indicates that the approach may work equally well on this type of problems as the more specialized MINN approach (Ramon & De Raedt, 2000). An experiment on a fully relational problem has mainly revealed the need for a more efficient implementation: the computational complexity of the approach is relatively high and currently precludes a meaningful comparison of our approach to other approaches, on this benchmark.

Many questions remain open. What is the best architecture for these networks? We have proposed two options; our Option 2 is clearly more expressive than the other, but both appear to learn well on the problems we have considered. Among the three architectures for recurrent networks, there was a tendency of the Williams-Zipfer architecture to perform somewhat better than the others in our experiments, but again further experimentation is necessary.

What subclass of Perlich and Provost's category 3-4 can be learned using this approach? We have focused here on "combining selection and aggregation", which is a special case of multidimensional aggregation for which it is intuitively easier to see that our relational neural networks should be able to represent them. Even for this subclass, formal results would be desirable, and further experiments should be conducted.

Our recurrent neural networks in principle learn from ordered sets; they might be made to learn more effectively if their parameters are somehow constrained so that the network expresses a function of an unordered set. (Randomly permuting the elements of a set, each time the set is fed to the network, is also an option, but may not be the most efficient one.)

It is clear that there are connections between this approach and some existing approaches for learning from structured data that are not directly connected with statistical relational learning. We expect that some further investigation into those approaches may advance the field of relational learning.

5. Acknowledgements

Hendrik Blockeel is a post-doctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen). We thank Martijn van Otterlo, Kristian Kersting and Jan Ramon for useful suggestions concerning this work.

References

Basilio, R., Zaverucha, G., & Barbosa, V. C. (2001). Learning logic programs with neural networks. *Proceedings of the Eleventh International Conference on Inductive Logic Programming*. Springer-Verlag.

Berka, P. (2000). Guide to the financial data set. *The ECML/PKDD 2000 Discovery Challenge*.

Botta, M., Giordana, A., & Piola, R. (1997). Fonn: Combining first order logic with connectionist learning. *Proceedings of the 14th International Conference on Machine Learning* (pp. 46–56). Morgan Kaufmann.

Dietterich, T. G., Lathrop, R. H., & Lozano-Pérez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Frasconi, P., Gori, M., & Sperduti, A. (1998). A general framework for adaptive processing of data structures. *IEEE-NN*, *9*, 768–786.

Goller, C., & Küchler, A. (1996). Learning task-dependent distributed representations by back-propagation through structure. *Proceedings of the IEEE International Conference on Neural Networks (ICNN-96)* (pp. 347–352).

Knobbe, A., Siebes, A., & Marseille, B. (2002). Involving aggregate functions in multi-relational search. *Principles of Data Mining and Knowledge Discovery, Proceedings of the 6th European Conference* (pp. 287–298). Springer-Verlag.

Muggleton, S., & De Raedt, L. (1994). Inductive logic programming : Theory and methods. *Journal of Logic Programming*, *19,20*, 629–679.

Perlich, C., & Provost, F. (2003). Aggregation-based feature invention and relational concept classes. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 167–176). ACM Press.

Ramon, J., & De Raedt, L. (2000). Multi instance neural networks. *Proceedings of the ICML-Workshop on Attribute-Value and Relational Learning*.

Ramon, J., Driessens, K., & Demoen, B. (2002). Neural logic programs. Unpublished.

Sperduti, A., & Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, *8*, 714–735.

Learning Spatial Configuration Models Using Modified Dirichlet Priors

Matthew Boutell BOUTELL@CS.ROCHESTER.EDU
Christopher Brown BROWN@CS.ROCHESTER.EDU
Department of Computer Science, University of Rochester, Rochester, NY 14627 USA

Jiebo Luo JIEBO.LUO@KODAK.COM
Research and Development Labs, Eastman Kodak Company, Dewey Ave., Rochester, NY 14650 USA

Abstract

Semantic scene classification is a challenging problem in computer vision. Special-purpose semantic object and material (e.g., sky and grass) detectors help, but are faulty in practice. In this paper, we propose a generative model of outdoor scenes based on spatial configurations of objects in the scene. Because the number of semantically-meaningful regions (for classification purposes) in the image is expected to be small, we infer exact probabilities by utilizing a brute-force approach. However, it is impractical to obtain enough training data to learn the joint distribution of the configuration space.

To help overcome this problem, we propose a smoothing technique that modifies the naive uniform (Dirichlet) prior by using model-based graph-matching techniques to populate the configuration space. The proposed technique is inspired by the backoff technique from statistical language models. We compare scene classification performance using our method with two baselines: no smoothing and smoothing with a uniform prior. Initial results on a small set of natural images show the potential of the method. Detailed exploration of the behavior of the method on this set may lead to future improvements.

1. Introduction

Semantic scene classification, categorizing photographs at a high level into discrete categories such as *beach*, *mountain*, or *indoor*, is a useful, yet challenging problem in computer vision. It can help with image organization and with content-based image retrieval. Most approaches (Vailaya et al., 1999; Szummer & Picard, 1998; Torralba et al., 2003) typically use low-level (e.g., color, texture) features and classifiers to achieve reasonable results.

Higher-level features, such as the output from object and material detectors, can also help classify scenes. An advantage to this approach is its modularity, allowing independently-developed, domain-sensitive detectors to be used. Only recently has object and material detection in natural environments become accurate enough to consider using in a practical system. Recent work using object detection for other tasks (Mulhem et al., 2001; Song & Zhang, 2002; Smith et al., 2003) has achieved some success using object presence or absence alone as evidence. However, faulty detectors present a continuing difficulty for this approach.

Figure 1 shows an image, true material identities of key regions (color-coded), and simulated detector results, expressed as likelihoods that each region is labeled with a given material. The problem is how to determine which scene type best explains the observed (imperfect) evidence.

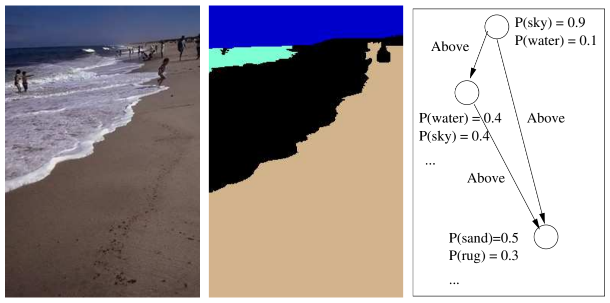


Figure 1. (a) A beach image (b) Its manually-labeled materials. The true configuration includes *sky above water*, *water above sand*, and *sky above sand*. (c) The underlying graph showing detector results and spatial relations.

How does one overcome detector errors? One principled way is to use a probabilistic inference system (vs. a rule-based one, such as (Mulhem et al., 2001)) to classify a scene based on the presence or absence of semantic materials. Furthermore, we can extract additional useful evidence from the input image, such as spatial relationships between the detected regions, to improve scene classification.

In this paper, we study statistical relational learning of *scene configurations*, consisting of both materials *and* their spatial relations. We propose a generative model of scene classification that uses material detectors and full scene configurations. The main limitation of this model is obtaining enough training data to learn the joint distribution of the *configuration space* (materials in specific configurations). To this end, we also propose a smoothing technique that improves upon the naive uniform (Dirichlet) prior by using model-based graph-matching techniques to populate the configuration space. Our technique is inspired by graph matching and backoff techniques. It is used in learning only; the inference phase needs no adaptation. We compare scene classification performance using our method with two baselines, no smoothing and a uniform prior, to show its promise.

2. Scene Configurations

Scene configurations consist of both the actual spatial arrangement (graph edge labels) of regions and the identities of those regions (node labels), as illustrated in Figure 1.

Our terminology is as follows: let n be the number of distinct regions detected in the image, M be the small set of semantically critical materials for which detectors are used, SR be the set of spatial relations, and C be the set of configurations of materials in a scene. Then an upper bound on the number of scene configurations, $|C|$, is $|M|^n|SR|^{\binom{n}{2}}$ (in a fully connected graph).

In (Luo et al., 2003), the spatial relations *above, below, far above, far below, beside, enclosed*, and *enclosing* (i.e., $|SR| = 7$)) were shown to be effective for spatially-aware material detection within outdoor scenes. We adopt essentially the same spatial relations in this study.

In the inference phase, the spatial arrangement of the test image is known (computed); thus, its graph need only be compared with those of training images with the same arrangement. We restrict our attention in this paper to learning the distribution of region identities within a *fixed* spatial arrangement, of which there are $|M|^n$ configurations. For example, an image with two regions, R_1 *above* R_2 has only $|M|^2$ configurations.

Adding to our terminology, we can formalize the scene classification problem as follows: let S be the set of scene classes considered, and $E = \{E_1, E_2, ...E_n\}$ be the detector evidence, where each $E_j = \{e_1, e_2, \ldots e_{|M|}\}$ is a likelihood vector for the identity of region j.

In this framework, we seek to calculate:

$$\arg\max_i P(S_i|E) \propto \arg\max_i P(S_i)P(E|S_i) \qquad (1)$$

Taking the joint distribution of $P(E|S_i)$ with C yields

$$\arg\max_i P(S_i)\sum_{c\in C} P(E, c|S_i) \qquad (2)$$

Conditioning on c gives

$$\arg\max_i P(S_i)\sum_{c\in C} P(E|c, S_i)P(c|S_i) \qquad (3)$$

2.1. Relation to Graphical Models

Figure 2a shows a graphical *representation* (not graphical model) for a single scene. While it is possible to represent this system with a grapical model, we chose a different approach in this study. On the surface, it looks like a standard two-level Markov Random Field (MRF) (Geman & Geman, 1984; Freeman et al., 2000). As in these MRFs, evidence nodes in our representation are conditionally dependent only on the identity of the underlying region's scene node, while the scene nodes are dependent on each other. However, this is not a typical graphical model. Fundamentally, we are solving a different problem than those for which MRFs are used. MRFs are typically used to regularize input data (Geman & Geman, 1984; Chou, 1988), finding $P(C|E)$, the single configuration (within a single model) that best explains the observed faulty evidence. In constrast, we are trying to perform *cross-model comparison*, $P(S|E)$, comparing how well the evidence matches each model in turn. To do this, we need to sum across *all possible* configurations of the scene nodes (Equation 3). In this framework, we need to use a brute force approach.

Further, at this coarse segmentation, even distant (in the underlying image) nodes may be strongly correlated, e.g., sky and pavement in urban scenes. Thus, we cannot factorize the scene structure (as could be done in low-level vision problems) and instead assume a fully-connected scene structure. Fortunately, for scene classification, and particularly for landscape images, the number of critical material regions of interest, n, is generally small ($n \leq 6$) [1], so a brute-force approach to maximizing Equation 3 is tractable.

2.2. Scene Classification System

Figure 2b shows the full scene classification system. After each scene model computes $\sum_{c \in C} P(E|c, S_i)P(c|S_i)$ for scene class S_i, the results are compared at the top level to make a decision (incorporating priors for the scene classes if available).

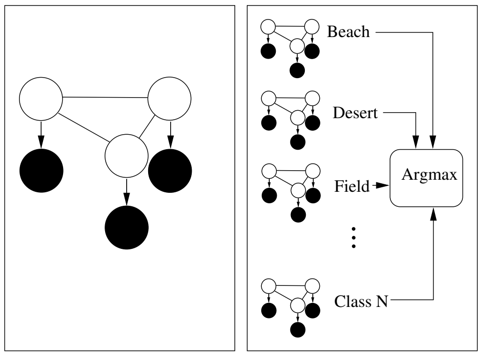


Figure 2. (a) Graphical representation for a single scene class. The observed nodes (detector inputs; shown as filled circles) are each connected to a single node in the fully-connected scene graph (which represents a configuration, treated as a single hidden state in the brute-force approach). (b) Full system: a bank of scene models. The instantiated detector inputs to each scene are the same for each class.

We can learn $P(E|c, S)$ relatively easily. As described above, a reasonable assumption is that a detector's output on a region depends only on the object present in that region and not on other objects or upon the class of the scene. This allows us to factor the distribution into $\prod_{j=1}^{n} P(E_j|c_j)$; each of which describes a single detector's characteristics and can be learned by counting detection frequencies on a training set of regions or fixed using using domain knowledge.

However, $P(c|S)$ is difficult to factor because of the strong dependency between regions. The resulting joint distribution is sparsely populated: there are $O(|M|^n)$ parameters (the counts of each configuration) to learn, and only $|T_S|$ training images of scene class S. The sparseness is exacerbated by correlation between objects and scenes. How do we deal with this sparse distribution?

[1]The material detectors can be imbued with the ability to merge regions, so over-segmentation is rare.

2.3. Naive Approaches to Smoothing

The simplest approach is to do nothing. This adds no ambiguity to the distribution. However, without smoothing, we have $P(c|S) = 0$ for each configuration $C \notin T_S$. This automatically rules out, by giving zero probability to, any valid configuration not seen in the sparse training set, regardless of the evidence: clean, but unsatisfying.

Another simple technique is use a uniform Dirichlet prior on the configurations, implemented by adding a matrix of pseudo-counts of value ϵ to the matrix of configuration counts. However, this can allow for too much ambiguity, because in practice, many configurations should be impossible, for example configurations containing *snow* in the *Desert* model. We seek the middle ground: allowing some matches with configurations not in the training set, but not indiscriminately allowing all matches.

2.4. Graph-based Smoothing

Our goal is to smooth using the training set and knowledge of the image domain. Specifically, we compute $P(c|S)$ as follows. Fix the spatial arrangement of materials. Let $T_S = \{G_{1,S}, G_{2,S}, \ldots G_{|T_S|,S}\}$ be the set of graphs of training images of class S with that spatial arragnment. For $1 \leq j \leq r$, let $N_S^j \in M^r$ be r-dimensional matrices of counts. The configuration, c, is an index into the matrices. Let $sub_j(G)$ denote a subgraph of graph G with j nodes and $\equiv$ denote graph isomorphism.

Then define

$$N_S^j(c) = |\{G_{i,S}\}| : sub_j(c) \equiv sub_j(G_{i,S})) \qquad (4)$$

$$N_S(c) = \sum_{j=1}^{r} \alpha_j N_S^j(c) \qquad (5)$$

$$P(c|S) = \frac{N_S(c)}{\sum_{\tilde{c} \in C} N_S(\tilde{c})} \qquad (6)$$

Each N^j represents the pseudo-counts of the subgraphs of size j occurring in the training set. N^r is the standard count of full scene configurations occurring in the training set. As the subgraph size decreases, the subgraphs are less-specific to the training set, and so should contribute less to the overall distribution. Thus, we expect that the parameters α_j will decrease monotonically. Furthermore, as j decreases, each N_j becomes more densely populated. Intuitively, this is like radial basis function smoothing, in that points "close" to the training points are given more weight in the small area near the peaks than in the larger area at the tails. Finally the counts are normalized to obtain $P(c|S)$. For example, consider the contribution to N of a *single* training configuration "sky above water above sand": each configuration containing "sky above sand", "sky above water", or "sand above water" receives weight α_2 and any configuration containing sky, water, or sand receives weight $\alpha_1 < \alpha_2$; other configurations receive no weight.

The desired result of modifying the uniform Dirichlet prior in this way is that the weight a configuration receives is a function of how closely it matches configurations seen in the training set. While our proposed technique is inspired mainly by the graph matching literature, it can also be viewed as backprojection and as a backoff technique; we discuss each connection in Section 4.

3. Experimental Results

We have a database of 923 images in 5 classes: Beach, Desert, Fall Foliage, Field, and Mountain. Each image is automatically segmented using the algorithm described in (Comaniciu & Meer, 2002), and the semantically-critical regions are manually labeled with their true materials (i.e., ground truth), as in Figure 1b. The ground truth labels correspond to those materials (e.g., sky, grass, foliage, rocks) expected to predict these scenes. Other regions are left unlabeled.

To simulate actual detectors, which are faulty, we randomly perturbed the ground truth to create simulated detector responses. We set the detection rates of individual material detectors on each true material (both *true positive rates*, e.g., how often the grass detector fires on grass regions, and *false positive rates*, e.g., how often the grass detector fires on water regions) by counting performance of corresponding actual detectors on a validation set (or estimating them in the case of detectors to be developed in the future). When they fire, they are assigned a belief that is distributed normally with mean μ. The parameter μ can be set differently for true and false positive detections; varying the ratio between the two is a convenient way to simulate detectors with different accuracies.

Spatial relations are computed using a computationally efficient version of the "weighted walk-through" approach (Berretti et al., 2001), detailed in (Luo et al., 2003). We simplify the relations by ignoring the "far" modifier and the enclosure relations (which occur rarely). We focus further on the single spatial arrangement occurring most often in training: of the 256 images with 3 regions, 172 have a vertical structure, R_1 *above* R_2, R_2 *above* R_3, and R_1 *above* R_3.

We learn $P(c|S)$ using the proposed smoothing method and compare it with learning using two baselines: no smoothing and smoothing by a uniform prior added to the counts.

We perform leave-one-out cross-validation to estimate scene classification performance using the brute force inference method of Equation 3. We eliminate the effect of the priors over scene types by setting them equal. Because we are simulating faulty detectors, we can vary their performance and compare the methods across the spectrum of detector accuracy (Figure 3). While the subgraph smoothing method performs better than the two baselines at all detector accuracies, we admit the difference is small. We believe optimizing the smoothing weights, α_j, should improve performance; learning those parameters is the subject of future work.

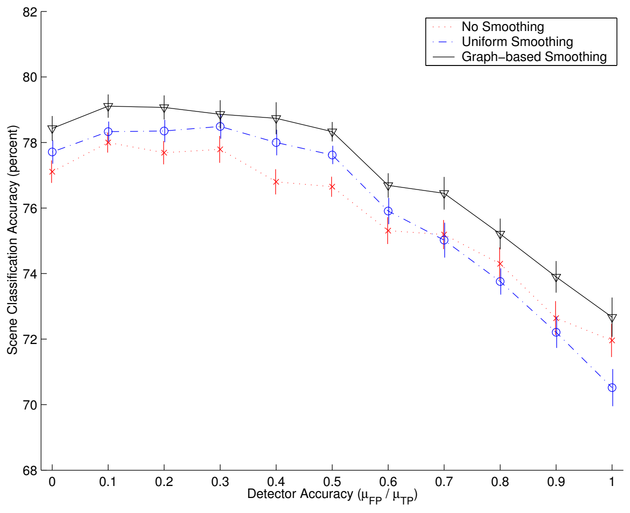


Figure 3. Classification accuracy of the methods as a function of detector accuracy. The smoothing method performs better than the baselines at nearly all detector accuracies. Standard error is shown ($n = 30$).

4. Discussion

As expected, the smoothing technique helps to classify images having a plausible, but rarely-occurring, scene configuration. Figure 4 shows a number of examples. The mountain scene on the left with the configuration "gray sky above snow above grass" was classified correctly by our method, but failed when no smoothing was applied, because that specific configuration did not occur in the training set, but its subgraphs did. Other similar examples are the desert scene in the configuration "blue sky above blue sky above bare ground" and the field scene in the configuration "foliage above blue sky above grass".

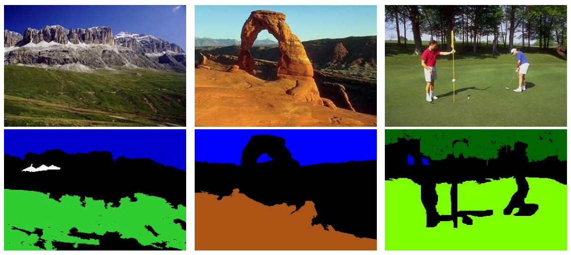


Figure 4. Some images for which the baseline methods fail, but the proposed method succeeds. Top: original scenes. Bottom: hand-labeled regions.

4.1. Related Techniques for Graph Matching

Presently we are doing *exact* graph matching in the sense that we demand an isomorphism for the arcs and nodes, but *inexact* matching in that we are matching *attributed* graphs, those with values or labels attached to the nodes and arcs. We do *multiple-matching*: we are matching into a database of graphs, looking for the best match. Graph isomorphism is a problem of unknown complexity. Inexact graph matching (differing number of nodes) is known to be NP-complete, but is a basic utility for recognition problems. Thus graph matching has a long history in image and scene understanding.

Probabilistic techniques in graph matching, often using relaxation, have been used for some time (Hancock & Kittler, 1990). A comparison of various search strategies appears in (Williams et al., 1999), and (Shams et al., 2001) compares various matching algorithms to one based on maximizing mutual information. Hierarchical relaxation has been used in a Bayesian context (Wilson & Hancock, 1999). Mixture models have been explored for EM matching: a weighted sum of Hamming distances is used as a matching metric in (Finch et al., 1998). Generally, only unary attributes and binary relations are used in these probabilistically-founded searches. More complex relations can be used in relaxation-like schemes as in (Skomorowski, 1999). Various schemes using learning and Bayes nets for inexact matching are explored in (Bengoetxea et al., 2000).

4.2. Related Concepts

One way to view our method is as a backprojection (Swain & Ballard, 1991) technique. If we view the configuration space C as an r-dimensional space, subgraphs of lower dimension (size $i < r$) can be backprojected into C to populate the space. Figure 5 shows an example with $r = 3$ and $i = 2$. The 3D configuration space is sparsely populated in the absence of smoothing. The training points are projected into 2D along 3 axes (the three subgraphs). The resulting 2D spaces corresponding to the same spatial configurations are integrated to combine evidence from different training examples. Finally they are backprojected into the original 3D space (with lower weights than the original counts).

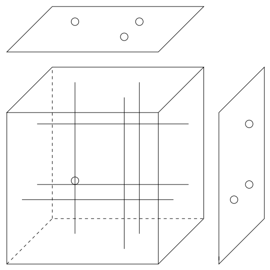


Figure 5. Backprojection using our technique. For legibility in this 3D example, only one training point and two backprojection directions (of the three possible with this spatial configuration) are shown. The training set generalizes through combining counts of subgraphs of multiple training configurations.

Our technique can also be viewed as a backoff technique, as commonly used in speech recognition (Manning & Schutze, 1999); a hierarchical, more principled model is presented in (MacKay & Peto, 1994). If there is insufficient data to learn a trigram model for a given word, one can use a less-specialized, but more densely-populated, bigram or unigram model. However, we pre-compute the probabilities in the learning phase; inference needs no special treatment.

5. Conclusions and Future Work

We have presented a generative model for classifying scenes using faulty material detectors and spatial configurations of materials present in the image. This approach poses a challenge to statistical relational learning, as scene configurations attempt to capture correlations between sets of materials and scene types. Initial results on a small set of landscape images have also shown that performance can be improved by using a smart smoothing technique using subgraphs of the training images.

Clearly, this is work in progress. Future investigation will involve experimentation using real material detectors and a much larger number of images. We also plan to expand the library of spatial arrangements (e.g., R_1 above R_2, R_1 above R_3, R_2 *beside* R_3) and to address the accompanying scalability issues through investigating prototypical spatial arrangements and factorization of the scene models.

More detailed analysis of the behavior of the method may lead to future improvements, such as in learning the parameters of the model. Another interesting direction is to incorporate theoretical work on improving purely uniform priors (Nemenman et al., 2001).

Acknowledgments

This research was supported by a grant from Eastman Kodak Company and by the NSF under grant number EIA-0080124.

References

Bengoetxea, E., Larranaga, P., Bloch, I., Perchanta, A., & Boeres, C. (2000). Inexact graph matching using learning and simulation of Bayesian networks. *Proc. CaNew workshop, ECAI 2000.*

Berretti, S., Bimbo, A. D., & Vicario, E. (2001). Spatial arrangement of color in retrieval by visual similarity. *Pattern Recognition*, *35*, 1661–1674.

Chou, P. (1988). *The theory and practice of Bayesian image labeling.* Doctoral dissertation, University of Rochester, Rochester, NY.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*, 603–619.

Finch, A. W., Wilson, R. C., & Hancock, E. R. (1998). Symbolic graph matching with the EM algorithm. *Pattern Recognition*, *31*, 1777–1790.

Freeman, W., Pasztor, E., & Carmichael, O. (2000). Learning low-level vision. *International Journal of Computer Vision*, *40*, 24–57.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 721–741.

Hancock, E. R., & Kittler, J. (1990). Edge-labeling using dictionary-based relaxation. *IEEE-TPAMI*, *12*, 165–181.

Luo, J., Singhal, A., & Zhu, W. (2003). Towards holistic scene content classification using spatial context-aware scene models. *IEEE Conference on Computer Vision and Pattern Recognition.* Madison, WI.

MacKay, D., & Peto, L. (1994). A hierachical Dirichlet language model. *Natural Language Engineering*, *1*, 1–19.

Manning, C., & Schutze, H. (1999). *Foundations of statistical natural language processing.* Cambridge, MA: MIT Press.

Mulhem, P., Leow, W. K., & Lee, Y. K. (2001). Fuzzy conceptual graphs for matching images of natural scenes. *IJCAI* (pp. 1397–1404).

Nemenman, I., Shafee, F., & Bialek, W. (2001). Entropy and inference, revisited. *NIPS.*

Shams, L., Brady, M., & Schall, S. (2001). Graph matching vs. mutual information maximization for object detection. *Neural Networks*, *14*, 345–354.

Skomorowski, M. (1999). Use of random graph parsing for scene labelling by probabilistic relaxation. *Pattern Recognition Letters*, *60*, 949–956.

Smith, J., Lin, C., Naphade, M., Natsev, A., & Tseng, B. (2003). Multimedia semantic indexing using model vectors. *IEEE ICME.* Baltimore, MD.

Song, Y., & Zhang, A. (2002). Analyzing scenery images by monotonic tree. *ACM Multimedia Systems Journal.*

Swain, M. J., & Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, *7*.

Szummer, M., & Picard, R. W. (1998). Indoor-outdoor image classification. *IEEE International Workshop on Content-based Access of Image and Video Databases.* Bombay, India.

Torralba, A., Murphy, K., Freeman, W., & Rubin, M. (2003). Context-based vision system for place and object recognition. *International Conference on Computer Vision.*

Vailaya, A., Figueiredo, M., Jain, A., & Zhang, H. (1999). Content-based hierarchical classification of vacation images. *Proc. IEEE Multimedia Systems '99 (International Conference on Multimedia Computing and Systems).* Florence, Italy.

Williams, M. L., Wilson, R. C., & Hancock, E. R. (1999). Deterministic search for relational graph matching. *Pattern Recognition*, *32*, 1255–1271.

Wilson, R. C., & Hancock, E. R. (1999). Graph matching with hierarchical discrete relaxation. *Pattern Recognition Letters*, *20*, 80–96.

Relational Markov Networks for Collective Information Extraction

Razvan C. Bunescu RAZVAN@CS.UTEXAS.EDU
Raymond J. Mooney MOONEY@CS.UTEXAS.EDU
Department of Computer Sciences, The University of Texas at Austin, 1 University Station C0500, Austin, TX 78712-0233

Abstract

Most information extraction (IE) systems treat separate potential extractions as independent. However, in many cases, considering influences *between* different potential extractions could improve overall accuracy. Statistical methods based on *undirected* graphical models, such as *conditional random fields* (CRFs), have been shown to be an effective approach to learning accurate IE systems. We present a new IE method that employs Relational Markov Networks, which can represent arbitrary dependencies between extractions. This allows for "collective information extraction" that exploits the mutual influence between possible extractions. Experiments on learning to extract protein names from biomedical text demonstrate the advantages of this approach.

1. Introduction

Information extraction (IE), locating references to specific types of items in natural-language documents, is an important task with many practical applications. Since IE systems are difficult and time-consuming to construct, most recent research has focused on empirical techniques that automatically construct information extractors by training on supervised corpora (Cardie, 1997; Califf, 1999). One of the current best empirical approaches to IE is *conditional random fields* (CRF's) (Lafferty et al., 2001). CRF's are a restricted class of *undirected graphical models* (Jordan, 1999) designed for sequence segmentation tasks such as IE, part-of-speech (POS) tagging (Lafferty et al., 2001), and shallow parsing (Sha & Pereira, 2003). In a recent follow-up to previously published experiments comparing a large variety of IE-learning methods (including HMM, SVM, MaxEnt, and rule-based methods) on the task of tagging references to human proteins in Medline abstracts (Bunescu et al., 2004), CRF's were found to significantly out-perform competing techniques.

As typically applied, CRF's, like almost all IE methods, assume separate extractions are independent and treat each potential extraction in isolation. However, in many cases, considering influences *between* extractions can be very useful. For example, in our protein-tagging task, repeated references to the same protein are common. If the context surrounding one occurrence of a phrase is very indicative of it being a protein, then this should also influence the tagging of another occurrence of the same phrase in a different context which is not indicative of protein references.

Relational Markov Networks (RMN's) (Taskar et al., 2002) can be seen as a generalization of CRF's that allow for *collective classification* of a set of arbitrarily related entities by integrating information from features of individual entities as well as the relations between them. Results on classifying connected sets of web pages have verified the advantage of this approach (Taskar et al., 2002). In this paper, we present an approach to *collective information extraction* using RMN's that simultaneously extracts all of the information from a document by exploiting the textual content and context of each relevant substring as well as the document relationships between them. Experiments on human protein tagging demonstrate the advantages of collective extraction on several annotated corpora of Medline abstracts.

2. The RMN Framework for Entity Recognition

Assume we are given a collection of training documents D where all named entities have been manually annotated. We associate with each document $d \in D$ a set of candidate entities $d.E$, in our case a restricted set of token sequences from the document. Each entity $e \in d.E$ is characterized by a set of boolean features $e.F$. This set of features is the same for all candidate entities, and it can be assimilated with the relational database definition of a table. One particular feature is $e.label$ which is set to 1 if e is considered a valid extraction, and 0 otherwise. In our document model, labels are the only hidden features, and the inference procedure will try to find a most probable assignment of values to labels, given the current model parameters.

Each document is associated with an undirected graphical model, with nodes corresponding directly to entity features, one node for each feature of each candidate entity in the document. The set of edges is created by matching *clique templates* against the entire set of entities $d.E$. A clique template is a procedure that finds all subsets of entities satisfying a given constraint and then connects feature nodes associated with the entities in each subset so that they form a clique.

Formally, there is a set of clique templates C, with each template $c \in C$ defined by:

1. A matching operator M_c for selecting subsets of entities.

2. A selected set of features $S_c = \langle X_c, Y_c \rangle$ for entities returned by the matching operator. X_c denotes the observed features, while Y_c refers to the hidden labels.

3. A clique potential ϕ_c that gives the compatibility of each possible configuration of values for the features in S_c, s.t. $\phi_c(s) \geq 0, \forall s \in S_c$.

Given a set, E, of nodes, $M_c(E) \subseteq 2^E$ consists of subsets of entities whose selected feature nodes S_c are to be connected in a clique. In previous applications of RMNs, the selected subsets of entities for a given template have the same size; however, our clique templates may match a variable number of entities. The set S_c may contain the same feature from different entities. Usually, for each entity in the matching set, its label is included in S_c. Depending on the number of hidden labels in Y_c, we define two categories of clique templates:

- **Local Templates** are all templates $c \in C$ for which $|Y_c| = 1$. They model the correlations between an entity's observed features and its label.

- **Global Templates** are all templates $c \in C$ for which $|Y_c| > 1$. They capture influences between multiple entities from the same document.

After the graph model for a document d has been completed with cliques from all templates, the probability distribution over the random field of hidden entity labels $d.Y$ given the observed features $d.X$ is computed as:

$$P(d.Y|d.X) = \frac{1}{Z(d.X)} \prod_{c \in C} \prod_{G \in M_c(d.E)} \phi_C(G.X_c, G.Y_c) \quad (1)$$

where $Z(d.X)$ is the normalizing partition function:

$$Z(d.X) = \sum_{Y} \prod_{c \in C} \prod_{G \in M_c(d.E)} \phi_C(G.X_c, G.Y_c) \quad (2)$$

3. Candidate Entities and Entity Features

Like most entity names, almost all protein mentions in our data are base noun phrases or parts of them. Therefore, such substrings are used to determine candidate entities. To avoid missing options, we adopt a very broad definition of base noun phrase.

Definition 1: A *base noun phrase* is a maximal contiguous sequence of tokens whose POS tags are from {*"JJ", "VBN", "VBG", "POS", "NN", "NNS", "NNP", "NNPS", "CD", "−"*}, and whose last word (the head) is tagged either as a noun, or a number.

Candidate extractions consist of base NPs, augmented with all their contiguous subsequences headed by a noun or number.

The set of features associated with each candidate is based on the feature templates introduced in (Collins, 2002), used there for training a ranking algorithm on the extractions returned by a maximum-entropy tagger. Many of these features use the concept of *word type*, which allows a different form of token generalization than POS tags. The word type is created by replacing any maximal contiguous sequences of capital letters with 'A', of lower-case letters with 'a', and of digits with '0'. For example, the word *TGF-1* would be mapped to type *A-0*. Consequently, each token position i in a candidate extraction provides three types of information: the word itself, its POS tag, and its type.

The left and right boundaries of a candidate extraction generate bigram and trigram features that combine words and word types. Other useful features are the head (last word), prefix and suffix lists of words from the candidate entity. A more detailed account of these feature templates is given in (Collins, 2002).

4. Local Clique Templates

Each feature template instantiates numerous features. For example, the candidate extraction `HDAC1 enzyme` has the head word *HD=enzyme*, the short type *ST=A0_a*, the prefixes *PF=A0* and *PF=A0_a*, and the suffixes *SF=a* and *SF=A0_a*. All other features depend on the left or right context of the entity. Feature values that occur less than three times are filtered out. If, after filtering, we are left with h distinct boolean features (f_i=v_j), we create h local (clique) templates $LT_1, LT_2, ..., LT_h$. Each template's matching operator is set to match any single-entity set. The collection of features S_i corresponding to template LT_i applied to the singleton entity set $\{e\}$ is $S_i = \langle X_i,\ Y_i \rangle = \langle \{ e.f_i{=}v_j \},\ \{e.label\} \rangle$. The 2-node cliques created by all h templates around one entity are illustrated in Figure 1.

Each entity has a label node connected to its own set of h binary feature nodes. This leads to an excessive num-

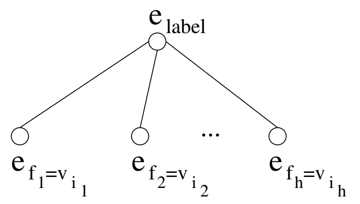


Figure 1. RMN generated by local templates.

ber of nodes in the model, most of which have the value zero. To reduce the number of nodes, we could remove the observed nodes from the graph, which then results in many one-node clique potentials (due to the observed features) being associated with the same label node. Because these clique potentials may no longer be distinguished in the MRF graph, in order to have all of them as explicit as possible in the graphical model, we decided to transform the relational Markov network into its equivalent *factor graph* representation. Factor graphs (Kschischang et al., 2001) are bipartite graphs that express how a global function of many variables (the probability $P(d.Y|d.X)$ in Equation 1) factors into a product of local functions (the potentials $\phi_C(G.X_c, G.Y_c)$ in Equation 1). Factor graphs subsume many different types of graphical models, including Bayesian networks and Markov random fields. The sum/max-product algorithms used for inference in factor graphs generalize a wide variety of algorithms including the forward/backward algorithm, the Viterbi algorithm, and Pearl's belief propagation algorithm (Pearl, 1988). To obtain the factor graph for a given Markov random field, we copy all nodes from the MRF, and create a new node for each instantiated clique potential. Each potential node is then linked to all nodes from the associated clique. However in this case, instead of creating a potential node for each feature-value pair as in the initial MRF model, we create a potential node only for the binary features that are 1 for the given entity. Correspondingly, the table associated with the potential will be reduced from 4 to 2 values. As an example, Figure 2 shows that part of the factor graph which is generated around the entity label for `HDAC1 enzyme`.

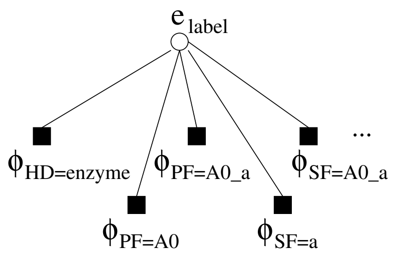


Figure 2. Factor Graph for local templates.

Note that the factor graph above has an equivalent RMN graph consisting of a one-node clique only, on which it is hard to visualize the various potentials involved. There are cases where different factor graphs may yield the same underlying RMN graph, which makes the factor graph representation preferable.

5. Global Clique Templates

Global clique templates enable us to model hypothesized influences between entities from the same document. They connect the label nodes of two or more entities, which, in the factor graph, translates into potential nodes connected to at least two label nodes. In our experiments we use three global templates:

Overlap Template (OT): No two protein names overlap in the text i.e if the span of one protein is $[s_1, e_1]$ and the span of another protein is $[s_2, e_2]$, and $s_1 \leq s_2$, then $e_1 < s_2$.

Repeat Template (RT): If multiple entities in the same document are repetitions of the same name, their labels tend to have the same value (i.e. most of them are protein names, or most of them are not protein names). Later we discuss situations in which repetitions of the same protein name are not tagged as proteins, and design an approach to handle this.

Acronym Template (AT): It is common convention that a protein is first introduced by its long name, immediately followed by its short-form (acronym) in parentheses.

5.1. The Overlap Template

The overlap template matches any two overlapping candidate entities and connects their label nodes through a potential node that forbids the case in which both have label-value 1, as illustrated in Table 1.

An alternative solution for the overlap template is to create a potential node for each token position that is covered by at least two candidate entities in the document, and connect it to their label nodes. The difference in this case is that the potential node will be connected to a variable number of entity label nodes. However this approach is better since it leads to fewer nodes being created in the document factor graph, which results in faster inference.

Table 1. Overlap Potential.

ϕ_{OT}	$e_1.label = 0$	$e_1.label = 1$
$e_2.label = 0$	1	1
$e_2.label = 1$	1	0

5.2. The Repeat Template

We could specify the potential for the repeat template in a similar 2-by-2 table, this time leaving the table entries to be learned, given that it is not a hard constraint. However we can do better by noting that the vast majority of cases where a repeated protein name is not also tagged as a protein happens when it is part of a larger phrase that *is* tagged. For example, `HDAC1 enzyme` is a protein name, therefore `HDAC1` is not tagged in this phrase, even though it was tagged previously in the abstract where it was not followed by `enzyme`. We need a potential that allows two entities with the same text to have different labels if the entity with label-value 0 is inside another entity with label-value 1. But a candidate entity may be inside more than one "including" entity, and the number of including entities may vary from one candidate extraction to another. We solve this problem, by introducing a logical OR clique template that matches a variable number of entities. When this template matches a subset of entities $e_1, e_2, ..., e_n$, it will create an auxiliary OR entity e_{or}, with a single feature $e_{or}.label$. The potential function is set so that it assigns a non-zero potential only when $e_{or}.label = e_1.label \vee e_2.label \vee ... \vee e_n.label$. The cliques are only created as needed, e.g. when the auxiliary OR variable is required by repeat and acronym clique templates.

Figure 3 shows the factor graph for a sample instantiation of the repeat template using the OR template. Here, u and v represent two same-text entities, u_1, u_2, ... , u_n are all entities that include u, and $v_1, v_2, ..., v_m$ are entities that include v. To avoid clutter, all entities in this and subsequent factor graphs stand for their corresponding label features. The potential function can either be preset to prohibit unlikely label configurations, or it can be learned to represent an appropriate soft constraint. In our experiments, it was learned since this gave slightly better performance.

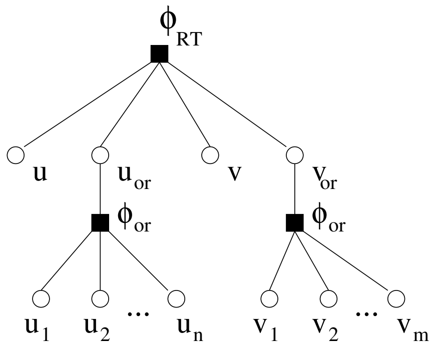


Figure 3. Repeat Factor Graph.

5.3. The Acronym Template

One approach to the acronym template would be to use an extant algorithm for identifying acronyms and their long forms in a document, and then define a potential function that would favor label configurations in which both the acronym and its definition have the same label. One such algorithm is described in (Schwartz & Hearst, 2003), achieving a precision of 96% at a recall rate of 82%. However, because this algorithm would miss a significant number of acronyms, we have decided to implement a softer version as follows: detect all situations in which a single word is enclosed between parentheses, such that the word length is at least 2 and it begins with a letter. Let v denote the corresponding entity. Let u_1, u_2, ..., u_n be all entities that end exactly before the open parenthesis. If this is a situation in which v is an acronym, then one of the entities u_i is its corresponding long form. Consequently, we use a logical OR template to introduce the auxiliary variable u_{or}, and connect it to v's node label through an acronym potential, as illustrated in Figure 4. For example, consider the phrase `the antioxidant superoxide dismutase - 1 ( SOD1 )`, where both `superoxide dismutase - 1` and `SOD1` are tagged as proteins. `SOD1` satisfies our criteria for acronyms, thus it will be associated with the entity v in Figure 4. The candidate long forms are u_1 = `antioxidant superoxide dismutase - 1`, u_2 = `superoxide dismutase - 1`, and u_3 = `dismutase - 1`.

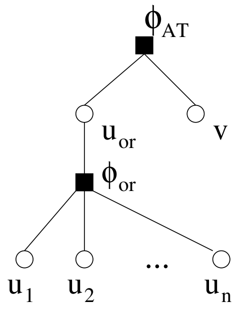


Figure 4. Acronym Factor Graph.

6. Inference in Factor Graphs

Given the clique potentials, the inference step for the factor graph associated with a document involves computing the most probable assignment of values to the hidden labels of all candidate entities:

$$Y^* = arg \max_Y P(d.Y|d.X) \qquad (3)$$

where $P(d.Y|d.X)$ is defined as in Equation 1. A brute-force approach is excluded, since the number of possible label configurations is exponential in the number of candidate entities. The sum-product algorithm (Kschischang et al., 2001) is a message-passing algorithm that can be used for computing the marginal distribution over the label variables in factor graphs without cycles, and with a minor change (replacing the sum operator used for marginalization with a max operator) it can also be used for deriving the most probable label assignment. In our case, in order to get an acyclic graph, we would have to use local templates only. However, it has been observed that the algorithm often converges in general factor graphs, and when it converges, it gives a good approximation to the correct marginals. The algorithm works by altering the belief at each label node by repeatedly passing messages between the node and all potential nodes connected to it (Kschischang et al., 2001).

The time complexity of computing messages from a potential node to a label node is exponential in the number of label nodes attached to the potential. Since this "fan-in" can be large for OR potential nodes (and also for the second solution to overlap potential nodes), this step required optimization. Fortunately, due to the special form of the OR and overlap potentials, and the normalization before each message-passing step, we were able to develop a linear-time algorithm for these special cases. For example, the formulae for computing the OR messages for the sum-product algorithm are shown in Equations 4, with the relevant messages illustrated in Figure 5 (to avoid clutter, e and ϕ stand for e_{OR} and ϕ_{OR} respectively).

$$\begin{aligned}
\mu_{\phi \to e}(0) &= \prod_{i=1}^{n} \mu_{e_i \to \phi}(0) \\
\mu_{\phi \to e}(1) &= 1 - \mu_{\phi \to e}(0) \\
\mu_{\phi \to e_i}(0) &= \mu_{\phi \to e}(1) + \\
&\quad \frac{\mu_{\phi \to e}(0)}{\mu_{e_i \to \phi}(0)} \left(\mu_{\phi \to e}(0) - \mu_{\phi \to e}(1)\right) \\
\mu_{\phi \to e_i}(1) &= \mu_{\phi \to e}(1)
\end{aligned} \tag{4}$$

7. Learning Potentials in Factor Graphs

Following a maximum likelihood estimation, we shall use the log-linear representation of potentials:

$$\phi_C(G.X_c, G.Y_c) = exp\{\mathbf{w_c f_c}(G.X_c, G.Y_c)\}$$

Let $\mathbf{w}$ be the concatenated vector of all potential parameters $\mathbf{w_c}$. One approach to finding the maximum-likelihood solution for $\mathbf{w}$ is to use a gradient-based method, which requires computing the gradient of the log-likelihood with respect to potential parameters $\mathbf{w_c}$. It can be shown that this gradient is equal with the difference between the empirical counts of $\mathbf{f_c}$ and their expectation under the current

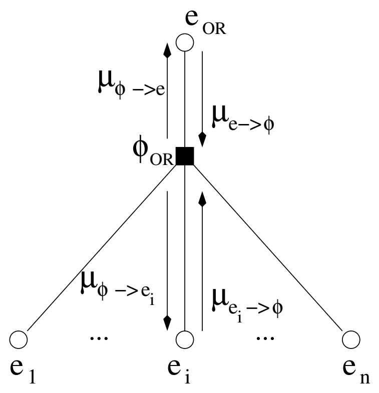


Figure 5. Messages in OR Factor Graph.

set of parameters $\mathbf{w}$. This expectation is expensive to compute, since it requires summing over all possible configurations of candidate entity labels from a given document. To circumvent this complexity, we use the Collins' voted perceptron approach (Collins, 2002), which approximates the full expectation of $\mathbf{f_c}$ with the $\mathbf{f_c}$ counts for the most likely labeling under the current parameters, $\mathbf{w}$. In all our experiments, the perceptron was run for 50 epochs, with a learning rate set at 0.01.

8. Experimental Results

We have tested the RMN approach on two datasets that have been hand-tagged for human protein names. The first dataset is Yapex[1] which consists of 200 Medline abstracts. The second dataset is Aimed[2] which has been previously used for training the protein interaction extraction systems in (Bunescu et al., 2004). It contains 225 Medline abstracts, of which 200 are known to describe interactions between human proteins, while the other 25 do not refer to any interaction. We compared the performance of three systems: **LT-RMN** is the RMN approach using local templates and the overlap template, **GLT-RMN** is the full RMN approach, using both local and global templates, and **CRF**, which uses a CRF for labeling token sequences. We used the CRF implementation from (McCallum, 2002) with the set of tags and features used by the Maximum-Entropy tagger described in (Bunescu et al., 2004). All Medline abstracts were tokenized and then POS tagged using Brill's tagger (Brill, 1995). Each extracted protein name in the test data was compared to the human-tagged data, with the positions taken into account. Two extractions are considered a match if they consist of the same character sequence in the same position in the text. Results are shown in Tables 2 and 3 which give average precision, recall, and F-measure

[1]URL: www.sics.se/humle/projects/prothalt/

[2]URL: ftp.cs.utexas.edu/mooney/bio-data/

using 10-fold cross validation.

Table 2. Extraction Performance on Yapex.

Method	Precision	Recall	F-measure
LT-RMN	70.79	53.81	61.14
GLT-RMN	69.71	65.76	67.68
CRF	72.45	58.64	64.81

Table 3. Extraction Performance on Aimed.

Method	Precision	Recall	F-measure
LT-RMN	81.33	72.79	76.82
GLT-RMN	82.79	80.04	81.39
CRF	85.37	75.90	80.36

These tables show that, in terms of F-measure, the use of global templates for modeling influences between possible entities from the same document significantly improves extraction performance over the local approach (a one-tailed t-test for statistical significance results in a p value less than 0.01 on both datasets). There is also a small improvement over CRF's, with the results being statistically significant only for the Yapex dataset, corresponding to a p value of 0.02. We hypothesize that further improvements to the LT-RMN approach would push the GLT-RMN performance even higher. The tagging scheme used by CRFs, in which each token is assigned a tag, is essentially different from the RMN approach, where candidate extractions are either rejected or accepted. In the tagging approach used by CRFs, extracted entities are available only after tagging is complete, thereby making it difficult to account for influences between them during tagging.

9. Related Work

There have been some previous attempts to use global information from repetitions, acronyms, and abreviations during extraction. In (Chieu & Ng, 2003), a set of global features are used to improve a Maximum-Entropy tagger; however, these features do not fully capture the mutual influence between the labels of acronyms and their long forms, or between entity repetitions. In particular, they only allow earlier extractions in a document to influence later ones and not vice-versa. The RMN approach handles these and potentially other mutual influences between entities in a more complete, probabilistically sound manner.

10. Conclusions and Future Work

We have presented an approach to collective information extraction that uses Relational Markov Networks to reason about the mutual influences between multiple extractions. A new type of clique template – the logical OR template – was introduced, allowing a variable number of relevant entities to be used by other clique templates. Soft correlations between repetitions and acronyms and their long form in the same document have been captured by global clique templates, allowing for local extraction decisions to propagate and mutually influence each other. Experimental results showed that a collective approach to extraction significantly improves performance.

Regarding future work, a richer set of features for the local templates would likely improve performance. Currently, LT-RMN's accuracy is still significantly less than CRF's, which limits the performance of the full system. Another limitation is the approximate inference used by both RMN methods. The number of factor graphs for which the sum-product algorithm did not converge was non-negligible, and our approach stopped after a fix number of iterations. Besides exploring improvements to loopy belief propagation that increase computational cost (Yedidia et al., 2000), we intend to examine alternative approximate-inference methods such as Gibbs sampling, and other Monte Carlo algorithms.

A natural next step is to integrate IE subtasks like named entity recognition and coreference resolution, such that decisions made in one subtask influence decisions made in the other. The context of a pronoun referring to an entity can help in disambiguating the class of that entity through the use of a general repeat template. Recent work in anaphora resolution using RMNs (McCallum & Wellner, 2003) and the joint solving of two different NLP tasks using dynamic CRFs (McCallum et al., 2003) show the benefit of an integrated, collective approach.

11. Acknowledgements

This work was partially supported by grants IIS-0117308 and IIS-0325116 from the NSF.

References

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, *21*, 543–565.

Bunescu, R., Ge, R., Kate, R. J., Marcotte, E. M., Mooney, R. J., Ramani, A. K., & Wong, Y. W. (2004). Comparative experiments on learning information extractors for proteins and their interactions. *Special Issue in the Journal Artificial Intelligence in Medicine on Summarization and Information Extraction from Medical Documents*. To appear.

Califf, M. E. (Ed.). (1999). *Papers from the AAAI-1999*

Workshop on Machine Learning for Information Extraction. Orlando, FL: AAAI Press.

Cardie, C. (1997). Empirical methods in information extraction. *AI Magazine*, *18*, 65–79.

Chieu, H. L., & Ng, H. T. (2003). Named entity recognition with a maximum entropy approach. *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-2003)* (pp. 160–163). Edmonton, Canada.

Collins, M. (2002). Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-02)* (pp. 489–496). Philadelphia, PA.

Jordan, M. I. (Ed.). (1999). *Learning in graphical models*. Cambridge, MA: MIT Press.

Kschischang, F. R., Frey, B., & Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*, 498–519.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of 18th International Conference on Machine Learning (ICML-2001)* (pp. 282–289). Williams College, MA.

McCallum, A., Rohanimanesh, K., & Sutton, C. (2003). Dynamic conditional random fields for jointly labeling multiple sequences. *NIPS-2003 Workshop on Syntax, Semantics, Statistics*. Whistler, Canada.

McCallum, A., & Wellner, B. (2003). Toward conditional models of identity uncertainty with application to proper noun coreference. *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web* (pp. 79–86). Acapulco, Mexico.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Mateo,CA: Morgan Kaufmann.

Schwartz, A. S., & Hearst, M. A. (2003). A simple algorithm for identifying abbreviation definitions in biomedical text. *Proceedings of the 8th Pacific Symposium on Biocomputing* (pp. 451–462). Lihue, HI.

Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. *Proceedings of Human Language Technology and the Meeting of the North American Association for Computational Linguistics* (pp. 134–141). Edmonton, Canada.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. *Proceedings of 18th Conference on Uncertainty in Artificial Intelligence (UAI-02)* (pp. 485–492). Edmonton, Canada.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2000). Generalized belief propagation. *Advances in Neural Information Processing Systems 12* (pp. 689–695). Denver, CO.

Clustering in Relational Biological Data

Aynur Dayanik AYNUR@CS.RUTGERS.EDU
Computer Science Department, 110 Frelinghuysen Road, Piscataway, NJ 08854 USA

Craig G. Nevill-Manning CRAIGNM@GOOGLE.COM
Google, Inc., 1440 Broadway, New York, NY 10018 USA

Abstract

The scientific endeavor of biology is becoming increasingly reliant on data in electronic form, and it is therefore necessary for biologists to manage and understand large quantities of data. Publicly available data including biological sequences, biological structures, and literature in the life sciences have grown to such an extent that computing is essential simply to store and access it. Here we describe a clustering approach by exploiting the relational structure of biological data to help with the next step: to enhance understanding of the data by combining techniques from information retrieval with those from bioinformatics. By computing over a network of sequence-structure-literature relationships it is possible to infer clusters of related articles, sequences and structures. This paper describes the general framework and its application to several biological domains.

1. Introduction

The growth of bioinformatics has coincided with the growth of the worldwide web. This happy coincidence, in conjunction with savvy policy on the part of publishers and the National Library of Medicine, has resulted in a body of data that is singularly well connected. For example, whenever a paper is published in the biological literature, any biological sequences or structures that were determined or analyzed in the course of the research must be submitted to the appropriate databases. The corresponding abstract in MEDLINE is then annotated with the ID of the sequence or structure. This linking allows researchers to find experimental data very easily once they have identified a paper of interest, or conversely to find an analysis of a particular sequence or structure. The National Center for Biotechnology Information (NCBI), part of the National Library of Medicine, provides online access to MEDLINE abstracts, GenBank sequences, and many other data types, through their Entrez system. The ability to browse data and literature seamlessly is important, but the underlying data has much greater potential.

Clustering is the task of grouping a set of objects into different subsets such that objects belonging to the same cluster are highly similar to each other. Convential clustering algorithms employ distance (or similarity) measure to form the clusters (Kirsten et al., 2000). On the other hand, graph partitioning algorithms exploit the structure of a graph to find highly connected objects. Rich relational structure of biological data can be represented as a graph for clustering biological data. Clustering biological data would be useful not only for exploring the data but also for discovering implicit links between the objects.

Here we describe a technique for clustering of biological objects: sequences, structures and literature. We use METIS, a multilevel graph partitioning system, to form the clusters. This process identifies subsets of nodes that are highly connected to each other, but are less strongly connected to the rest of the graph. These clusters are formed based on the pairwise relationships among biological data, so we can evaluate their topical cohesiveness by examining independent metadata such as Gene Ontology (GO) annotations and terms in MEDLINE abstracts. We also evaluate the clusters by hand for relevance, and find that the clusters are highly topical.

The organization of the paper is as follows. The next section describes the databases we used. In section 3, we describe the construction of a graph from the databases, and then present our graph partitioning approach in section 4. Section 5 describes the BioIR system we built. In section 6, we present and discuss the empirical results to assess the quality of clusters. Finally, we end with a summary.

2. Data Sources

In this section, we will briefly describe the data sources we used to construct our graph.

MEDLINE**:** MEDLINE is a digital collection of life science literature consisting of over twelve million abstracts. MEDLINE articles contain links to the sequences and structures that the article discuss. The MEDLINE collection we used contained about 100,000 abstracts.

SWISS-PROT**:** The Swiss Protein Database (SWISS-PROT) is a curated protein sequence database (Bairoch & Apweiler, 2000). The database contains high-quality annotation including descriptions of each protein's function. SWISS-PROT entries are cross-referenced to several other databases, including MEDLINE, PROSITE and the PDB. SWISS-PROT has about 120,000 protein sequences.

PDB: The Protein Data Bank (PDB) contains 3–D structural data of biological macromolecules (proteins and nucleic acids) (Berman et al., 2000). The PDB entries are also cross-referenced to the primary citations in MEDLINE and other databases including ENZYME and SWISS-PROT. PDB has about 20,000 structures.

3. Constructing the Graph

Using the relationships between biological data objects, we construct a weighted undirected graph where nodes correspond to entries from the databases listed in Section 2, including MEDLINE abstracts, protein sequences from SWISS-PROT, structures from PDB. Table 1 shows excerpts from a MEDLINE record that contains references to three structures in PDB, along with the title and abstract of the paper.

Edges in the graph correspond to explicit links between entries encoded in the databases, such as the sequence annotations in MEDLINE abstracts, and pairwise similarity relationships between same type of objects. We use BLAST (Altschul et al., 1997), a sequence alignment technique, to compute similarities between protein sequences. We employ MG[1] (Witten et al., 1999), a full-text retrieval engine, to compute similarities between MEDLINE abstracts. We use the SCOP (Murzin et al., 1995), a database of hierarchical classification of PDB entries based on structural similarities, to relate PDB entries to each other. We assume a relationship between two PDB entries if they are in the same leaf of the SCOP hierarchy.

Figure 1 shows an example graph of biological entities, including edges between abstracts and sequences, abstracts and structures, and between sequences and structures as well as between same type of objects by similarity relationships.

We assign weights to edges as follows. We assign a weight of 100 to explicit edges encoded by the databases (as for 100% relatedness). We normalize the similarity scores between same type of objects computed by MG and BLAST to the range [1,100]. We assign a weight of 100 to PDB-PDB relationships obtained using SCOP since SCOP classifications are done by biologists.

[1]Available at http://www.cs.mu.oz.au/mg/.

PMID- 11807546
TI - Structural basis for the activation of anthrax adenylyl cyclase exotoxin by calmodulin.
AB - Oedema factor, a calmodulin-activated adenylyl cyclase, is important in the pathogenesis of anthrax. Here we report the X-ray structures of oedema ...
SI - **PDB**/1K8T
SI - **PDB**/1K90
SI - **PDB**/1K93
...

Table 1. A sample MEDLINE file linking to PDB entries

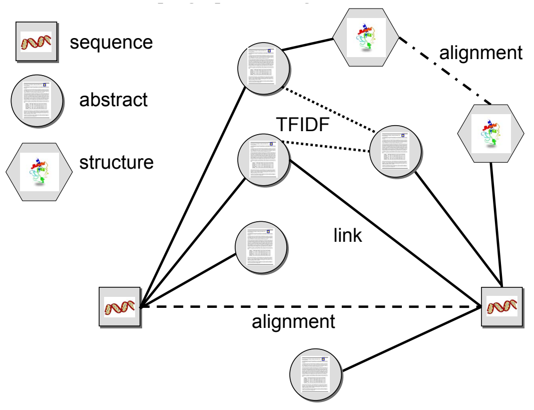


Figure 1. An example graph of sequences, abstracts, and structures related by explicit references and similarity relationships.

4. Graph Partitioning

The objective of graph partitioning is to partition the graph into k roughly equal parts such that the sum of the weights connecting different parts is minimized, thereby each part is highly similar. The graph partitioning problem is NP-complete. However, many heuristics have been developed that find a reasonably good partition.

Traditional graph partitioning algorithms compute a partition of a graph by operating directly on the graph, and they are usually slow. On the other hand, multilevel graph partitioning algorithms reduce the size of the graph by collapsing vertices and edges, partition the smaller graph, and then coarsen it to construct a partition for the original graph. These algorithms are generally fast and produce high-quality partitions. We chose the *pmetis* program provided by the METIS[2] software, a publicly available graph partitioning software package. The partitioning algorithm used by *pmetis* is based on multilevel recursive bisection described in (Karypis & Kumar, 1998).

5. The BioIR System

We built a system, called BioIR, to test our approach. We stored all the entities in the databases described in Section 2, and the relationships between them in a MySQL database. Then we created a graph using these tables as explained in Section 3, and stored the nodes and the edges between them in the graph back in MySQL.

The graph is not completely connected: there are many disconnected subgraphs. There is one large connected component as well as 864 connected components of size at most 100. We partitioned the largest connected component to obtain about 1000 clusters of 200 nodes each. We chose 1000 as the number of clusters since the PROSITE, database of sequence motifs/patterns, has about 1000 entries. Therefore, we can use PROSITE for quantitative evaluation of the clustering. Also, browsing clusters of size 200 would be manageable by biologists. We kept all other small size connected components as clusters themselves and stored all the resulting clusters in a MySQL database.

5.1. Identifying Descriptive Terms From Abstracts

We aim to identify words that best describe the set of documents in clusters by analyzing the MEDLINE articles of the clusters. These descriptive words can be used as index terms to identify the contents of the clusters. We identified the descriptive words as follows. We considered the words in the title and abstract of all articles in a cluster after eliminating stop words. We removed all punctuation, and converted all uppercase letters to lowercase. Then we ranked the resulting words by calculating p-values considering the entire set of MEDLINE articles in our collection. p-value calculation was described in subsection 6.4. We kept the top twenty most significant words, the ones having the smallest p-values, in our database for each cluster. We use the resulting set of twenty words to index the clusters, and build a search utility against this index using MySQL.

6. Experimental Results and Discussion

First, to quantify the quality of the produced clustering, we computed the entropy and purity of the clustering for SWISS-PROT and PDB entries by taking PROSITE and SCOP classifications as reference classifications. Note that we did not use PROSITE at all to obtain the clustering. However, we used SCOP to relate PDB entries. We are interested in seeing how well we recover the relational structure of PDB entries.

[2]Available at http://www-users.cs.umn.edu/~karypis/metis/.

Second, we evaluated our system on several biological domains, described in subsection 6.2 by carrying out a user study to understand the quality of the clusters. Also, to quantify the quality of the sample clusters analyzed by a domain expert, we analyzed the SWISS-PROT to GO mappings and MEDLINE abstracts in clusters to extract common words to see their relevance to the topics of interests.

6.1. Evaluation of Overall Clustering quality

In general, two different metrics are used to measure the quality of a clustering. The first metric is the widely used *entropy* measure that considers how the various classes of objects are distributed within each cluster, and the second measure is the *purity* measure that considers the extend to which each cluster contained objects from primarily one class.

Let C_r denote a particular cluster of size n_r. The entropy of this cluster is defined as

$$E(C_r) = -\frac{1}{logq}\sum_{i=1}^{q}\frac{n_r^i}{n_r}\, log\frac{n_r^i}{n_r}, \tag{1}$$

where q is the number of classes in the dataset, and n_r^i is the number of objects of the ith class that were assigned to the rth class. The overall entropy of the clustering, where k is the number of clusters, is then defined as the sum of the individual cluster entropies weighted according to the cluster size:

$$Entropy = \sum_{r=1}^{k}\frac{n_r}{n}E(C_r). \tag{2}$$

A perfect clustering will be the one consisting of clusters that contain objects from only a single class. In this case, the entropy will be zero. In general, the smaller the entropy values, the better the clustering solution is.

The purity of this cluster is defined as

$$P(C_r) = \frac{1}{n_r}max_i(n_r^i). \tag{3}$$

The overall purity of the clustering is defined as a weighted sum of the individual cluster purities and is computed as

$$Purity = \sum_{r=1}^{k}\frac{n_r}{n}P(C_r). \tag{4}$$

In general, the larger the values of purity, the better the clustering solution is.

Table 2 shows the entropy and purity values computed for SWISS-PROT and PDB entries using PROSITE and SCOP classifications as reference classifications, respectively. Recall that the closer the entropy value to 0, the better the clustering is. Also, the closer the purity value to 1, the better the clustering is.

As a baseline, we created clusters by randomyly assigning the objects in our graph to 1000 clusters and computed the entropy and purity measures for SWISS-PROT, PDB entity types. Table 2 also shows the average results of 10 random partitioning experiments. The average entropy value for 10 random partitionings is much higher than those of our graph partitioning, and the average purity value for 10 random partitionings is much lower than those of our graph partitioning. These suggest that we discover meaningful groupings by our graph partitioning method.

	SwissProt		**PDB**	
Method	Entropy	Purity	Entropy	Purity
METIS	0.1180	0.4129	0.1145	0.7334
Random	0.5596	0.0246	0.4358	0.0873

Table 2. Entropy and purity values for SWISS-PROT and PDB clusterings using PROSITE and SCOP classifications as references, respectively.

6.2. Biological Domains

The following biological domains were carefully examined by our domain expert, a Ph.D. candidate in Molecular Biology. We give a brief description of each domain below.

Calmodulin: Calmodulin is a ubiquitous intracellular receptor for calcium ions that functions by changing its shape upon binding to calcium so that it can bind to and activate/inactivate other proteins. Most proteins activitated by calmodulin are so-called CaM-kinases.

Chemotaxis: This is a bacterial signaling pathway involved in chemotaxis. Repellents activate receptors that, with the assistance of CheW, activate CheA. Attractants inhibit CheA. CheA activates CheY which causes the flagella to rotate such that the bacteria tumble. CheZ inactivates CheY.

Rhodopsin and Gt: Rhodopsin is a 7-pass transmembrane G-protein linked receptor containing a pigment, 11-cis-retinal. Light changes the structure of the pigment which causes Rhodopsin to bind with transducin (Gt), a trimeric G-protein. Upon binding, Gt looses its alpha subunit which diffuses and binds GMP phosphotase, activating it and eventually leading to signaling.

U1 U2 U5 U4 U6 spliceosome: The spliceosome is a protein, RNA complex reponsible for splicing introns out of nascient mRNA during its maturation. U1, U2, U4, U5 and U6 are among the different snRNPs present in Eukaryotic nuclei - they consist of both protein and small RNA molecules.

Ubiquitin: Ubiquitin-dependent protein degradation plays a role in many cellular processes including transcriptional regulation, cell cycle progression and DNA repair. Ubiquitin is a highly conserved 8kDa protein whose many cellular functions are mediated by its covalent ligation to other proteins.

Apoptosis: Apoptosis, or programmed cell death, plays a fundamental role during tissue development, injury and degeneration. The biochemical pathways of programmed cell death are also used to destroy cells with damaged DNA and cells that are infected with viruses.

p53 Signaling Pathway: p53 is a transcription factor whose main function is to prevent the cell from progressing through the cell cycle when DNA damage has occurred. p53 may either halt the cell cycle until the DNA can be repaired or else it may cause the cell to undergo apoptosis.

Insulin Signaling Pathway: Insulin, a small protein that acts as a hormone, is secreted by the pancreas in response to increased glucose levels in the blood. Most cells of the body have receptors which bind insulin. Upon binding of insulin, the cell activates other receptors designed to absorb glucose from the blood stream into the cell. Insulin is a necessary hormone and insulin deficiency or resistance results in diabetes.

6.3. Expert Analysis

The domain expert evaluated sixteen clusters – two clusters for each topic of interest, e.g, calmodlin, apoptosis, etc. Three different types of entities were considered (PDB, SWISS-PROT and GO terms) to determine how many of them were relevant to the topic of interest. Although GO was not used to obtain the clusters in any way – therefore, they are not in the clusters, GO terms were assigned to the clusters using SWISS-PROT to GO mappings as described in subsection 6.4. Also, overall cluster qualities are reported for each cluster manually examined.

Table 3 shows the evaluation results judged by the domain expert. For each entity type, a relevancy score between 1 and 10 was assigned where 10 means all entities of that particular type are highly topical and 1 means that none of them are relevant. Almost all entity types for all clusters have high scores. Therefore, we can conclude that all the sections evaluated by the expert are highly relevant to the topics considered.

Topic	**Cluster**	PDB	SW	GO term	Overall
calmodulin	1794	10	10	8	10
calmodulin	1815	5	7	5	5
rhodopsin	1402	10	9	10	10
rhodopsin	1400	3	5	10	7
spliceosome	1634	10	10	10	10
spliceosome	1648	N/A	8	6	7
chemotaxis	1072	9	9	9	9
chemotaxis	1071	5	6	4	5
apoptosis	1670	10	10	10	10
apoptosis	1669	10	9	10	10
ubiquitin	1665	7	2	8	8
ubiquitin	1666	7	10	9	8
insulin	1473	10	10	10	9
insulin	1472	10	7	10	9
p53	1674	10	8	10	9
p53	1722	7	7	7	8

Table 3. Evaluation of sample clusters by our domain expert. Scores range from 1 to 10, where 10 means all of the objects are relevant, and 1 means none of them are relevant.

Biologists note that the SWISS-PROT to GO mapping is incomplete because not all SWISS-PROT sequences are fully annotated. For example, in one cluster for the topic "apoptosis", the SWISS-PROT gene for E1B is not annotated with apoptosis even though it is involved in apoptosis. Similarly, in another cluster for the topic "apoptosis", the SWISS-PROT annotation for the CASP-1 genes do not refer to apoptosis, but some MEDLINE articles indicate that it is involved in apoptosis. So, since the SWISS-PROT GO annotation is incomplete, the relevance scores that we obtain based on the GO terms through automated means may underestimate the relevance of the cluster contents.

Another interesting point is that our domain expert first thought that 30S ribosomal protein in cluster 1665 was unrelated to ubiquitin, therefore assigned a score of 2. However, the immediate links to MEDLINE articles as provided by our system suggested that it should be relevant. We asked to reconsider whether 30S ribosomal protein could be related to ubiquitin, and the SWISS-PROT sequences in this cluster were reevaluated. After examining some immediate neighbors (MEDLINE articles) of these entities in the graph, our domain expert found out that they were indeed relevant to ubiqutin, and now believes that the GO terms assigned to these clusters (nucleus and structural constituent of ribosome) are very relevant to ubiquitin. This 'discovery' aspect of our system is important – it demonstrates that the clusters can bring to light relationships that are not obvious at first glance.

6.4. Correlation between clusters and GO categories – GO Term Assignment to Clusters

The Gene Ontology Consortium (2000) produces a controlled vocabulary for genes and gene products, called GO. GO[3] provides three structured networks of defined terms to describe gene product attributes. These three GO ontologies are referred to as Biological Process, Molecular Function and Cellular Component.

To show how much correlation we obtained between clusters and GO categories, we assigned GO terms to the clusters using the SWISS-PROT to GO mappings. Before explaining how we did this, it is important to note that we did not use GO to construct the graph; we just use it to provide a biological validation.

p-value Calculation:

Consider a two class population, and suppose we take a sample of size n from this population. Let A and B represent the number of objects for two classes, and N be the total number of objects. Let a, b and n be the corresponding numbers in our sample population. Thus, $A + B = N$, and $a + b = n$. Let us define the hypotheses

$$H_0 : \text{type-}A \text{ objects appear at random,}$$
$$H_a : \text{not at random.}$$

We reject the null hypothesis H_0 if

$$\text{p-value} = \textstyle\sum_{x \geq a} \binom{A}{x}\binom{B}{n-x} / \binom{N}{n}, \quad (5)$$

i.e., the probability of observing at least a numbers of class A at random is close to 0, e.g., p-value ≤ 0.001. Since the calculation of (5) is computationally expensive, we use an approximation instead. If the objects were selected with replacement, then the number of type-A objects in a sample of size n will have approximately Binomial distribution with success probability $p = A/N$, and the p-value becomes

$$\text{p-value} = \textstyle\sum_{x \geq a} \binom{n}{x} p^x (1-p)^{n-x}. \quad (6)$$

We consider the entire set of Gene Ontology (GO) annotations for the clusters. For each GO term for SWISS-PROT sequences within a specific cluster, we compute a p-value as in (6) where

N = total number of SWISS-PROT sequences,

A = actual number of A GO category SWISS-PROT sequences,

n = number of SWISS-PROT sequences in the cluster,

a = number of A GO category SWISS-PROT sequences in the cluster,

p = the proportion of the SWISS-PROT sequences containing A GO category.

[3]http://www.geneontology.org/

We give a general GO biological assessment to a cluster based on those GO annotations with p-values of less than 0.001.

Table 4 presents the GO term assignments to the selected sample clusters. As can be seen from these tables, GO terms assigned to the clusters are also highly topically related.

6.5. Analysis of Abstracts by descriptive keywords

Table 5 shows the twenty most significant words extracted from the articles in clusters as described in subsection 5.1. These words are highly topically relevant to the main topics considered for all but one cluster.

7. Summary

The relational structure of biological data has heretofore mainly served to facilitate browsing. Here we have used the implied graph as a computational object, partitioned it using standard techniques, and thus produced clusters of biological objects. These clusters exhibit strong topicality, as measured by both quantitative and qualitative manual evaluations, and by concentration of keywords and protein classifications. Because computation can be done on a large scale, these clusters reveal relationships that manual traversal of the graph do not. Furthermore, we believe that treating the graph as a computational object has applications in addition to producing topical clusters– for example, to information retrieval and data mining. In our future work, we plan to investigate statistical relational learning algorithms to predict links between biological objects for knowledge discovery.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 9986085. We thank David Synder for evaluating the sample clusters, and for his helpful comments. We thank Rose Oughtred for helpful discussions.

References

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI–BLAST: A new generation of protein database search programs. *Nucleic Acids Res.*, *25*, 3389–3402.

Bairoch, A., & Apweiler, R. (2000). The SWISS–PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res*, *28*, 45–48.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., & Bourne, P. E. (2000). The Protein Data Bank. *Nucleic Acids Research*, *28*, 235–242.

Karypis, G., & Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, *20*, 359–392.

Kirsten, M., Wrabel, S., & Horvath, T. (2000). Distance-based approaches to relational learning and clustering. In *Relational data mining*, 213–230. Springer-Verlag New York, Inc.

Murzin, A. G., Brenner, S. E., Hubbard, T., & Chothia, C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, *247*, 536–540.

The Gene Ontology Consortium (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, *25*, 25–29.

Witten, I. H., Moffat, A., & Bell, T. C. (1999). *Managing gigabytes: Compressing and indexing documents and images*. San Francisco, CA: Morgan Kaufmann. 2 edition.

Chemotaxis cluster 1072					
go id	**go name**	**a**	**n**	**A**	**-log(p_value)**
GO:0007600	sensory perception (BP)	52	83	412	243.02
GO:0006935	chemotaxis (BP)	33	83	300	144.67
GO:0030435	sporulation (BP)	19	83	404	66.11

Apoptosis cluster 1670					
go id	**go name**	**a**	**n**	**A**	**-log(p_value)**
GO:0006915	apoptosis (BP)	65	88	318	337.88
GO:0008234	cysteine-type peptidase (MF)	40	88	462	164.63
GO:0016787	hydrolase (MF)	43	88	10857	49.19
GO:0005634	nucleus (CC)	13	88	5856	8.04

Insulin cluster 1473					
go id	**go name**	**a**	**n**	**A**	**-log(p_value)**
GO:0019838	growth factor binding (MF)	31	40	48	223.3
GO:0005179	hormone (MF)	8	40	1082	19.81
GO:0005180	peptide hormone (MF)	3	40	276	9.1

Table 4. GO assignments of the sample clusters. In the GO term column, the GO annotation types are shown in parentheses: BP, MF and CC stand for biological_process, molecular_function and cellular_component, respectively. a is the number of the particular category SWISS-PROT sequences in the cluster, n denotes the number of SWISS-PROT sequences in the cluster, and A is the actual number of the particular GO category SWISS-PROT sequences.

Topic	**Cluster**	**Descriptive words extracted from the MEDLINE articles in clusters**
calmodulin	1794	calmodulin n-cam calmodulin-dependent cam-dependent cams cam-binding adhesion ca ca2 cabp neural brain ng-cam kinase calcium domain calcium-binding chicken calcium-dependent molecule
rhodopsin	1402	bacteriorhodopsin retinal schiff rhodopsins chromophore light-driven halorhodopsin proton pump halobacterium photocycle pharaonis asp85 transmembrane light phototaxis opsin visual pumping retinal-binding
spliceosome	1634	splicing snrnp u1 u2 ribonucleoprotein spliceosome sr nuclear sf2 asf rna pre-mrnas rna-binding u2af factor rs alternative factors splice spliceosomal
chemotaxis	1072	chemotaxis chea cheb chew cher response regulator bacterial swimming chez flagellar phosphorylation phosphotransfer chemotactic salmonella typhimurium swarm transduction flim crystal
apoptosis	1670	apoptosis death apoptotic fadd cd95 programmed necrosis apo-1 fas-mediated fasl flice mort1 cell daxx fas-induced effector tumor cells death-inducing signaling
ubiquitin	1665	polyubiquitin ubiquitin-specific ubiquitin-like deubiquitinating ubp ubi ubiquitin-dependent extension ubiquitins conjugates fusion ubiquitin-beta-galactosidase ubiquitin-encoding degradation ribosomal nedd8 repeats ubiquitin-activating tetraubiquitin pr
insulin	1473	insulin-like igfbps igfbp igfbp-1 diabetes igf growth autophosphorylation igfs factor-binding receptor igfbp-2 insulin-stimulated igfbp-3 mellitus igfbp-5 igfbp-4 igf-i factor igf-binding
p53	1674	tumor suppressor cancer p53-dependent tumors p53-binding p53-mediated cancers lines li-fraumeni carcinomas tumor-suppressor mutations cell human tp53 tumour damage breast carcinoma

Table 5. The most significant words extracted from the MEDLINE articles in each cluster; sorted in ascending order of p-value. The extracted words within each sample cluster are highly topically related.

Markov Logic: A Unifying Framework for Statistical Relational Learning

Pedro Domingos PEDROD@CS.WASHINGTON.EDU
Matthew Richardson MATTR@CS.WASHINGTON.EDU
Department of Computer Science and Engineeering, University of Washington, Seattle, WA 98195-2350, USA

Abstract

Interest in statistical relational learning (SRL) has grown rapidly in recent years. Several key SRL tasks have been identified, and a large number of approaches have been proposed. Increasingly, a unifying framework is needed to facilitate transfer of knowledge across tasks and approaches, to compare approaches, and to help bring structure to the field. We propose *Markov logic* as such a framework. Syntactically, Markov logic is indistinguishable from first-order logic, except that each formula has a weight attached. Semantically, a set of Markov logic formulas represents a probability distribution over possible worlds, in the form of a log-linear model with one feature per grounding of a formula in the set, with the corresponding weight. We show how approaches like probabilistic relational models, knowledge-based model construction and stochastic logic programs are special cases of Markov logic. We also show how tasks like collective classification, link prediction, link-based clustering, social network modeling, and object identification can be concisely formulated in Markov logic. Finally, we briefly describe learning and inference algorithms for Markov logic, and report positive results on a link prediction task.

1. The Need for a Unifying Framework

Many (if not most) real-world application domains are characterized by the presence of both uncertainty and complex relational structure. Statistical learning focuses on the former, and relational learning on the latter. Statistical relational learning (SRL) seeks to combine the power of both. Research in SRL has expanded rapidly in recent years, both because of the need for it in applications, and because statistical and relational learning have individually matured to the point where combining them is a feasible research enterprise. A number of key SRL tasks have been identified, including collective classification, link prediction, link-based clustering, social network modeling, object identification, and others. A large and growing number of SRL approaches have been proposed, including knowledge-based model construction (Wellman et al., 1992; Ngo & Haddawy, 1997; Kersting & De Raedt, 2001), stochastic logic programs (Muggleton, 1996; Cussens, 1999), PRISM (Sato & Kameya, 1997), probabilistic relational models (Friedman et al., 1999), relational Markov models (Anderson et al., 2002), relational Markov networks (Taskar et al., 2002), relational dependency networks (Neville & Jensen, 2003), structural logistic regression (Popescul & Ungar, 2003), relational generation functions (Cumby & Roth, 2003), CLP(BN) (Costa et al., 2003), and others.

While the variety of problems and approaches in the field is valuable, it makes it difficult for researchers, students and practitioners to identify, learn and apply the essentials. In particular, for the most part, the relationships between different approaches and their relative strengths and weaknesses remain poorly understood, and innovations in one task or application do not easily transfer to others, slowing down progress. There is thus an increasingly pressing need for a unifying framework, a common language for describing and relating the different tasks and approaches. To be most useful, such a framework should satisfy the following desiderata:

1. *The framework must subsume both first-order logic and probabilistic graphical models.* Otherwise some current or future SRL approaches will

fall outside its scope.

2. *SRL problems should be representable clearly and simply in the framework.*

3. *The framework must facilitate the incorporation of domain knowledge into SRL.* Because the search space for SRL algorithms is very large even by AI standards, domain knowledge is critical to success. Conversely, the ability to incorporate rich domain knowledge is one of the most attractive features of SRL.

4. *The framework should facilitate the extension to SRL of techniques from statistical learning, inductive logic programming, probabilistic inference and logical inference.* This will speed progress in SRL by taking advantage of the large extant literature in these areas.

In the next section we propose a framework that we believe meets all of these desiderata. We then describe how several SRL approaches and tasks can be formulated in this framework. Finally, we illustrate how existing learning and inference techniques can be applied within it to yield practical algorithms.

2. Markov Logic

Markov logic is a simple yet powerful combination of Markov networks and first-order logic. Recall that a Markov network is a model for the joint distribution of a set of variables $X = (X_1, X_2, \ldots, X_n) \in \mathcal{X}$ (Pearl, 1988), often conveniently represented as a *log-linear model*:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x)\right) \tag{1}$$

where Z is a normalization factor, and the $f_j(x)$'s are *features* of the state x (i.e., functions with $\mathcal{X}$ as the domain). Essentially every probabilistic model of interest to SRL can be represented as a Markov network or log-linear model, including Bayesian networks, decision trees, logistic regression, etc. *Markov logic* raises the expressiveness of Markov networks to encompass first-order logic. Recall that a first-order domain is defined by a set of *constants* (which we assume finite) representing objects in the domain (e.g., `Anna`, `Bob`) and a set of *predicates* representing properties of those objects and relations between them (e.g., `Smokes(x)`, `Friend(x,y)`). (For simplicity, we ignore functions in this paper; see Richardson and Domingos (2004) for a more complete treatment.) A predicate can be *grounded* by replacing its variables with constants (e.g., `Smokes(Anna)`, `Friend(Anna,Bob)`). A *world* assigns a truth value to each possible ground predicate. A first-order knowledge base (KB) is a set of formulas in first-order logic, constructed from predicates using logical connectives and quantifiers. Essentially all the relational languages used in SRL (e.g., logic programs, frame-based systems, database query languages) are special cases of first-order logic.

A formula in Markov logic is a formula in first-order logic with an associated weight. We call a set of formulas in Markov logic a *Markov logic network* or MLN. MLNs define probability distributions over possible worlds (Halpern, 1990) as follows.

Definition 2.1 *A Markov logic network L is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ (Equation 1) as follows:*

1. *$M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground predicate is true, and 0 otherwise.*

2. *$M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L.*

A first-order KB can be seen as a set of hard constraints on the set of possible worlds: if a world violates even one formula, it has zero probability. The basic idea in Markov logic is to soften these constraints: when a world violates one formula in the KB it is less probable, but not impossible. The fewer formulas a world violates, the more probable it is. A formula's associated weight reflects how strong a constraint it is: the higher the weight, the greater the difference in log probability between a world that satisfies the formula and one that does not, other things being equal. As weights increase, an MLN increasingly resembles a purely logical KB. In the limit of all infinite weights, the MLN represents a uniform distribution over the worlds that satisfy the KB.

An MLN without variables (i.e., containing only ground formulas) is an ordinary Markov network. Any log-linear model over Boolean variables can be represented as an MLN, since each state of a Boolean clique is defined by a conjunction of literals. (This extends trivially to discrete variables, and to binary encoding of numeric variables.)

An MLN can be viewed as a *template* for constructing Markov networks. In different worlds (different sets of constants) it will produce different networks, and these may be of widely varying size, but all will have certain regularities in structure and parameters, given by the MLN (e.g., all groundings of the same formula will have the same weight).

3. SRL Approaches

Since Markov logic subsumes first-order logic and probabilistic graphical models, it subsumes all representations used in SRL that are formed from special cases of them. However, it is enlightening to see how these representations map into Markov logic, and here we informally do this for a few of the most popular ones.

3.1. Knowledge-Based Model Construction

Knowledge-based model construction (KBMC), the oldest SRL approach (Wellman et al., 1992; Ngo & Haddawy, 1997; Kersting & De Raedt, 2001), is a combination of logic programming and Bayesian networks. KBMC, like all other SRL approaches based on logic programming, is a restriction of Markov logic to KBs containing only Horn clauses. As in Markov logic, nodes in KBMC represent ground predicates. The parents of a node are the predicates appearing in the bodies of Horn clauses having the node as a consequent. The conditional probability of a node given the truth values of its parent rule bodies is specified by a combination function (e.g., noisy OR, logistic regression, arbitrary CPT). A KBMC model is translated into Markov logic by writing down a set of formulas for each first-order predicate Pk(...) in the domain. Each formula is a conjunction containing Pk(...) and one literal per parent of Pk(...) (i.e., per first-order predicate appearing in a Horn clause having Pk(...) as the consequent). A subset of these literals are negated; there is one formula for each possible combination of positive and negative literals. The weight of the formula is $w = \log[p/(1-p)]$, where p is the conditional probability of the child predicate when the corresponding conjunction of parent literals is true, according to the combination function used. If the combination function is logistic regression, it can be represented using only a linear number of formulas, taking advantage of the fact that it is a (conditional) Markov network with a binary clique between each predictor and the response. Noisy OR can similarly be represented with a linear number of parents.

3.2. Stochastic Logic Programs

Stochastic logic programs (SLPs) (Muggleton, 1996; Cussens, 1999) are a combination of logic programming and log-linear models. Puech and Muggleton (2003) showed that SLPs are a special case of KBMC, and thus they can be represented in Markov logic in the same way.

3.3. Probabilistic Relational Models

Probabilistic relational models (PRMs) (Friedman et al., 1999) are a combination of frame-based systems and Bayesian networks. PRMs can be represented in Markov logic by defining a predicate $\mathtt{S}(\mathtt{x},\mathtt{v})$ for each (propositional or relational) attribute of each class, where $\mathtt{S}(\mathtt{x},\mathtt{v})$ means "The value of attribute S in object x is v." A PRM is then translated into Markov logic by writing down a formula for each line of each (class-level) conditional probability table (CPT) and value of the child attribute. The formula is a conjunction of literals stating the parent values and a literal stating the child value, and its weight is the logarithm of $P(x|Parents(x))$, the corresponding entry in the CPT. In addition, the MLN contains formulas with infinite weight stating that each attribute must take exactly one value. Notice that this approach handles all types of uncertainty in PRMs (attribute, reference and existence uncertainty).

3.4. Relational Markov Networks

Relational Markov networks (RMNs) (Taskar et al., 2002) are a combination of Markov networks and conjunctive queries, a subset of the SQL database query language. An RMN is simply an MLN with a formula (in particular, a conjunction of literals) for each possible state of each clique template in the RMN, with the corresponding weight.

3.5. Structural Logistic Regression

In structural logistic regression (SLR) (Popescul & Ungar, 2003), the predictors are the output of SQL queries over the input data. Just as a logistic regression model is a discriminatively-trained Markov network, an SLR model is a discriminatively-trained MLN.[1]

3.6. Relational Dependency Networks

In a relational dependency network (RDN), each node's probability conditioned on its Markov blanket is

[1]Use of SQL aggregates requires that their definitions be imported into the MLN.

given by a decision tree (Neville & Jensen, 2003). Every RDN has a corresponding MLN in the same way that every dependency network has a corresponding Markov network, given by the stationary distribution of a Gibbs sampler operating on it (Heckerman et al., 2000).

4. SRL Tasks

In this section, we show how key SRL tasks can be concisely formulated in Markov logic, making it possible to bring the full power of logical and statistical learning and inference approaches to bear on them.

4.1. Collective Classification

The goal of ordinary classification is to predict the class of an object given its attributes. In collective classification, we also take into account the classes of related objects. Attributes can be represented in Markov logic as predicates of the form $\mathtt{A(x,v)}$, where $\mathtt{A}$ is an attribute, $\mathtt{x}$ is an object, and $\mathtt{v}$ is the value of $\mathtt{A}$ in $\mathtt{x}$. The class is a designated attribute $\mathtt{C}$, representable by $\mathtt{C(x,v)}$, where $\mathtt{v}$ is $\mathtt{x}$'s class. Classification is now simply the problem of inferring the truth value of $\mathtt{C(x,v)}$ for all $\mathtt{x}$ and $\mathtt{v}$ of interest given all known $\mathtt{A(x,v)}$. Ordinary classification is the special case where $\mathtt{C(xi,v)}$ and $\mathtt{C(xj,v)}$ are independent for all $\mathtt{xi}$ and $\mathtt{xj}$ given the known $\mathtt{A(x,v)}$. In collective classification, the Markov blanket of $\mathtt{C(xi,v)}$ includes other $\mathtt{C(xj,v)}$, even after conditioning on the known $\mathtt{A(x,v)}$. Relations between objects are represented by predicates of the form $\mathtt{R(xi,xj)}$. A number of interesting generalizations are readily apparent, for example $\mathtt{C(xi,v)}$ and $\mathtt{C(xj,v)}$ may be indirectly dependent via unknown predicates, possibly including the $\mathtt{R(xi,xj)}$ predicates themselves. Background knowledge can be incorporated by stating it in first-order logic, learning weights for the resulting formulas, and possibly refining them (see Richardson and Domingos (2004) for an example).

4.2. Link Prediction

The goal of link prediction is to determine whether a relation exists between two objects of interest (e.g., whether Anna is Bob's Ph.D. advisor) from the properties of those objects and possibly other known relations. The formulation of this problem in Markov logic is identical to that of collective classification, with the only difference that the goal is now to infer the value of $\mathtt{R(xi,xj)}$ for all object pairs of interest, instead of $\mathtt{C(x,v)}$.

4.3. Link-Based Clustering

The goal of clustering is to group together objects with similar attributes. In model-based clustering, we assume a generative model $P(X) = \sum_C P(C)P(X|C)$, where X is an object, C ranges over clusters, and $P(C|X)$ is X's degree of membership in cluster C. In link-based clustering, objects are clustered according to their links (e.g., objects that are more closely related are more likely to belong to the same cluster), and possibly according to their attributes as well. This problem can be formulated in Markov logic by postulating an unobserved predicate $\mathtt{C(x,v)}$ with the meaning "$\mathtt{x}$ belongs to cluster $\mathtt{v}$," and having formulas in the MLN involving this predicate and the observed ones (e.g., $\mathtt{R(xi,xj)}$ for links and $\mathtt{A(x,v)}$ for attributes). Link-based clustering can now be performed by learning the parameters of the MLN, and cluster memberships are given by the probabilities of the $\mathtt{C(x,v)}$ predicates conditioned on the observed ones.

4.4. Social Network Modeling

Social networks are graphs where nodes represent social actors (e.g., people) and arcs represent relations between them (e.g., friendship). Social network analysis (Wasserman & Faust, 1994) is concerned with building models relating actors' properties and their links. For example, the probability of two actors forming a link may depend on the similarity of their attributes, and conversely two linked actors may be more likely to have certain properties. These models are typically Markov networks, and can be concisely represented by formulas like $\forall \mathtt{x} \forall \mathtt{y} \forall \mathtt{v}\, \mathtt{R(x,y)} \Rightarrow (\mathtt{A(x,v)} \Leftrightarrow \mathtt{A(y,v)})$, where $\mathtt{x}$ and $\mathtt{y}$ are actors, $\mathtt{R(x,y)}$ is a relation between them, $\mathtt{A(x,v)}$ represents an attribute of $\mathtt{x}$, and the weight of the formula captures the strength of the correlation between the relation and the attribute similarity. For example, a model stating that friends tend to have similar smoking habits can be represented by the formula $\forall \mathtt{x} \forall \mathtt{y}\, \mathtt{Friends(x,y)} \Rightarrow (\mathtt{Smokes(x)} \Leftrightarrow \mathtt{Smokes(y)})$. Notice that this formula is false as a universally quantified statement in first-order logic, but is true in some domains as a probabilistic statement in Markov logic (Lloyd-Richardson et al., 2002). As well as encompassing existing social network models, Markov logic allows richer ones to be easily stated (e.g., by writing formulas involving multiple types of relations and multiple attributes, as well as more complex dependencies between them). These models can then be learned and applied using techniques like those in Richardson and Domingos (2004) (see next section).

4.5. Object Identification

Object identification (also known as record linkage, de-duplication, and others) is the problem of determining which records in a database refer to the same real-world entity (e.g., which entries in a bibliographic database represent the same publication). This problem is of crucial importance to many companies, government agencies, and large-scale scientific projects. One way to represent it in Markov logic is by defining a predicate $\texttt{Same}(\texttt{x},\texttt{y})$ with the meaning "$\texttt{x}$ represents the same real-world entity as $\texttt{y}$." This predicate is applied both to records and their fields (e.g., $\texttt{Same}$("ICML", "Intl. Conf. on Mach. Learn.")). The dependencies between record matches and field matches can then be represented by formulas like $\forall\texttt{x}\forall\texttt{y}\ \texttt{Same}(\texttt{x},\texttt{y}) \Leftrightarrow \texttt{Same}(\texttt{fi}(\texttt{x}),\texttt{fi}(\texttt{y}))$, where $\texttt{x}$ and $\texttt{y}$ are records and $\texttt{fi}(\texttt{x})$ is a function returning the value of the $\texttt{i}$th field of record $\texttt{x}$. We have successfully applied this approach to de-duplicating the Cora database of computer science papers (Parag & Domingos, 2004). Because it allows information to propagate from one match decision (i.e., one grounding of $\texttt{Same}(\texttt{x},\texttt{y})$) to another via fields that appear in both pairs of records, it effectively performs collective object identification, and in our experiments outperformed the traditional method of making each match decision independently of all others. For example, matching two references may allow us to determine that "ICML" and "MLC" represent the same conference, which in turn may help us to match another pair of references where one contains "ICML" and the other "MLC." Markov logic also allows additional information to be incorporated into a de-duplication system easily, modularly and uniformly. For example, transitive closure is incorporated by adding the formula $\forall\texttt{x}\forall\texttt{y}\forall\texttt{z}\ \texttt{Same}(\texttt{x},\texttt{y}) \wedge \texttt{Same}(\texttt{y},\texttt{z}) \Rightarrow \texttt{Same}(\texttt{x},\texttt{z})$, with a weight that can be learned from data.

5. Implementation

In principle, any inductive logic programming (ILP) approach can be used to learn the structure of an MLN, and any approach for learning Markov network parameters (e.g., conjugate gradient or iterative scaling) can be used to learn the weights. Likewise, any method for inference in Markov networks (e.g., Markov chain Monte Carlo, belief propagation) can be used to perform inference in grounded MLNs, and logical inference methods can be used to construct the subsets of these networks relevant to a particular query. Logical inference can also be used to find modes of the distribution, which, if the KB is satisfiable and all weights are positive, are the satisfying assignments of truth values to ground predicates. When no satisfying assignments exist, modes can still be found using methods like MaxWalkSat, a variation of the WalkSat satisfiability search algorithm for finding truth assignments that maximize the sum of weights of satisfied clauses (Selman et al., 1996).

In Richardson and Domingos (2004), we describe one possible implementation of Markov logic, using MaxWalkSat and Gibbs sampling for inference, the CLAUDIEN ILP system (De Raedt & Dehaspe, 1997) for structure learning, and a pseudo-likelihood method for parameter learning (Besag, 1975). We have tested this approach on a link prediction task (predicting which students are advised by which faculty from a multi-relational database describing our department), and found that it outperforms a purely relational learner (CLAUDIEN), a purely statistical learner (Bayesian networks, restricted or not to a naive Bayes structure), and a pure knowledge-based approach (manually constructed first-order KB).

6. Conclusion

The rapid growth in the variety of SRL approaches and tasks has led to the need for a unifying framework. In this paper we propose *Markov logic* as a candidate for such a framework. Markov logic subsumes first-order logic and Markov networks, and allows a wide variety of SRL tasks and approaches to be formulated in a common language. Initial experiments with an implementation of Markov logic have yielded good results.

Acknowledgements

We are grateful to Julian Besag, Nilesh Dalvi, Alon Halevy, Henry Kautz, Tian Sang, Bart Selman, Dan Suciu, and Wei Wei for helpful discussions. This research was partly supported by an IBM Ph.D. Fellowship to the first author, and by ONR grant N00014-02-1-0408.

References

Anderson, C., Domingos, P., & Weld, D. (2002). Relational Markov models and their application to adaptive Web navigation. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 143–152). Edmonton, Canada: ACM Press.

Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician*, *24*, 179–195.

Costa, V. S., Page, D., Qazi, M., , & Cussens, J.

(2003). CLP(BN): Constraint logic programming for probabilistic knowledge. *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence* (pp. 517–524). Acapulco, Mexico: Morgan Kaufmann.

Cumby, C., & Roth, D. (2003). Feature extraction languages for propositionalized relational learning. *Proceedings of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data* (pp. 24–31). Acapulco, Mexico: IJCAII.

Cussens, J. (1999). Loglinear models for first-order probabilistic reasoning. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (pp. 126–133). Stockholm, Sweden: Morgan Kaufmann.

De Raedt, L., & Dehaspe, L. (1997). Clausal discovery. *Machine Learning*, *26*, 99–146.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (pp. 1300–1307). Stockholm, Sweden: Morgan Kaufmann.

Halpern, J. (1990). An analysis of first-order logics of probability. *Artificial Intelligence*, *46*, 311–350.

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., & Kadie, C. (2000). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, *1*, 49–75.

Kersting, K., & De Raedt, L. (2001). Towards combining inductive logic programming with Bayesian networks. *Proceedings of the Eleventh International Conference on Inductive Logic Programming* (pp. 118–131). Strasbourg, France: Springer.

Lloyd-Richardson, E., Kazura, A., Stanton, C., Niaura, R., & Papandonatos, G. (2002). Differentiating stages of smoking intensity among adolescents: Stage-specific psychological and social influences. *Journal of Consulting and Clinical Psychology*, *70*.

Muggleton, S. (1996). Stochastic logic programs. In L. de Raedt (Ed.), *Advances in inductive logic programming*, 254–264. Amsterdam, Netherlands: IOS Press.

Neville, J., & Jensen, D. (2003). Collective classification with relational dependency networks. *Proceedings of the Second International Workshop on Multi-Relational Data Mining* (pp. 77–91). Washington, DC: ACM Press.

Ngo, L., & Haddawy, P. (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, *171*, 147–177.

Parag, & Domingos, P. (2004). *Collective record linkage* (Technical Report). Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://www.cs.washington.edu/-homes/pedrod/crl.pdf.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco, CA: Morgan Kaufmann.

Popescul, A., & Ungar, L. H. (2003). Structural logistic regression for link analysis. *Proceedings of the Second International Workshop on Multi-Relational Data Mining* (pp. 92–106). Washington, DC: ACM Press.

Puech, A., & Muggleton, S. (2003). A comparison of stochastic logic programs and Bayesian logic programs. *Proceedings of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data* (pp. 121–129). Acapulco, Mexico: IJCAII.

Richardson, M., & Domingos, P. (2004). *Markov logic networks* (Technical Report). Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://www.cs.washington.-edu/homes/pedrod/mln.pdf.

Sato, T., & Kameya, Y. (1997). PRISM: A symbolic-statistical modeling language. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (pp. 1330–1335). Nagoya, Japan: Morgan Kaufmann.

Selman, B., Kautz, H., & Cohen, B. (1996). Local search strategies for satisfiability testing. In D. S. Johnson and M. A. Trick (Eds.), *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge*, 521–532. Washington, DC: American Mathematical Society.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence* (pp. 485–492). Edmonton, Canada: Morgan Kaufmann.

Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications*. Cambridge, UK: Cambridge University Press.

Wellman, M., Breese, J. S., & Goldman, R. P. (1992). From knowledge bases to decision models. *Knowledge Engineering Review*, *7*.

Probabilistic Entity-Relationship Models, PRMs, and Plate Models

David Heckerman HECKERMA@MICROSOFT.COM
Christopher Meek MEEK@MICROSOFT.COM
One Microsoft Way, Redmond, WA 98052

Daphne Koller KOLLER@CS.STANFORD.EDU
Computer Science Department, Stanford, CA 94305

Abstract

We introduce a graphical language for relational data called the probabilistic entity-relationship (PER) model. The model is an extension of the entity-relationship model, a common model for the abstract representation of database structure. We concentrate on the directed version of this model—the directed acyclic probabilistic entity-relationship (DAPER) model. The DAPER model is closely related to the plate model and the probabilistic relational model (PRM), existing models for relational data. The DAPER model is more expressive than either existing model, and also helps to demonstrate their similarity.

1. Introduction

For over a century, statistical modeling has focused primarily on "flat" data—data that can be encoded naturally in a single two-dimensional table having rows and columns. The disciplines of pattern recognition, machine learning, and data mining have had a similar focus. Notable exceptions include hierarchical models (e.g., Good, 1965) and spatial statistics (e.g., Besag, 1974). Over the last decade, however, perhaps due to the ever increasing volumes of data being stored in databases, the modeling of non-flat or *relational data* has increased significantly. During this time, several graphical languages for relational data have emerged including plate models (e.g., Buntine, 1994; Spiegelhalter, 1998) and probabilistic relational models (PRMs) (e.g., Friedman, Getoor, Koller, and Pfeffer, 1999. These models are to relational data what ordinary graphical models (e.g., directed-acyclic graphs and undirected graphs) are to flat data.

In this paper, we introduce a new graphical model for relational data—the probabilistic entity-relationship (PER) model. This model class is more expressive than either PRMs or plate models. We concentrate on a particular type of PER model—the directed acyclic probabilistic entity-relationship (DAPER) model—in which all probabilistic arcs are directed. It is this version of PER model that is most similar to the plate model and PRM. We define new versions of the plate model and PRM such their expressiveness is equivalent to the DAPER model, and then (in the expanded tech report, Heckerman, Meek, and Koller, 2004) compare the new and old definitions. Consequently, we both demonstrate the similarity among the original languages as well as enhance their abilities to express conditional independence in relational data. Our hope is that this demonstration of similarity will foster greater communication and collaboration among statisticians who mostly use plate models and computer scientists who mostly use PRMs.

We in fact began this work with an effort to unify traditional PRMs and plate models. In the process, we discovered that it was important to make both entities and relationships (concepts discussed in detail in the next section) first class objects in the language. We in turn discovered an existing language that does this—the entity-relationship (ER) model—a commonly used model for the abstract representation of database structure. We then extended this language to handle probabilistic relationships, creating the PER model.

We should emphasize that the languages we discuss are neither meant to serve as a database schema nor meant to be built on top of one. In practice, database schemas are built up over a long period of time as the needs of the database consumers change. Conse-

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the first author.

quently, schemas for real databases are often not optimal or are completely unusable as the basis for statistical modeling. The languages we describe here are meant to be used as statistical modeling tools, independent of the schema of the database being modeled.

This work borrows heavily from concepts surrounding PRMs described in (e.g.) Friedman et al. (1999) and Getoor et al. (2002). Where possible, we use similar nomenclature, notation, and examples.

2. ER Models

We begin with a description of a language for modeling the data itself. The language we discuss is the *entity-relationship (ER) model*, a commonly used abstract representation of database structure (e.g., Ullman and Widom, 2002). The creation of an ER model is often the first step in the process of building a relational database. Features of anticipated data and how they interrelate are encoded in an ER model. The ER model is then used to create a relational schema for the database, which in turn is used to build the database itself.

It is important to note that an ER model is a representation of a database structure, not of a particular database that contains data. That is, an ER model can be developed prior to the collection of any data, and is meant to anticipate the data and the relationships therein.

When building ER models, we distinguish between entities, relationships, and attributes. An *entity* corresponds to a thing or object that is or may be stored in a database or dataset[1]; a *relationship* corresponds to a specific interaction among entities; and an *attribute* corresponds to a variable describing some property of an entity or relationship. Throughout the paper, we use examples to illustrate concepts.

Example 1 A university database maintains records on students and their IQs, courses and their difficulty, and the courses taken by students and the grades they receive.

In this example, we can think of individual students (e.g., john, mary) and individual courses (e.g., cs107, stat10) as entities.[2] Naturally, there will be many students and courses in the database. We refer to the set of students (e.g., {john,mary,...}) as an *entity set.* The set of courses (e.g., {cs107,stat10,...}) is another entity set. Most important, because an ER model can be built before any data is collected, we need the concept of an *entity class*—a reference to a set of entities without a specification of the entities in the set. In our example, the entity classes are Student and Course.

A relationship is a tuple of pointers to entities—an indication that those referenced entities are somehow related. In our example, a possible relationship is the pair (john, cs107), meaning that john took the course cs107. Using nomenclature similar to that for entities, we talk about relationship sets and relationship classes. A *relationship set* is a collection of like relationships—that is, a collection of relationships each relating entities from a fixed list of entity classes. In our example, we have the relationship set of student-course pairs. A *relationship class* refers to an unspecified set of like relationships. In our example, we have the relationship class Takes.

The IQ of john and the difficulty of cs107 are examples of *attributes.* We use the term *attribute class* to refer to an unspecified collection of like attributes. In our example, Student has the single attribute class Student.IQ and Course has the single attribute class Course.Diff. Relationships also can have attributes; and relationship classes can have attribute classes. In our example, Takes has the attribute class Takes.Grade.

An ER model for the structure of a database graphically depicts entity classes, relationships classes, attribute classes, and their interconnections. An ER model for Example 1 is shown in Figure 1a. The entity classes (Student and Course) are shown as rectangular nodes; the relationship class (Takes) is shown as a diamond-shaped node; and the attribute classes (Student.IQ, Course.Diff, and Takes.Grade) are shown as oval nodes. Attribute classes are connected to their corresponding entity or relationship class, and the relationship class is connected to its associated entity classes. (Solid edges are customary in ER models. Here, we use dashed edges so that we can later use solid edges to denote probabilistic dependencies.)

An ER model describes the potential attributes and relationships in a database. It says little about actual data. A *skeleton for a set of entity and relationship classes* is specification of the entities and relationships associated with a particular database. That is, a skeleton for a set of entity and relationship classes is collection of corresponding entity and relationship sets. An example skeleton for our university-database example is shown in Figure 1b.

[1]In what follows, we make no distinction between a database and a dataset.

[2]In a real database, longer names would be needed to define unique students and courses. We keep the names short in our example to make reading easier.

An ER model *applied to a skeleton* defines a specific set of attributes. In particular, for every entity class and every attribute class of that entity class, an attribute is defined for every entity in the class; and, for every relationship class and every attribute class of that relationship class, an attribute is defined for every relationship in the class. The attributes defined by the ER model in Figure 1a applied to the skeleton in Figure 1b are shown in Figure 1c. In what follows, we use *ER model* to mean both the *ER diagram*—the graph in Figure 1a—and the mechanism by which attributes are generated from skeletons.

A skeleton still says nothing about the values of attributes. An *instance for an ER model* consists of (1) a skeleton for the entity and relationship classes in that model, and (2) an assignment of a value to every attribute generated by the ER model and the skeleton. That is, an instance of an ER model is an actual database.

3. PER Models

Let us now turn to the probabilistic modeling of relational data. To do so, we introduce a specific type of probabilistic entity-relationship model: the directed acyclic probabilistic entity-relationship (DAPER) model. Roughly speaking, a DAPER model is an ER model with directed (solid) arcs among the attribute classes that represent probabilistic dependencies among corresponding attributes, and local distribution classes that define local distributions for attributes. Recall that an ER model applied to a skeleton defines a set of attributes. Similarly, a DAPER model applied to a skeleton defines a set of attributes as well as a DAG model for these attributes. Thus, a DAPER model can be thought of as a language for expressing conditional independence among unrealized attributes that eventually become realized given a skeleton.

As with the ER diagram and model, we sometimes distinguish between a *DAPER diagram*, which consists of the graph only, and the *DAPER model*, which consists of the diagram, the local distribution classes, and the mechanism by which a DAPER model defines a DAG model given a skeleton.

Example 2 In the university database (Example 1), a student's grade in a course depends both on the student's IQ and on the difficulty of the course.

The DAPER model (or diagram) for this example is shown in Figure 2a. The model extends the ER model in Figure 1 with the addition of arc classes and local distribution classes. In particular, there is an *arc class* from Student.IQ to and arc class from Takes.Grade and from Course.Diff to Takes.Grade. These arc classes are denoted as a solid directed arc. In addition, there is a single local distribution class for Takes.Grade (not shown).

Just as we expand attribute classes in a DAPER model to attributes in a DAG model given a skeleton, we expand arc classes to arcs. In doing so, we sometimes want to limit the arcs that are added to a DAG model. In the current problem, for example, we want to draw an arc from attribute c.Diff for course c to attribute Takes(s, c').Grade for course c' and any student s, only when $c = c'$. This limitation is achieved by adding a *constraint* to the arc class—namely, the constraint course[Diff] = course[Grade] (see Figure 2a). Here, the terms "course[Diff]" and "course[Grade]" refer to the entities c and c', respectively—the entities associated with the attributes at the ends of the arc.

The arc class from Student.IQ to Takes.Grade has a similar constraint: student[IQ] = student[Grade]. This constraint says that we draw an arc from attribute s.IQ for student s =student[IQ] to Takes(s', c).Grade for student s'=student[Grade] and any course c only when $s = s'$. As we shall see, constraints in DAPER models can be quite expressive—for example, they may include first-order expressions on entities and relationships.

Figure 2c shows the DAG (structure) generated by the application of the DAPER model in Figure 2a to the skeleton in Figure 2b. (The attribute names in the DAG model are abbreviated.) The arc from stat10.Diff to Takes(mary,cs107).Grade (e.g.) is disallowed by the constraint on the arc class from Course.Diff to Takes.Grade.

Regardless of what skeleton we use, the DAG model generated by the DAPER model in Figure 2a will be acyclic. In general, as we show in Heckerman et al. (2004), if the attribute classes and arc classes in the DAPER diagram form an acyclic graph, then the DAG model generated from any skeleton for the DAPER model will be acyclic. Weaker conditions are also sufficient to guarantee acyclicity. We describe one in Heckerman et al. (2004).

In general, a *local distribution class* for an attribute class is a specification from which local distributions for attributes corresponding to the attribute class can be constructed, when a DAPER model is expanded to a DAG model. In our example, the local distribution class for Takes.Grade—written p(Takes.Grade|Student.IQ, Course.Diff)—is a

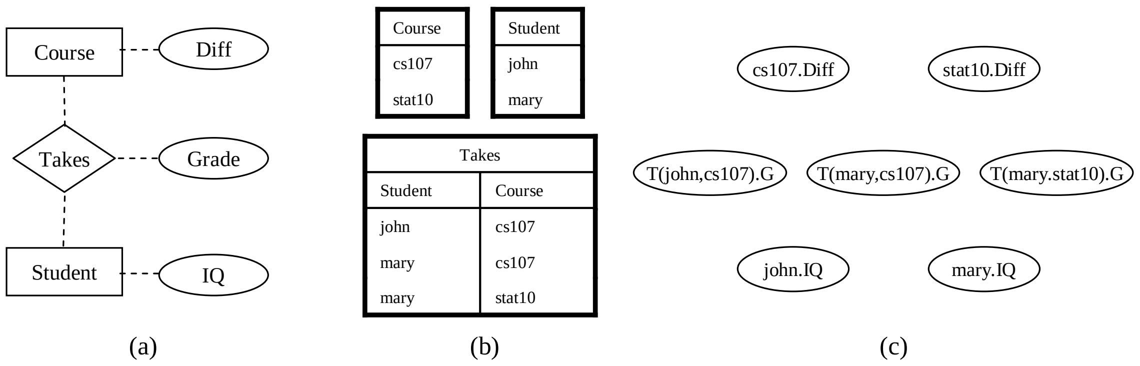


Figure 1. (a) An entity-relationship (ER) model depicting the structure of a university database. (b) An example skeleton for the entity and relationship classes in the ER model. (c) The attributes defined by the application of the ER model to the skeleton. The attribute names are abbreviated.

specification from which the local distributions for Takes(s, c).Grade, for all students s and courses c, can be constructed. In our example, each attribute Takes(s, c).Grade will have two parents: s.IQ and c.Diff. Consequently, the local distribution class need only be a single local probability distribution. We discuss more complex situations in Heckerman et al. (2004).

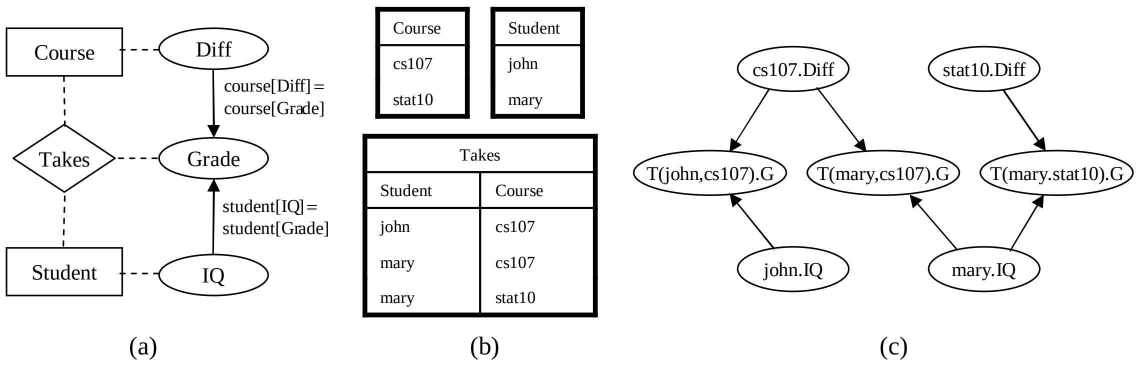
Whereas most of this paper concentrates issues of representation, the problems of probabilistic inference, learning local distributions, and learning model structure are also of interest. For all of these problems, it is natural to extend the concept of an instance to that of a *partial instance*; an instance in which some of the attributes do not have values. A simple approach for performing probabilistic inference about attributes in a DAPER model given a partial instance is to (1) explicitly construct a ground graph, (2) instantiate known attributes from the partial instance, and (3) apply standard probabilistic inference techniques to the ground graph to compute the quantities of interest. One can improve upon this simple approach by utilizing the additional structure provided by a relational model—for example, by caching inferences in subnetworks. Koller and Pfeffer (1997), for example, have done preliminary work in this direction. With regards to learning, note that from a Bayesian perspective, both learning about both the local distributions and model structure can be viewed as probabilistic inference about (missing) attributes (e.g., parameters) from a partial instance. In addition, there has been substantial research on learning PRMs (e.g., Getoor et al., 2002) and much of this work is applicable to DAPER models.

4. Plates Models and PRMs

Plate models were developed independently by Buntine (1994) and the BUGS team (e.g., Spiegelhalter 1998) as a language for compactly representing graphical models in which there are repeated measurements. We know of no formal definition of a plate model, and so we provide one here. This definition deviates slightly from published examples of plate models, but it enhances the expressivity of such models while retaining their essence (see Heckerman et al., 2004).

According to our definition, plate and DAPER models are equivalent. The invertible mapping from a DAPER to plate model is as follows. Each entity class in a DAPER model is drawn as a large rectangle—called a *plate*. The plate is labeled with the entity-class name. Plates are allowed to intersect or overlap. A relationship class for a set of entity classes is drawn at the named intersection of the plates corresponding to those entities. If there is more than one relationship class among the same set of entity classes, the plates are drawn such that there is a distinct intersection for each of the relationship classes. Attribute classes of an entity class are drawn as ovals inside the rectangle corresponding to the entity but outside any intersection. Attribute classes associated with a relationship class are drawn in the intersection corresponding to the relationship class. Arc classes and constraints are drawn just as they are in DAPER models. In additon, local distribution classes are specified just as they are in DAPER models.

The plate model corresponding to the DAPER model in Figure 2a is shown in Figure 3a. The two rectangles are the plates corresponding to the Student and Course entity classes. The single relationship class between Student and Course—Takes—is repre-



Course
cs107
stat10

Student
john
mary

Takes

Student	Course
john	cs107
mary	cs107
mary	stat10

Figure 2. (a) A directed acyclic probabilistic entity-relationship (DAPER) model showing that a student's grade in a course depends on both the student's IQ and the difficulty of the course. (b) An example skeleton for the entity and relationship classes in the ER model (the same one shown in the previous figure). (c) The DAG model (structure) defined by the application of the DAPER model to the ER skeleton.

sented as the named intersection of the two plates. The attribute class Student.IQ is drawn inside the Student plate and outside the Course plate; the attribute class Course.Diff is drawn inside the Course plate and outside the Student plate; and the attribute class Takes.Grade is drawn in the intersection of the Student and Course plate. The arc classes and their constraints are identical to those in the DAPER model.

Probabilistic Relational Models (PRMs) were developed in (e.g.) Friedman et al. (1999) explicitly for the purpose of representing relational data. The PRM extends the *relational model*—another commonly used representation for the structure of a database—in much the same way as the PER model extends the ER model. In this paper, we shall define directed PRMs such that they are equivalent to DAPER models and, hence, plate models. This definition deviates from the one given by (e.g.) Friedman et al. (1999), but enhances the expressivity of the language as previously defined (see Heckerman et al., 2004).

The invertible mapping from a DAPER model to a directed PRM (by our definition) takes place in two stages. First, the ER-model component of the DAPER model is mapped to a relational model in a standard way (e.g., Ullman and Widom, 2002). In particular, both entity and relationship classes are represented as tables. Foreign keys—or what Getoor et al. 2002 call *reference slots*—are used in the relationship-class tables to enocde the entity-relationship connections in the ER model. Attribute classes for entity and relationship classes are represented as attributes or columns in the corresponding tables of the relational model. Second, the probabilistic components of the DAPER model are mapped to those of the directed PRM. In particular, arc classes and constraints are drawn just as they are in the DAPER model.

The directed PRM corresponding to the DAPER model in Figure 2a is shown in Figure 3b. (The local distribution for Takes.Grade is not shown.) The Student entity class and its attribute class Student.IQ appear in a table, as does the Course entity class and its attribute class Course.Diff. The Takes relationship and its attribute class Takes.Grade is shown as a table containing the foreign keys Student and Course. The arc classes and their constraints are drawn just as they are in the DAPER model.

5. Details

In this short discussion, we have omitted many of the technical details of DAPER models as well as important facets of modeling relational data including the use of restricted relationships, self relationships, and probabilistic relationships. In addition, we have not described several important classes of PER model that expand into graphical models other than traditional DAG models. These topics are covered in Heckerman et al. (2004).

References

[Besag, 1974] Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, B*, 36:192–236.

[Buntine, 1994] Buntine, W. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225.

[Friedman et al., 1999] Friedman, N., Getoor, L.,

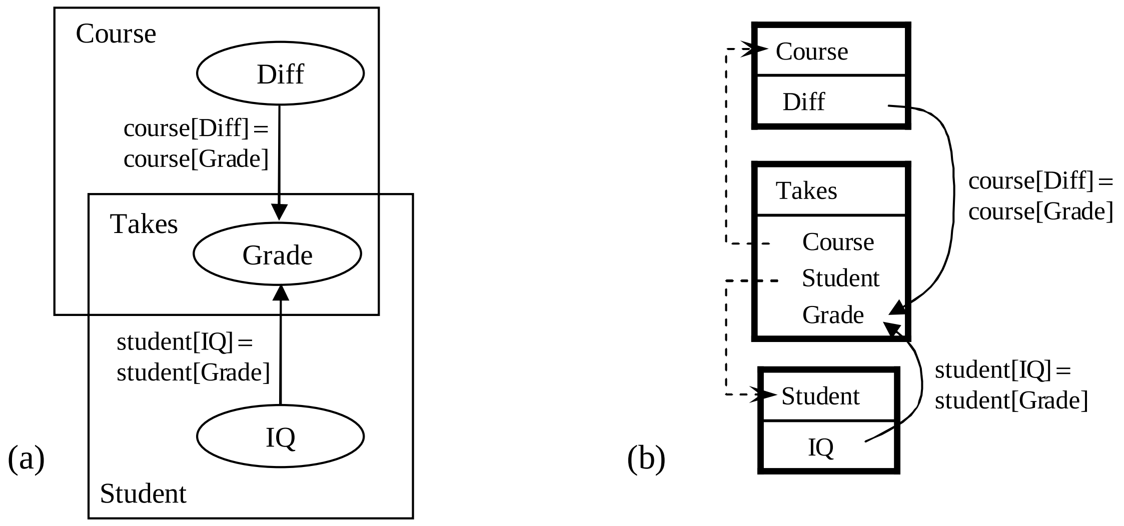

Course
Diff
course[Diff] = course[Grade]
Takes
Grade
student[IQ] = student[Grade]
IQ
(a)
Student
Course
Diff
course[Diff] = course[Grade]
Takes
Course
Student
Grade
student[IQ] = student[Grade]
Student
IQ
(b)

Figure 3. A plate model (a) and probabilistic relational model (b) corresponding the DAPER model in Figure 2a.

Koller, D., and Pfeffer, A. (1999). Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence,* Stockholm, Sweden, pages 1300–1309. International Joint Conference on Artificial Intelligence.

[Getoor et al., 2002] Getoor, L., Friedman, N., Koller, D., and Pfeffer, A. (2002). Learning probabilistic relational models of link structure. *Journal of Machine Learning Research*, 3:679–707.

[Good, 1965] Good, I. (1965). *The Estimation of Probabilities.* MIT Press, Cambridge, MA.

[Heckerman et al., 2004] Heckerman, D., Meek, C., and Koller, D. (2004). Probabilisitic models for relational data. Technical Report MSR-TR-2004-30, Microsoft Research, Redmond, WA. http://research.microsoft.com/research/pubs/view.aspx?tr_id=734.

[Koller and Pfeffer, 1997] Koller, D. and Pfeffer, A. (1997). Object-oriented Bayesian networks. In Geiger, D. and Shenoy, P., editors, *Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence,* Providence, RI, pages 302–313. Morgan Kaufmann, San Mateo, CA.

[Spiegelhalter, 1998] Spiegelhalter, D. (1998). Bayesian graphical modelling: A case-study in monitoring health outcomes. *Applied Statistics*, 47:115–134.

[Ullman and Widom,] Ullman, J. and Widom, J. *A First Course in Database Systems.* Prentice Hall, Upper Saddle River, NJ.

Relational Decision Networks

William H. Hsu
Dept. of Computing & Information Sciences
Kansas State University
bhsu@cis.ksu.edu

Roby Joehanes
Dept. of Computing & Information Sciences
Kansas State University
robbyjo@cis.ksu.edu

Abstract

Decision-theoretic intelligent agents must function under uncertainty and be able to reason and learn about objects and relations in the context of action and utility. This paper presents a new relational graphical model (RGM), analogous to the probabilistic relational model (PRM), for representation of decisions under uncertainty. It first analyzes some basic properties of the representation and gives an adaptation of several decision network inference algorithms to these relational decision networks. It then describes some early experimentation with algorithms learning link structure in PRMs, discussing how these can be adapted to learning in decision networks. Finally, it considers the problem of representing dynamic relations in decision networks and sketches an extension of the dynamic PRM representation to include choice and utility.

1 INTRODUCTION

Uncertainty is a common feature of decision problems for which the *decision network* or *influence diagram* is currently one of the most widely-used graphical models. Decision networks represent the state of the world as a set of variables, and model probabilistic dependencies, action, and utility. Though they provide a synthesis of probability and utility theory, decision networks are still unable to compactly represent many real-world domains, a limitation shared by other propositional graphical models such as flat Bayesian ntworks and dynamic Bayesian networks. Decision domains can contain multiple objects and classes of objects, as well as multiple kinds of relations among them. Meanwhile, objects, relations, choices, and valuations can change over time. For example, a supply chain consists of multiple raw materials, components of manufactured goods, multiple goods, and a market that may consist of multiple redistributors and buyers.

Capturing such a domain in a decision network would require not only an exhaustive representing of all possible objects and relations among them, but also a combinatorially fast-growing space of choices and valuations. [JT99] This raises two problems. The first one is that the inference using such a dynamic decision network would likely exhibit near-pathological complexity, making the computational cost prohibitive. The second is that reducing the rich structure of domains such as supply-chain management and enterprise resource planning (ERP) to very large, "flat" decision network would make it much more difficult for human beings to comprehend. This paper addresses these two problems by introducing an extension of decision networks that captures the relational structure of some decision domains, and by adapting methods for efficient inference in this representation.

First-order formalisms that can represent objects and relations, as opposed to just variables have a long history in AI. Recently, significant progress has been made in combining them with a principled treatment of uncertainty. In particular, probablistic relational models, or PRMs, are an extension of Bayesian networks that allows reasoning with classes, objects, and relations. [FGKP99] The representation we introduce in this paper extends PRMs to decision problems in the same way that the decision networks extend Bayesian networks. We therefore call it the *relational decision network* or RDN. We develop two inference procedures for RDNs: the first based upon the traditional variable elimination algorithm developed by Shenoy [She92] and Cowell [Cow94], the second a more efficient one based upon an adaptive importance sampling-based algorithm [CD00, HGJ$^+$03].

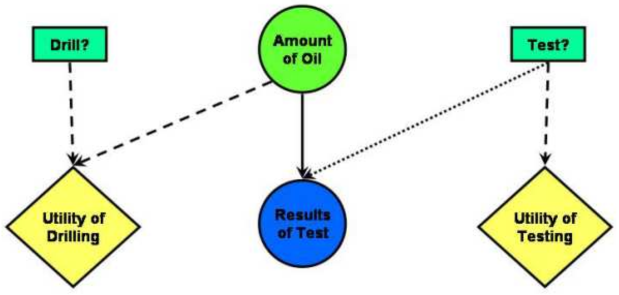


Figure 1: Decision network for the Oil Wildcatter problem. [She92]

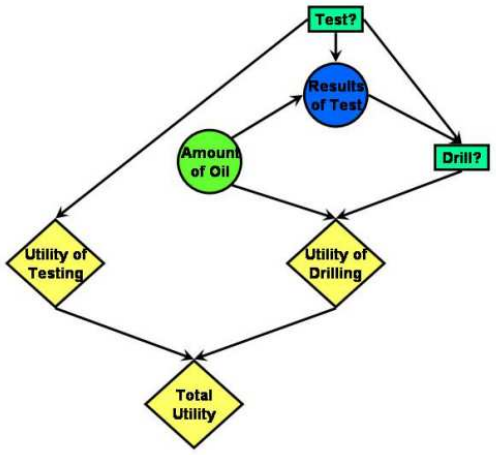


Figure 2: Influence diagram for the Oil Wildcatter problem. [She92]

2 DECISION NETWORKS AND INFLUENCE DIAGRAMS

Decision networks, introduced by Howard and Matheson [HM81], contain three kinds of nodes: *chance* (or *uncertainty*) nodes representing random variables as in a Bayesian network; *choice* (or *decision*) nodes) representing decisions to be made; and one or more *utility* nodes, each denoting a random variable ranging over utilities of the outcomes (as an aggregate of cost and benefit). These are depicted using circles, rectangles, and diamonds, respectively. [RN03, Nea04]

In a decision network, the edges are interpreted as follows [Nea04]:

1. Edges into chance nodes denote conditioning: The values of chance nodes are probabilistically dependent on the values of parents.
2. Edges into choice nodes denote sequence: The values of parents are known at the time the decision is made.
3. Edges into utility nodes denote deterministic functions of the values of parents.

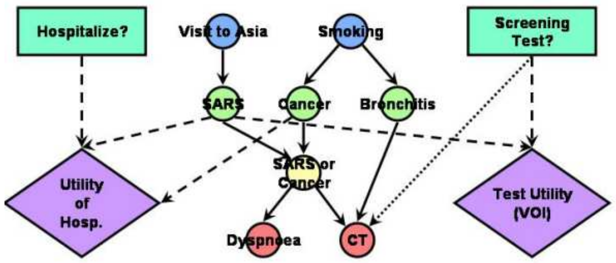


Figure 3: The DEC-Asia decision network.

Figure 1 depicts the decision network for the *Oil Wildcatter* problem. [She92, CDLS99] This graphical model represents the joint distribution and utility of a decision problem with boolean choice nodes and ternary chance nodes ($\mathsf{Amount} \in \mathsf{Dry}, \mathsf{Wet}, \mathsf{Soaking}$; $\mathsf{Result} \in \mathsf{Poor}, \mathsf{Intermediate}, \mathsf{Good}$).

Figure 2 shows an equivalent influence diagram, with links between choice (decision) nodes. These graphical models specify a decision sequence: (Test, Result, Drill, Oil), corresponding to a decision tree of orderings over choice, chance, and outcome utility nodes.

3 PROBABILISTIC RELATIONAL MODELS

Probabilistic relational models (PRMs) extend the flat (propositional) representation of the variables and the conditional dependencies among them to an object-relational representation. Before proceeding to discussion the decision network analogues of PRMs, we briefly review the PRM family and the relevant components of a PRM specification.

A *relational schema* is a set of classes $\mathcal{C} = C_1, C_2, \ldots, C_k$, where every class C is associated with a ground set of *propositional attributes* $\mathcal{A}(C)$ and a set of *relational attributes*, also known as its *reference slots*,

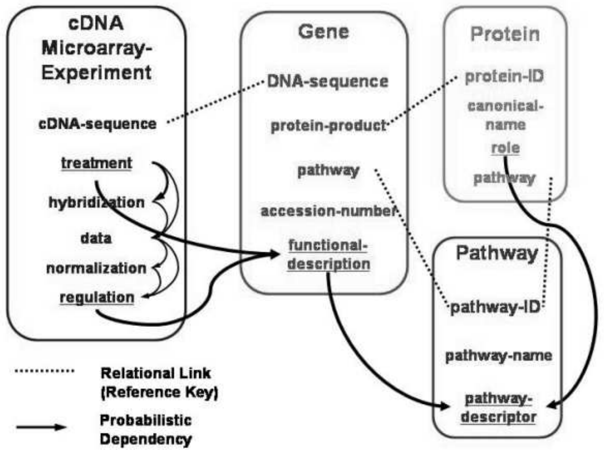


Figure 4: Probabilistic relational model.

$\mathcal{R}(C)$. Propositional attributes A are instance variables $C.A$ ranging over finite domains $V(C.A)$. Similarly, relational attributes $C.R$ each correspond to a set of member objects of a class and therefore range over the inclusion-exclusion (power set) $2^{C'}$ of some class C' from $\mathcal{C}$. That is, $C.R$ represents the set of members of the concept class C', so it is a selector function between C and $2^{C'}$. [SDW03]

As an example extending the DEC-Asia decision network above, the Patient schema might be used in to represent partial or total patient records, with classes corresponding to information about a patient's pulmonary medical history, smoking history, travel itinerary, and groups of people contacted. The propositional attributes of the medical history include the patient's age, previous contagious pulmonary diseases contracted, and currently extant diseases; the relational attributes might include the patient's membership in a list of quarantine subjects and *links* between patients denoting specific exposure incidents and contexts. Note that some of these are static and some, such as clusters of at-risk destinations and groups of people, may be dynamic relational attributes.

An *instantiation* of a schema is a set of objects, each belonging to some class $C \in \mathcal{C}$, where every object is specified using lists of its propositional and relational attributes and their nesting according to the (possibly hierarchical) schema. For example, an instantiation of the Patient schema might be a particular patient case history, with all available descriptive data and test results specified.

A *probabilistic relational model (PRM)* encodes a probability distribution over the set of possible instantiations I of some schema [FGKP99]. The *object skeleton* of an instantiation is the set of its member objects, abstracted over propositional attributes (i.e., with their values unspecified). The *relational skeleton* of an instantiation is the set of its member objects with all relational attributes specified for a **given** set of propositional attribute values. In the case of "known structure", the relational skeleton is provided as input and the PRM specifies a full conditional distribution for every attribute A of every class C using this skeleton. The underlying ground network consists of arcs into an attribute in C and parent sets consisting of attributes of C and other classes. The eligible classes that may contain parents of a node in C are those related by some *slot chain*, i.e., a composition of relational attributes. In order to implement this aggregation, a function mapping multiple attribute values into one is required. Aggregation functions could include descriptive statistics such as modes, median, and moments (mean and variance) of relational attributes. In the example from the previous section, the total or per capita average of diagnosed severe acute respiratory syndrome (SARS) cases in countries visited by a person is an example of an aggregate.

As a further example to illustrate slot chains, Figure 4 depicts a PRM for the domain of computational genomics, particular gene expression modeling from DNA hybridization microarrays. Slot chains can be traced using the reference keys (dotted lines). This PRM contains tables for individual microarrays or gene chips (admitting aggregation of chip objects into classes), putative gene function (where known or hypothesized), putative pathway membership (where known or hypothesized), and protein production (a target aspect of discovered function).

A PRM Π for a relational schema S is defined as follows. For every class C and every propositional attribute $A \in \mathcal{A}(C)$, we have:

1. A set of *parents* $Pa(C.A) = \{Pa_1, Pa_2, \ldots, Pa_i\}$, where each Pa_i has the form $C.B$ or $\gamma(C.\tau.B)$, where τ is a slot chain and $\gamma()$ is an aggregation function.

2. A *conditional probability function or table* for $P(C.A|Pa(C.A))$.

Let $\mathcal{O}$ be the set of objects in the relational skeleton of Π. The probability distribution over the instantiations I of S, over which the Π abstracts, is:

$$P(I) = \Pi_{obj \in \mathcal{O}} \Pi_{A \in \mathcal{A}(obj)} P(obj.A|Pa(obj.A))$$

This allows a PRM to be flattened into a large Bayesian network containing ground (propositional) chance nodes, with one variable for every attribute of every object in the relational skeleton of Π and belief functions (usually deterministic) for the aggregation operations. The latter are open-ended in form and omitted from the formula for brevity.

As Getoor *et al.* [GFKT02] and Sanghai *et al.* [SDW03] note, the most general case currently amenable to learning is where an object skeleton is provided and structure and parameter learning problems must be solved in order to specify a distribution over relational attributes. In the DEC-Asia domain, a PRM might specify a distribution over possible transmission vectors of a SARS-infected person (the itinerary, the locale of contamination, the set of persons contacted).

4 RELATIONAL DECISION NETWORKS

4.1 RDN Representation

We now extend decision networks to relational representations using a simple and straightforward synthesis of decision network and PRM specifications.

Definition 1 The relational decision schema S for a decision network B consists of three sets of classes $\mathcal{C_X} = C_{X_1}, C_{X_2}, \ldots, C_{X_n}$, $\mathcal{C_D} = C_{D_1}, C_{D_2}, \ldots, C_{D_m}$, and $\mathcal{C_O} = C_{O_1}, C_{O_2}, \ldots, C_{O_l}$, where every class C_X is associated with a ground set of *propositional attributes* $\mathcal{A}(C_X)$ and a set of *relational attributes* $\mathcal{R}(C_X)$. Propositional decision attributes A are instance variables $C.A$ ranging over the finite chance, decision, and utility domains $V_{\mathrm{X}(C.A)}$, $V_{\mathrm{D}(C.A)}$, $V_{\mathrm{O}(C.A)}$. Relational attributes $C.R$ each correspond to a set of member objects of a class and, for all chance, decision, and outcome (utility) nodes, respectively, range over the power sets:

1. **Objects**: $2^{C_{X'}}$ of some class $C_{X'}$ from $\mathcal{C_X}$
2. **Actions**: $2^{C_{D'}}$ of some class $C_{D'}$ from $\mathcal{C_D}$
3. **Outcomes**: $2^{C_{O'}}$ of some class $C_{O'}$ from $\mathcal{C_O}$

Thus the relational attributes $C.R$ can include distinguished member *action identifiers* and *outcome identifiers* specifying a representation for equivalence classes of decisions and outcomes. Note that the range of actions may be continuous (e.g., in intelligent control or continuous decision problems) and the range of *utilities* may also be continuous. C_D and C_O specify only membership in S.

Definition 2: A *relational decision network (RDN)* for a relational schema S is a PRM $M = (B, D, U)$ with distinguished decision and choice nodes, factored

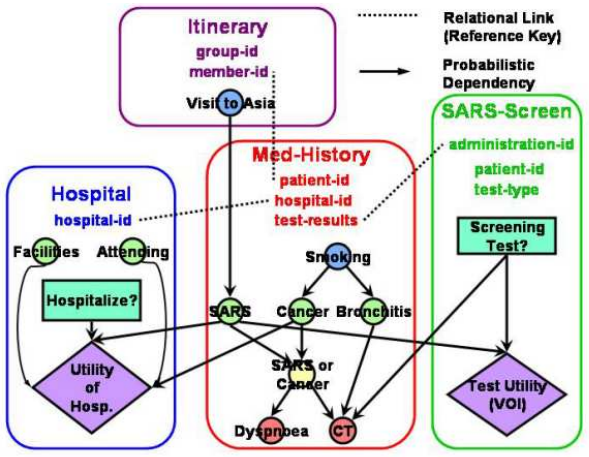


Figure 5: Relational decision network for the DEC-Asia domain.

according to S, where B encodes all conditional probability functions among chance nodes and their parents (choice and chance nodes), D encodes all sequential decision functions, and U encodes all conditional outcome and utility functions.

When the decision network's object skeleton is not known (i.e., the set of decisions and outcomes is not fully pre-specified), the RDN includes a boolean *existence variable* for propositional attributes of C_X, C_D, and C_O, and a boolean *reference slot variable* for relational attributes of C_X, C_D, and C_O.

Figure 5 shows a DRN for the DEC-Asia domain.

4.2 RDN Inference: Sampling-Based Algorithms

Our DBN algorithms include two sampling algorithms: Likelihood Weighting and Adaptive Importance Sampling (AIS). [Guo03] For brevity, we refer interested readers to Cheng and Druzdzel (2000) [CD00] for detailed descriptions of these algorithms.

A desired joint probability distribution function P(X) can be computed using the chain rule for Bayesian networks, given above. The most probable explanation (MPE) is a truth assignment, or more generally, value assignment, to a query Q = X E with maximal posterior probability given evidence e. Finding the MPE directly using Equation (1) requires enumeration of exponentially many explanations. Instead, a family of exact inference algorithms known as clique-tree propagation (also called join tree or junction tree propagation) is typically used in probabilistic reasoning applications. The first MPE (*belief revision*) algorithm for DAG models was developed by Pearl [Pea88]. The first general maximum *a posteriori* (MAP or *belief updat-*

ing) algorithms were developed by Pearl [Pea88] and independently by Shachter [Sha86]. The most popular extant implementation of belief updating is the junction tree algorithm of Lauritzen and Spiegelhalter [CDLS99]. Although exact inference is important in that it provides the only completely accurate baseline for the fitness function f, the problem for general BNs is $\#\mathcal{P}$-complete (thus, deciding whether a particular truth instantiation is the MPE is NP-complete) [Co90, Wi02]. [SJJ96] Approximate inference refers to approximation of the posterior probabilities given evidence. One stochastic approximation method called importance sampling [CD00] estimates the evidence marginal by sampling query node instantiations.

5 DYNAMIC EXTENSIONS

5.1 Dynamic Bayesian Networks

Dynamic Bayesian networks (DBNs) extend flat Bayesian networks to model problems with a temporal component. In a decision network, the state at time t is represented using a set of random variables $X_t = \{X_{1,t}, X_{2,t}, \ldots, X_{n,t}\}$ The state at time t is dependent on thoes at previous time steps. Each state in the system is assumed to depend only on the immediately preceding state (i.e., the system observes the Markov property in the first degree). The representation must therefore capture the transition distribution $P(X_{t+1}|X_t)$. This can be done using a two-time-slice Bayesian network fragment (2TBN) B_{t+1}, containing variables from X_{t+1} whose parents are variables from X_t or X_{t+1}, and variables from X_t without any parents. The process is also usually assumed to be stationary, so that the transition equations for all time slices are identical: $B_1 = B_2 = \ldots = B_t = B_{\rightarrow}$. Thus a DBN network is fully specified using a pair of Bayesian networks $(B_0, B_{\rightarrow}$, where B_0 represents the initial distribution $P(X_0)$ and $B_{\rightarrow}$ is a two-time slice Bayesian network, which as discussed above defines the transition distribution $P(X_{t+1}|X_t)$.

5.2 Dynamic Probabilistic Relational Models

Sanghai *et al.* recently developed a hybrid of PRMs and DBNs for decision-theoretic troubleshooting, which are called *dynamic probabilistic relational models (DPRMs)*. The key innovation of this related work is that it is the first representation to support relational aggregates in a temporal model. This is achieved by extending the 2TBN representation to a 2TPRM where each time slice contains a PRM. This extension is straightforward, with the slight complication that in flattening (or "unrolling") a PRM into a ground representation.

DRNs admit straightforward extension to dynamic domains using the 2 time slice dynamic PRM representation presented by Sanghai et al..

6 CONCLUSIONS AND FUTURE WORK

We have described a new representation for decision networks that combines the compact abstraction of PRMs with utility theoretic graphical model. We have considered several continuations of this research, grouped into four categories: applications, scalability, comparison to other decision models, and improvements to the ordering algorithm. DRNs are currently being implemented for use in Bayesian Network tools in Java (BNJ).

Acknowledgements

This work was partially supported by the National Science Foundation under a cooperative agreement.

References

[CD00] J. Cheng and M. J. Druzdzel. AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *Journal of Artificial Intelligence Research*, 13:155–188, 2000.

[CDLS99] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer, New York, 1999.

[Cow94] Robert G. Cowell. Decision networks: a new formulation for multistage decision problems. Technical Report 132, Department of Statstical Science, University College London, 1994.

[FGKP99] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-1999)*, pages 1300–1309, 1999.

[GFKT02] Lise Getoor, Nir Friedman, Daphne Koller, and Ben Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3(12):679–707, 2002.

[Guo03] H. Guo. *Algorithm Selection for Sorting and Probabilistic Inference: A Ma-*

chine Learning-based Approach. PhD thesis, Kansas State University, 2003.

[HGJ$^+$03] William H. Hsu, Haipeng Guo, Roby Joehanes, Benjamin B. Perry, and Julie A. Thornton. Bayesian network tools in java (bnj) v2. Distributed from URL: http://bndev.sourceforge.net, 2003.

[HM81] R.A. Howard and J.E. Matheson. Influence diagrams. In R.A. Howard and J.E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, volume II, pages 721–762. Strategic Decisions Group, Menlo Park, CA, 1981.

[JT99] E. Jorgensen and N. Toft. A Bayesian network based monitoring system for sow management. Technical Report 80, 1999.

[Nea04] R. E. Neapolitan. *Learning Bayesian Networks.* Prentice Hall, New York, 2004.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, NJ, 2 edition, 2003.

[SDW03] Sumit Sanghai, Pedro Domingos, and Daniel Weld. Dynamic probabilistic relational models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-2003)*, pages 302–313, 2003.

[Sha86] R. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[She92] P. Shenoy. Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40(3):463–484, 1992.

[SJJ96] L. K. Saul, T. Jaakkola, and M. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

BLOG: Relational Modeling with Unknown Objects

Brian Milch MILCH@CS.BERKELEY.EDU
Bhaskara Marthi MARTHI@CS.BERKELEY.EDU
Stuart Russell RUSSELL@CS.BERKELEY.EDU
Computer Science Division, University of California, Berkeley, CA 94720-1776 USA

Abstract

In many real-world probabilistic reasoning problems, one of the questions we want to answer is: how many objects are out there? Examples of such problems range from multi-target tracking to extracting information from text documents. However, most probabilistic modeling formalisms — even first-order ones — assume a fixed, known set of objects. We introduce a language called BLOG for specifying probability distributions over relational structures that include varying sets of objects. In this paper we present BLOG informally, by means of example models for multi-target tracking and citation matching. We discuss some attractive features of BLOG models and some avenues of future work.

1. Introduction

In many probabilistic reasoning problems, from multi-target tracking to extracting information from text documents, our task is to infer facts about the real-world objects that generated our data. The set of real-world objects involved is seldom known in advance. Thus, we need a probabilistic modeling formalism that allows for uncertainty about what objects exist.

Existing formalisms that combine probability with logic programming (Kersting & De Raedt, 2001; Sato & Kameya, 2001) make both the *unique names assumption* — that each term in the logical language refers to a distinct object — and the *domain closure assumption* — that the only objects are those denoted by the terms of the language. Thus, these formalisms only allow a fixed, known set of objects. Probabilistic relational models (PRMs) have been extended in several ways to allow unknown objects. PRMs may include *number variables* which specify the number of objects that stand in a given relation to an existing object (Koller & Pfeffer, 1998). A PRM may also include *existence variables*, which specify, for instance, whether a role exists for a given actor in a given movie (Getoor et al., 2002). Finally, PRMs have been extended to allow uncertainty about the total number of objects of a given type (Pasula et al., 2003). But there has been no unified syntax for describing all these kinds of uncertainty.

In this paper, we present a language called BLOG (Bayesian Logic) which is designed for reasoning about unknown objects. A BLOG model defines a probability distribution over model structures of a first-order logical language, which may include varying sets of objects. In (Milch et al., 2004), we discuss BLOG formally and prove that if a BLOG model satisfies certain acyclicity conditions, it defines a unique probability distribution. In this paper, we take a more informal approach, introducing the language by example. We also discuss how to assert evidence about unknown objects, and highlight some attractive features of BLOG models for information extraction tasks.

2. The aircraft domain

In this section, we describe one domain that we will use as a running example. Consider the problem of tracking an unknown number of aircraft, over an area that contains an unknown number of air bases. At each time step, an aircraft is either on the ground at some base, or flying with some position and velocity. Aircraft that are on the ground have some probability of beginning a flight to some destination at each step.

Suppose we observe the world through radar. For each time step t we receive a set of blips, each of which has an (x, y, z) location. A blip either is generated by some aircraft, in which case the location depends noisily on the aircraft's position, or is a false detection (resulting from clouds, etc.) We do not observe the true set of aircraft or airbases that exist, nor the identity of the aircraft generating a particular blip.

Here are some questions that we might be interested

Table 1. Language $\mathcal{L}_{\text{air}}$ for the aircraft domain.

Functor symbol	Type signature	Return type
Location	(AirBase)	R2Vector
HomeBase	(Aircraft)	AirBase
CurBase	(Aircraft, NaturalNum)	AirBase
State	(Aircraft, NaturalNum)	R6Vector
Dest	(Aircraft, NaturalNum)	AirBase
TakesOff	(Aircraft, NaturalNum)	Boolean
Lands	(Aircraft, NaturalNum)	Boolean
BlipTime	(RadarBlip)	NaturalNum
BlipSource	(RadarBlip)	Aircraft
ApparentPos	(RadarBlip)	R3Vector

in, given a set of observations:

- Did the blip at position $(3.3, 6.2, 9)$ at time 5 come from an aircraft, and if so, what is its destination?
- How many airbases exist?
- Are the blip at $(2, 4, 1)$ at time 3 and the blip at $(6, -1, 4)$ at time 9 from the same source?

3. Possible worlds

Our modeling approach is to specify a probability distribution over a set of possible worlds. A possible world for the aircraft domain consists of:

- a set of air bases, each with a location in $\mathbb{R}^2$;
- a set of aircraft, each with a home base;
- a function that maps each (a, t) pair to the air base where aircraft a is located at time t, or a null value if a is in the air at time t;
- a function that maps each (a, t) pair to aircraft a's state vector (position and velocity) in $\mathbb{R}^6$ at time t;
- a function that maps each (a, t) pair to the base that is aircraft a's current destination at time t;
- for each time $t \in \{1, \ldots, T\}$, a set of radar blips observed at that time, each with an apparent source position;
- a function that maps each radar blip to its source aircraft, or a null value if it is a false detection.

We use first-order logic to describe such sets of possible worlds in a formal way. A typed first-order language (Enderton, 2001) $\mathcal{L}$ consists of a set of type symbols and a set of functor symbols (i.e., function and predicate symbols). Each functor symbol has a *type signature* $(s_1, \ldots, s_k)$, where each s_i is a type symbol. Such a functor is known as a k-ary functor symbol; if $k = 0$, it is also known as a constant symbol. Each functor symbol also has a *return type* s, which is Boolean for predicate symbols. We also allow functors to take on the special value null. Table 1 shows the functors for the aircraft domain.

As *model structure* ω of a typed first-order language includes an *extension* $[s]^\omega$ for each type, which is the set of objects of that type in ω. Also, for each functor symbol f with signature $(s_1, \ldots, s_k)$ and return type s, it includes an *interpretation* $[f]^\omega$, which is a function from $[s_1]^\omega \times \cdots \times [s_k]^\omega$ to $[s]^\omega \cup \{\mathsf{null}\}$. We will use model structures of a language to represent possible worlds formally.[1] For example, model structures of $\mathcal{L}_{\text{air}}$ correspond to possible worlds in the aircraft domain. The syntax and semantics of terms and formulas of a typed first-order language are as in standard first-order logic, except that terms have types and the arguments to a functor must be of the appropriate types.

4. The probabilistic model

To describe a probability distribution over possible worlds, it is often useful to imagine a *generative process* that samples a possible world. For the aircraft domain, we generate a possible world ω as follows. First, create some number (chosen according to a prior) of AirBase objects, and sample a location for each one. For each base b, create some number of aircraft having b as home base. For each time step t (starting at 0), and each aircraft a, sample values for a's attributes at time t. Specifically, if a is in flight at time $t-1$, decide whether a lands at time t. If so, the base that used to be a's destination becomes its current base. Otherwise, sample a's state vector at time t conditioning on its state at time $t-1$ and the location of a's destination. If a is on the ground at time $t-1$, decide whether it takes off at time t, and if so, sample a new destination and initial state for a. Finally, for each a in flight at time t, choose whether to create a radar blip corresponding to a. Also, create some number of false detections at time t.

This generative process induces a probability distribution over possible worlds. We now present BLOG, a formal language for specifying such distributions. A BLOG model begins by specifying a typed first-order language $\mathcal{L}$. A type can either be user-defined or be chosen from a library of standard types such as NaturalNum. The functor symbols of $\mathcal{L}$ are divided into two sets : *random* and *nonrandom*. The model then specifies, for each type s, a set of *guaranteed objects* $G_{\mathcal{M}}(s)$ which exist in every possible world, and the value of each nonrandom functor f on each tuple of guaranteed objects. In the aircraft domain, for ex-

[1]Strictly speaking, we only allow structures over a particular universe of discourse. See (Milch et al., 2004) for details.

```
$\#\{\mathsf{AirBase}\}$:
    $\sim$ NumBasesDistrib()
$\mathsf{Location}(b)$:
    $\sim$ UniformLocation()
$\#\{\mathsf{Aircraft} : \mathsf{HomeBase} \mapsto b\}$:
    $\sim$ NumAircraftDistrib()
$\mathsf{TakesOff}(a, t)$:
    if $\mathsf{Greater}(t, 0) \wedge \neg\mathsf{InFlight}(a, \mathsf{Pred}(t))$
        then $\sim$ TakeoffBernoulli()
$\mathsf{Lands}(a, t)$:
    if $\mathsf{Greater}(t, 0) \wedge \mathsf{InFlight}(a, \mathsf{Pred}(t))$
        then $\sim$ LandingDistrib($\mathsf{State}(a, \mathsf{Pred}(t))$,
                                              $\mathsf{Location}(\mathsf{Dest}(a, \mathsf{Pred}(t)))$)
$\mathsf{CurBase}(a, t)$:
    if $t = 0$ then $= \mathsf{HomeBase}(a)$
    elseif $\mathsf{TakesOff}(a, t)$ then $=$ null
    elseif $\mathsf{Lands}(a, t)$ then $= \mathsf{Dest}(a, \mathsf{Pred}(t))$
    else $= \mathsf{CurBase}(a, \mathsf{Pred}(t))$
$\mathsf{InFlight}(a, t)$:
    $= (\mathsf{CurBase}(a, t) = \mathsf{null})$
$\mathsf{State}(a, t)$:
    if $\mathsf{TakesOff}(a, t)$
        then $\sim$ InitState($\mathsf{Location}(\mathsf{CurBase}(a, \mathsf{Pred}(t)))$)
    elseif $\mathsf{InFlight}(a, t)$
        then $\sim$ StateTransition($\mathsf{State}(a, \mathsf{Pred}(t))$,
                                              $\mathsf{Location}(\mathsf{Dest}(a, t))$)
$\mathsf{Dest}(a, t)$:
    if $\mathsf{TakesOff}(a, t)$
        then $\sim$ UniformChoice($\{\mathsf{AirBase}\ b\}$)
    elseif $\mathsf{InFlight}(a, t)$
        then $= \mathsf{Dest}(a, \mathsf{Pred}(t))$
$\#\{\mathsf{RadarBlip} : \mathsf{BlipSource} \mapsto a, \mathsf{BlipTime} \mapsto t\}$:
    if $\mathsf{InFlight}(a, t)$ then
        $\sim$ NumDetectionsDistrib()
$\#\{\mathsf{RadarBlip} : \mathsf{BlipSource} \mapsto \mathsf{null}, \mathsf{BlipTime} \mapsto t\}$:
    $\sim$ NumFalseAlarmsDistrib()
$\mathsf{ApparentPos}(r)$:
    if $\mathsf{BlipSource}(r) = \mathsf{null}$
        then $\sim$ FalseDetectionDistrib()
    else $\sim$ ObsDistrib($\mathsf{State}(\mathsf{BlipSource}(r), \mathsf{BlipTime}(r))$)
```

Figure 1. BLOG model for the aircraft domain.

ample, there are guaranteed objects for the natural numbers, and the nonrandom functor $\mathsf{Pred}(x)$ denotes the standard predecessor function.

The main part of the BLOG model consists of statements specifying conditional probability distributions. The generative process described above includes two types of sampling operations. The first type involves sampling the value of some functor applied to some objects. The second type involves creating a set of objects having a certain property (e.g. aircraft with a given home base), where the number of newly created objects is sampled from some distribution. BLOG has a statement type for each of these operations.

4.1. Dependency statements

A BLOG model includes a *dependency statement* for each random functor, that describes how to sample the value of that functor applied to each tuple of objects. Consider, for example, the dependency statement for State in Figure 1. Suppose we are in the process of sampling a world ω and are about to sample a value for $\mathsf{State}(a, t)$ for some particular objects a and t. We first check whether the condition after the if statement — $\mathsf{TakesOff}(a, t)$ — holds in ω (thus, our sampling process needs to have already chosen the value of $\mathsf{TakesOff}(a, t)$ at this point). Suppose it is false. We then check the condition $\mathsf{InFlight}(a, t)$ in ω. Suppose this condition does hold. We then compute $\mathsf{State}(a, \mathsf{Pred}(t))$ and $\mathsf{Location}(\mathsf{Dest}(a, t))$ and pass them to the *conditional probability distribution* (CPD) StateTransition, which we assume is defined elsewhere by the user using a language such as Java. This function samples a value for $\mathsf{State}(a, t)$ from the return type of State.

In general, the lefthand side of a dependency statement specifies the functor symbol being sampled, and provides variable names that will be used to refer to the arguments of the functor. The righthand side consists of an if-then-else statement. The sampling process checks the clauses of this if-then-else statement until it finds one that is true. It then instantiates the arguments for the CPD and calls it. The CPD is either defined by the user or part of a library of standard distributions, such as Gaussian. If none of the clauses of the if statement are satisfied, then the value is null by default.

In the dependency statement for State, the arguments to the CPD were just the values of certain functors. However, BLOG also allows passing a *set* of values into a CPD. There are two situations where this is necessary. First, we might want to select one element of the set. For example, the first clause for the Dest functor in Figure 1 results in a uniform choice over the set of all air bases. The second situation where we need to pass in a set is when the values have to be aggregated in some way by the CPD. For example, when reasoning about movies, the success of a movie might depend on the sum of the Fame variables of each actor in the movie.

4.2. Number statements

The probability distributions governing the number of objects satisfying a particular property are specified using *number statements*. Consider the third statement in Figure 1. The lefthand side indicates that this statement determines the number of aircraft having a particular home base b. The righthand side has the same form as in dependency statements, except that it must sample from a distribution over natural numbers.

There can in general be multiple number statements for each type. The lefthand side of each number statement contains a set of conditions, which form a *potential object pattern.* In the example, this set consisted of the single condition $\mathsf{HomeBase}(a) = b$. Our semantics require that each nonguaranteed object in a possible world satisfy exactly one potential object pattern, and so when a value n is sampled for a given number statement, exactly n new objects are created.

5. Evidence and Queries

A BLOG model specifies a probability distribution over possible worlds of a language $\mathcal{L}$. Therefore, we can in principle introduce evidence simply by conditioning on a first-order sentence of $\mathcal{L}$. However, this approach runs into problems. Suppose, for example, that we have observed a blip at time 8 at position $(9.6, 1.2, 32.8)$, and condition on the sentence $\exists r\,(\mathsf{BlipTime}(r) = 8 \wedge \mathsf{ApparentPos}(r) = (9.6, 1.2, 32.8))$. We now want to query the destination air base of the aircraft that generated this blip. Unfortunately, we can't do this, because we don't have a way of referring to the blip outside the existential quantifier.

This problem is handled in logical reasoning systems using *Skolem constants.* Instead of asserting an existentially quantified sentence, one extends the language to include a new constant symbol, known as a Skolem constant. In the example, we might introduce the new constant symbol $\mathsf{R1}$ and condition on the sentence $\mathsf{BlipTime}(\mathsf{R1}) = 8 \wedge \mathsf{ApparentPos}(\mathsf{R1}) = (9.6, 1.2, 32.8)$.

However, our observation process is often such that we observe *all* objects generated by certain instances of number statements. For example, at time t we observe all the radar blips generated by the two number statements for $\mathsf{RadarBlip}$ in Figure 1 for that value of t. The fact that our observations are exhaustive can significantly affect our beliefs. For example, the probability that there are 10 aircraft in flight might be high given that there is a blip on the screen, but low given that there is *only* one blip on the screen.

We therefore use a syntax that allows us to state this exhaustiveness property. Suppose that, in addition to the previously mentioned blip, we observed only one other blip at time 8, at $(2, 1.6, 3)$. We first write $\{b : \mathsf{BlipTime}(b) = 8\} = \{\mathsf{B1}, \mathsf{B2}\}$. This asserts that the constant symbols $\mathsf{B1}$ and $\mathsf{B2}$ refer to all the blips b such that $\mathsf{BlipTime}(b) = 8$. We also make a local unique-names assumption, that $\mathsf{B1}$ and $\mathsf{B2}$ refer to different objects. The precise probabilistic semantics of such evidence statements is given in (Milch et al., 2004).

We then assert our observations as evidence, with the statements $\mathsf{ApparentPosition}(\mathsf{B1}) = (9.6, 1.2, 32.8)$ and $\mathsf{ApparentPosition}(\mathsf{B2}) = (2, 1.6, 3)$. We may now use the symbols $\mathsf{B1}$ and $\mathsf{B2}$ in our queries. For example, we might ask about the posterior distribution of $\mathsf{Dest}(\mathsf{BlipSource}(\mathsf{B1}))$.

6. Using BLOG Models for Information Extraction

In (Marthi et al., 2003), we argued that information extraction (IE) – the task of inferring facts from text documents – is a promising application for first-order probabilistic models that allow unknown objects. In this section, we give a BLOG model for the citation matching domain discussed in (Pasula et al., 2003; Marthi et al., 2003), and highlight some attractive features of BLOG models for IE.

6.1. BLOG Model for Citation Matching

In the citation matching task, we are given some citations taken from the "works cited" lists of publications in a certain field. We wish to recover the true sets of researchers and publications in this field. For each publication, we want to infer the true title and author list; for each researcher, we want to infer a full name.

We use the following generative model for this domain. First, some number of researchers are generated, and a name is chosen for each one. Then a number of publications are generated, depending on the number of researchers (we do not assume the publications are generated by individual researchers). For each publication, we choose a number of authors, then choose the specific authors by sampling uniformly without replacement from the set of researchers. The title of the publication is chosen independently.

We choose not to model how the given list of citations is generated; instead, we just treat the citations as guaranteed objects. For each citation, the publication cited is chosen uniformly at random from the set of publications. The author names and title that appear in the citation are sampled according to some string corruption models (we assume the citation does not drop, add, or reorder authors). Finally, we construct the whole citation string, given the corrupted names and title.

A BLOG model for this domain is shown in Fig. 2. This model illustrates how BLOG allows us to define sets of objects that are passed as arguments to

```
$\#\{\mathsf{Researcher}\ r\}$:
    $\sim$ NumResearchersDistrib()
$\mathsf{Name}(r)$:
    $\sim$ NamePrior()
$\#\{\mathsf{Publication}\ p\}$:
    $\sim$ NumPubsDistrib($\{\mathsf{Researcher}\ r\}$)
$\mathsf{NumAuthors}(p)$:
    $\sim$ NumAuthorsDistrib()
$\mathsf{Author}(p, i)$:
    if $\mathsf{Less}(i, \mathsf{NumAuthors}(p))$
        then $\sim$ SampleUnused($\{\mathsf{Reseacher}\ r\}$,
                                      $\{\mathsf{Author}(p, j) : \mathsf{Less}(j, i)\}$)
$\mathsf{Title}(p)$:
    $\sim$ TitlePrior()
$\mathsf{PubCited}(c)$:
    $\sim$ UniformChoice($\{\mathsf{Publication}\ p\}$)
$\mathsf{NameAsCited}(c, i)$:
    if $\mathsf{Less}(i, \mathsf{NumAuthors}(\mathsf{PubCited}(c)))$
        then $\sim$ NameObs($\mathsf{Name}(\mathsf{Author}(\mathsf{PubCited}(c), i))$)
$\mathsf{TitleAsCited}(c)$:
    $\sim$ TitleObs($\mathsf{Title}(\mathsf{PubCited}(c))$)
$\mathsf{CitString}(c)$:
    $\sim$ CitDistrib($\{i, \mathsf{NameAsCited}(c, i)$ :
                          $\mathsf{Less}(i, \mathsf{NumAuthors}(\mathsf{PubCited}(c)))\}$,
                $\mathsf{TitleAsCited}(c)$)
```

Figure 2. Blog model for citation matching.

a CPD, such as $\{\mathsf{Author}(p, j) : \mathsf{Less}(j, i)\}$. We can even pass sets of pairs, such as $\{i, \mathsf{NameAsCited}(c, i) : \mathsf{Less}(i, \mathsf{NumAuthors}(\mathsf{PubCited}(c)))\}$. An interesting issue is how to represent arbitrary-length sequences, such as a publication's author list. The model in Fig. 2 illustrates one representation, where we have an infinite sequence of variables $\mathsf{Author}(p, i)$ for each publication p, but $\mathsf{Author}(p, i) = \mathsf{null}$ for $i \geq \mathsf{NumAuthors}(p)$.

6.2. Attractive features of BLOG models

We will now discuss several attractive properties that follow naturally from the Blog modeling approach, but are lacking in many other approaches to IE.

6.2.1. Reasoning about objects not mentioned in the text

In (Marthi et al., 2003), we discussed how a citation matching system should handle a query such as, "Did Mike Jordan have a paper in UAI '97?" if it has not seen any citations to such a paper. We considered a system that may have access to documents, such as Mike Jordan's home page, which it can identify as *exhaustive* lists of the papers with a certain property. If the system has seen Mike Jordan's home page and noted that it does not contain a UAI '97 paper, then the probability that such a paper exists is very low. On the other hand, if it has not seen any exhaustive lists, it should return a higher probability.

We are not aware of any existing IE systems that support reasoning about unmentioned objects. However, the ability to do such reasoning arises almost unavoidably in Blog models. Even in the simple model of Fig. 2, which does not include any notion of an exhaustive list, our possible worlds can include uncited publications. Since we assume that the target of each citation is chosen uniformly at random, we can make inferences about the number of uncited publications: for instance, if every cited publication has been cited at least 10 times, then the probability that there is also an uncited publication out there is low. Of course, reasoning about unmentioned object may entail extra complexity in inference. The point is that in a Blog model, a query about unmentioned objects has well-defined semantics.

6.2.2. Representing attributes once per object

A Blog model like the one in Fig. 2 distinguishes an object's true attributes (e.g., $\mathsf{Title}(p)$) from the attributes associated with individual mentions (e.g., $\mathsf{TitleAsCited}(c)$). This means that we can reconstruct an object's true attributes using clues from many separate mentions. There is also a more subtle advantage: when we compute the probability of a possible world, probabilities for object attributes are multiplied in only once per object, not once per mention.

This point is worth noting because defining object attributes on a per-mention basis might seem like a good way to avoid reasoning about unknown objects. Indeed, that is the approach taken in (McCallum & Wellner, 2003), which defines three unified probabilistic models for IE. Model 1 uses the same variables that we would use in a Blog model: a set of attribute variables for each object, plus a variable for each mention that specifies the object it refers to. Model 2, on the other hand, avoids any explicit representation of objects, and associates a set of object attribute variables with each mention. The object attribute variables for co-referring mentions are constrained to be equal.

To see how this simplifying step can be harmful, consider the task of inferring attributes for people mentioned in documents. For example, suppose we have a document where the first name "Dana" occurs many times, and the probability that all these occurrences of "Dana" refer to the same person is close to 1. Now suppose we want to infer Dana's sex. According to U.S. Census data, the probability that a person named "Dana" is female is around 0.75. So Model 2 would include a potential giving weight 0.75 to $\mathsf{Sex}(x) = \mathsf{female}$

and weight 0.25 to $\mathsf{Sex}(x) = \mathsf{male}$ when $\mathsf{FirstName}(x) =$ "Dana". But this potential would be multiplied into the joint probability once per mention. So with n mentions, the posterior probability that the occurrences of "Dana" refer to a female would be approximately $(0.75)^n/((0.75)^n + (0.25)^n)$, which approaches 1 as n increases. In a document with a large number of mentions, this duplication of potentials might outweigh contextual clues about Dana's sex (such as pronouns), leading to incorrect results.

6.2.3. Reasoning about coreference: Beyond pairwise compatibilities

In the Blog model of Fig. 2, we can ask for the posterior probability that three citations are coreferent — that is, they have the same PubCited value. This probability depends on the chance that a single publication, with some attributes, would yield the three observed citation strings. Most existing methods for coreference resolution do not attempt to approximate this probability. Instead, they use *pairwise* compatibility scores representing the probability that two mentions corefer. This simplification is made, for instance, in the transition from Model 2 to Model 3 in (McCallum & Wellner, 2003). Model 3 does not include any attribute variables; it just includes a Boolean coreference variable for each pair of mentions (with constraints to enforce transitivity of the coreference relation).

This simplification can lead to incorrect coreference resolution, even if we are not interested in the attribute values for their own sake. For example, suppose we are given a news article with mentions of "Stuart", "Jones", and "Alice", and some grammatical cues suggest these mentions are all coreferent. If we only look at pairwise compatibilities, we will not realize that this three-way coreference is very unlikely (even if we enforce transitivity). There could easily be a person named "Stuart Jones", "Alice Jones", or "Alice Stuart". But it is very unlikely that a person would be referred to sometimes as "Stuart", sometimes as "Jones", and sometimes as "Alice", all in the same news article. This problem arises because there is ambiguity about which attribute of a person (first or last name) is being specified by the mention "Stuart".

7. Future Work

The obvious question at this point is how to do inference (and parameter estimation, which requires inference if our data are only partially observed) in Blog models. There are really two questions here. The first is under what conditions the inference problem is even decidable, given that a Blog model may permit an unbounded number of objects. We hope to define a Markov chain Monte Carlo (MCMC) algorithm that is guaranteed to converge to the correct probability for a Blog query under certain broad conditions.

The other question is: when can we do approximate inference efficiently in practice? It is worth noting that approximate inference is NP-hard even for standard Bayesian networks (Dagum & Luby, 1993), but this has not prevented Bayesian networks from being a popular representational formalism. Researchers have developed a toolbox of approximate inference algorithms that are accurate and efficient on some practical problems. We plan to build upon this work to develop approximate inference algorithms for Blog models.

With this goal in mind, there are two major approaches to explore. The first is to place some upper bound on the number of potential objects that exist, so we can represent the distribution over possible worlds with a large but finite Bayesian network. For some queries, we may not even need to impose an upper bound, because only a finite number of objects may be relevant. We can then apply an approximate inference algorithm such as loopy belief propagation (Murphy et al., 1999).

The other approach is to use a stochastic sampling technique such as MCMC, where we allow different states of our Markov chain to include different numbers of objects. MCMC algorithms of this type have been implemented for Bayesian mixture modeling (Neal, 2000) and for the citation matching task described in Sec. 6.1 (Pasula et al., 2003). However, we would like a general algorithm that applies to any probability distribution defined as a Blog model. In some of the MCMC algorithms mentioned above, the states of the Markov chain are not fully specified possible worlds, but rather partial descriptions that leave out (for example) the uncited publications. We believe that such query-specific simplifications of the MCMC state space can be applied to Blog models in general, and may lay the foundation for practical approximate inference.

We are also interested in extending Blog to represent undirected and conditional models, such as those used in (McCallum & Wellner, 2003). The Blog models we have described in this paper can be thought of as extending probabilistic relational models (Friedman et al., 1999) to handle unknown objects; we should also be able to extend their undirected analogues, relational Markov networks (Taskar et al., 2002). The main technical problem is ensuring that the normalization constant in an undirected model remains finite, even when we have an unbounded number of objects and hence an infinite set of possible outcomes.

Acknowledgments

This work was supported by ONR MURI N00014-00-1-0637 and the DARPA CALO program. The authors are also supported by an NSF Graduate Research Fellowship (Milch), and an NSERC PGS-B fellowship (Marthi). We thank Mark Paskin for useful discussions.

References

Dagum, P., & Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, *60*, 141–153.

Enderton, H. B. (2001). *A mathematical introduction to logic*. Academic Press. 2nd edition.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *Proc. 16th IJCAI* (pp. 1300–1307).

Getoor, L., Friedman, N., Koller, D., & Taskar, B. (2002). Learning probabilistic models of link structure. *JMLR*, *3*, 679–707.

Kersting, K., & De Raedt, L. (2001). Adaptive Bayesian logic programs. *Proc. 11th Int'l Conf. on ILP*.

Koller, D., & Pfeffer, A. (1998). Probabilistic frame-based systems. *Proc. 15th AAAI* (pp. 580–587).

Marthi, B., Milch, B., & Russell, S. (2003). First-order probabilistic models for information extraction. *IJCAI Wksp on Learning Statistical Models from Relational Data*.

McCallum, A., & Wellner, B. (2003). Toward conditional models of identity uncertainty with application to proper noun coreference. *IJCAI Wksp on Information Integration on the Web*.

Milch, B., Marthi, B., & Russell, S. (2004). First-order probabilistic models with unknown objects. Unpublished manuscript.

Murphy, K. P., Weiss, Y., & Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. *Proc. 15th UAI* (pp. 467–475).

Neal, R. M. (2000). Markov chain sampling methods for Dirichlet process mixture models. *J. Computational and Graphical Statistics*, *9*, 249–265.

Pasula, H., Marthi, B., Milch, B., Russell, S., & Shpitser, I. (2003). Identity uncertainty and citation matching. In *NIPS 15*.

Sato, T., & Kameya, Y. (2001). Parameter learning of logic programs for symbolic-statistical modeling. *JAIR*, *15*, 391–454.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. *Proc. 18th UAI* (pp. 485–492).

Autocorrelation and Relational Learning: Challenges and Opportunities

Jennifer Nevillle JNEVILLE@CS.UMASS.EDU
Özgür Şimşek OZGUR@CS.UMASS.EDU
David Jensen JENSEN@CS.UMASS.EDU
University of Massachusetts, Amherst MA 01003-9264 USA

Abstract

Autocorrelation, a common characteristic of many datasets, refers to correlation between values of the same variable on related objects. It violates the critical assumption of instance independence that underlies most conventional models. In this paper, we provide an overview of research on autocorrelation in a number of fields with an emphasis on implications for relational learning, and outline a number of challenges and opportunities for model learning and inference.

1. Introduction

Autocorrelation refers to correlation between values of the same variable on related objects. More formally, it is defined with respect to a set of related instance pairs $(z_i, z_j) \in Z$ and a variable X defined on these instances, and is the correlation between the values of X on these instance pairs. Autocorrelation is a common characteristic of many datasets. For example, hyperlinked web pages are more likely to share the same topic than randomly selected pages (Taskar et al., 2002), and proteins located in the same place in a cell (e.g., mitochondria or cell wall) are more likely to share the same function (e.g., transcription or cell growth) than randomly selected proteins (Neville & Jensen, 2002).

The prevalence of autocorrelation is not unexpected—a number of widely occurring phenomena give rise to such dependencies. Temporal and spatial locality very often result in autocorrelated observations, due to temporal or spatial dependence of measurement errors, or the existence of a variable whose influence is correlated among instances that are located closely in time or space (Mirer, 1983; Anselin, 1998). Social phenomena such as social influence (Marsden & Friedkin, 1993), diffusion processes (Doreian, 1990), and the principle of homophily (McPherson et al., 2001) give rise to autocorrelated observations as well, through their influence on social interactions that govern the data generation process.

Presence of autocorrelation is a strong motivation for relational learning and inference. It is well known that in relational domains, joint inference over an entire dataset results in more accurate predictions than conditional inference over each instance independently (Macskassy & Provost, 2003; Chakrabarti et al., 1998; Taskar et al., 2002; Yang et al., 2002; Neville & Jensen, 2003). Recent work has shown that the improvement over conditional models increases with increased autocorrelation (Jensen et al., 2004)—autocorrelation allows inferences on one object to be useful for inferences on related objects.

The presence of autocorrelation, however, also presents additional challenges for learning. A major difficulty is that the assumption of independent data instances that underlie most conventional models is no longer valid. For instance, in models constructed from temporal and spatial datasets, autocorrelation has long been recognized as a source of increased bias and variance (Anselin, 1998). These problems are only more severe in relational data that do not exhibit the regularities of temporal and spatial datasets. For example, *linkage*—a measure of the number of related instances—can be far greater and can vary dramatically throughout the dataset, and it is known that linkage interacts with autocorrelation to increase variance and such variance can bias feature selection toward features with the least amount of evidence (Jensen & Neville, 2002).

Datasets exhibiting autocorrelation are common in many fields including sociology, economics, geography, and physics (Doreian, 1990). Social network analysis often examines networks of social interactions which exhibit homophily. For example, in elementary school friendship networks, same-gender ties are more likely than different-gender ties (Anderson et al., 1999). Economic analysis often examines datasets with repeated measures of the same variable over time, which typically exhibit temporal autocorrelation (e.g., stock prices). As a consequence, researchers in these fields have investigated the effects of autocorrelation in parameter estimation, hypothesis testing, and structure search.

A common finding in these disparate fields is that departures from independence cannot be ignored—they may cause unduly complex models, and biased, inconsistent, or inefficient estimators. One possible approach is to design new statistical procedures that are robust to autocorrelation. A second one is to model dependencies explicitly.

In this paper, we provide an overview of research on autocorrelation in these fields with an emphasis on implications for machine learning. The remainder of this paper is organized as follows: First, we provide an overview of work in temporal sequence analysis focusing on work in econometrics. This field has a long history of analyzing the effects of autocorrelation. We next discuss research in spatial statistics that extend one-dimensional temporal models to address the needs of higher-dimensional spatial data, and continue with work in social network analysis on general network data. We then briefly outline models in relational learning and discuss the utility and implications of work in related fields for relational learning models.

2. Temporal Sequential Models

Linear regression models are commonly employed in both natural and social sciences to model the dependence of a single response variable Y on a set of predictor variables $\mathbf{X} = \{X^1, \ldots, X^m\}$. The conventional linear regression model is specified as follows:

$$Y_i = \beta \mathbf{X_i} + \epsilon_i \tag{1}$$

where β is a vector of weights, ϵ is a normally-distributed error term with mean 0, and i is an index over data instances. The weight vector β is usually estimated using Ordinary Least Squares (OLS), which is known to be the Best Linear Unbiased Estimator (BLUE)—the minimum variance estimator for the class of linear unbiased estimators.

One of the implicit assumptions underlying these models is that instances are independent. However, this assumption is violated in many datasets consisting of observations over time. For example, the daily closing price of a stock market index (e.g., S&P500) can be represented as a time series. It is well known that stock prices exhibit autocorrelation over time—the best prediction of tomorrow's stock prices is based on today's prices (Wooldrige, 2003). [1]

If a conventional linear regression model is used to model autocorrelated data, the residuals of the model will be autocorrelated. This violates the modeling assumption of independent and identically distributed errors. For example, if equation 1 is used to regress a number of market indicators $\mathbf{X}$ (e.g. unemployment rate, federal interest rate) on the index price Y, errors will be similar for instances close in time due to the autocorrelation of Y. Serially correlated errors can be detected using a variety of statistics. The most widely-used is the Durbin-Watson statistic, which is a normalized sum of the squared differences of successive terms in a time series (Kennedy, 1998).

When the errors are autocorrelated, OLS estimators are unbiased, but they are no longer BLUE (Wooldrige, 2003). That is, there exist other unbiased linear estimators with lower variance. Not accounting for the autocorrelation structure results in larger sampling errors for the β estimates. Typically, this increased variance will bias hypothesis tests in the direction of increased Type I errors (rejecting the null hypothesis when it is true) and will result in incorrect conclusions of significance. Furthermore, the amount of bias will increase as the level of autocorrelation increases.

Autocorrelated errors typically arise in one of two situations. First, autocorrelated errors may be due to correlated measurement errors. For example, trading patterns can produce serially correlated estimates of stock returns even when there is no serial correlation for returns in general. Returns are measured using the price of the stock on the last trade in a given time period; if the measurement time period is short and the stock is sparsely traded, the estimates of return values will exhibit autocorrelation. Models that represent such autocorrelation dependencies among error terms are known generally as *disturbances models*, but are also referred to as heterogeneity models in spatial analysis, or as serial correlation models in temporal analysis. Second, autocorrelated errors may be due to correlation of the response values. For example, as was mentioned above, the price of an index today may

[1]Unfortunately, this characteristic cannot be used for accurate prediction because the chance of a stock's future price going up is the same as it going down. The overall process is is a random walk.

influence the price tomorrow. This case is typically modeled by including a lagged value of the response variable as a regressor. Models that represent these dependencies are known generally as *effects models*, but are also referred to as autoregressive models or as dependence models in spatial and social network analysis. Below we discuss each of these in turn.

2.1. Disturbances Model

Serial correlation implies that there is systematic dependence among the error terms of individual instances. The most common form is first-order serial correlation, in which the error term in one period includes a proportion of the error term in the previous period. The is commonly referred to as an AR(1) disturbance model:

$$Y_i = \beta \mathbf{X_i} + \mu_i, \text{ where } \mu_i = \rho\mu_{i-1} + \epsilon_i \tag{2}$$

where ρ is a parameter called the autocorrelation coefficient, whose absolute value is constrained to be less than 1. When $\rho = 0$, this model reduces to the standard linear model of equation 1.

When serial autocorrelation is present, analysts generally abandon OLS in favor of Generalized Least Squares (GLS) estimators that are BLUE. Unfortunately, knowledge of the correlation structure is needed for exact GLS estimates and in general this is not known apriori. Alternative Estimated Generalized Least Squares (EGLS) methods estimate ρ and β iteratively or jointly. EGLS estimators are neither linear nor unbiased but Monte Carlo studies have shown that EGLS is preferable to OLS in many situations (Kennedy, 1998). In particular, for the AR(1) disturbances model, EGLS is equal, or superior, to OLS when $\rho > 0.3$. The most frequently used EGLS methods are Cochrane-Orcutt iterative least squares, Durbin's two-stage method, Hildreth-Lu search procedure, and maximum likelihood. These four methods mainly differ in how they estimate ρ and are asymptotically equivalent if ϵ is distributed normally. Recent studies have shown that Bayesian estimation, which averages over a number of ρ estimates, is far superior to methods that use a single estimate (Kennedy, 1998).

2.2. Effects Model

Effects models take into account dependencies among the response values by including a lagged value of the response variable as one of the regressor variables. When lag equals 1 (e.g. first-order autocorrelation), the underlying model is referred to as an AR(1) effect model:

$$Y_i = \rho Y_{i-1} + \beta \mathbf{X_i} + \epsilon_i \tag{3}$$

When the underlying process is correctly modeled with equation 3, OLS estimators are biased but consistent as long as the errors are *contemporaneously* uncorrelated. This means that the n^{th} regressor term is not correlated with the n^{th} error term; it may be correlated with other error terms. In this case, analysts consider OLS to be the most appropriate estimator (Kennedy, 1998). In small samples, the OLS estimate for ρ is downward-biased, and the OLS estimate for β is upward-biased. In general however, there are no other estimators with superior small-sample properties so analysts prefer OLS for its asymptotic properties. Research has focused on obtaining unbiased OLS estimates for a range of specific autoregressive models, with recent work proposing a Monte Carlo based approach for models with non-normal error terms, higher-order autocorrelations, and exogenous variables (Tanizaki, 2000).

If, on the other hand, the errors are contemporaneously correlated, OLS estimators are biased and inconsistent. A two-step EGLS (as described in section 2.1) is not feasible in this situation because the residuals are correlated with the exogenous variables. The most common approach to take in this situation is instrumental variable (IV) estimation, which introduces extra *instument* variables to decouple the correlation between the regressors and the error terms to produce consistent estimators.

Autoregressive conditional heteroskedasticity (ARCH) models extend the basic AR models described above to model volatility clustering with non-constant variance that depends on past information (Engle, 1995). If the model does not include lagged-dependent variables, OLS estimators are BLUE, but non-linear maximum likelihood estimators are more efficient. If the model includes lagged-dependent variables then the OLS standard errors will not be consistent. In this case, EGLS estimators are asymptotically efficient and standard errors are asymptotically valid.

3. Spatial Models

Spatial datasets are analyzed in a number of fields including geography, biology, and economics. These datasets are typically represented in discrete or continuous two-dimensional space. For example, a spatial dataset may record soil properties throughout a spatial region. Equation 1 may also be used to model these data, for example to model the effects of soil properties on ground water contamination. In this case each vector index i indicates a point in space. We will focus on (simpler) models for discrete space where the data are represented as a lattice—each point in space corresponds to a node in the graph and is linked to a fixed number of other nodes that are closest with respect to a distance measure.

Tests for the presence of residual spatial autocorrelation are based on either OLS or ML estimates, including tests based on Moran's **I** statistic, and Wald, Likelihood Ratio, and Lagrange Multiplier tests (Anselin, 1998). If spatial data exhibit autocorrelation, the quality of OLS parameter estimates are affected in the same manner as was discussed for temporal data—OLS estimators are unbiased but they are no longer BLUE (Anselin, 1998). Again this results in biased hypothesis tests, with the amount of bias depending on the level of autocorrelation.

Dependencies among instances occur in the same manner as in the temporal model discussed above. Autocorrelated errors may be due to spatially correlated measurement errors. For example, a severe weather event may affect only part of the region, resulting in a cluster of correlated errors. On the other hand, autocorrelated errors may be due to spatial autocorrelation in the response variable itself—contamination levels are likely to correlated with the levels at nearby locations.

3.1. Disturbances Model

When the data exhibit autocorrelated disturbances, the error term of one instance influences the error terms of neighboring instances. A spatial disturbances model subsumes the first-order serial correlation model (equation 2) by allowing more general dependencies among the error terms:

$$Y_i = \beta \mathbf{X_i} + \mu_i, \text{ where } \mu_i = \rho \mathbf{W} \mu + \epsilon_i \tag{4}$$

Here $\mathbf{W}$ is an $n \times n$ weight matrix specifying the nature of dependencies among the disturbances, and ρ is the autocorrelation parameter. When $\rho = 0$ or W is uniformly 0, this model reduces to the standard linear model of equation 1. The matrix $\mathbf{W}$ is designed to represent the influence processes present in the network. Each entry w_{ij} denotes the influence node j has on node i. For example, in a first-order spatial disturbances model, row i has a value of 1 for each neighbor j of node i and all other entries are 0.

Spatial autocorrelation among error terms has been shown to affect the quality of OLS parameter estimates (Anselin, 1998). The effects are similar to those reported for temporal models—OLS estimators will be unbiased but inefficient and GLS estimators are BLUE but are of academic interest only because the correlation structure is generally unknown. Futhermore, the multidirectional nature of spatial dependencies limits the types of EGLS methods that will produce consistent estimates. Approaches based on ML or IV result in consistent estimates of ρ and therefore retain the asymptotic properties of consistency and efficiency. However, in small samples, OLS may sometimes perform equivalently, or better than EGLS, in terms of bias and mean squared error—though finite sample analysis is limited (Anselin, 1998).

3.2. Effects Model

The second type of dependency is again due to autocorrelation of the regressor values. The spatial effects model represents these dependencies with the following:

$$Y_i = \rho \mathbf{W} \mathbf{Y} + \beta \mathbf{X_i} + \epsilon_i \tag{5}$$

Again, when $\rho = 0$ or W is uniformly 0, this model reduces to the standard regression model (equation 1).

If the response variable is autocorrelated, OLS estimators will be biased, inconsistent, and inefficient regardless of the properties of the error term (Anselin, 1998). In temporal effects models (equation 3), the OLS estimates will be unbiased if the error terms show no serial correlation. The multidirectional nature of spatial dependencies however, introduces added complexity to the OLS estimates so the conditions for consistency are only met when autocorrelation is not present, when $\rho = 0$. This means that no consistent estimates can be obtained for OLS procedures, so spatial analogues of EGLS methods are not appropriate.

Maximum likelihood (ML) estimation does not suffer from the same effects that plague OLS estimation so it is the preferred method of estimation among analysts for both the disturbances and the effects model. ML estimators have attractive asymptotic properties—consistency, efficiency, normality—but are more complex and computationally intensive than OLS. We should also note here that the attractive asymptotic properties of ML estimation do not hold uniformly, but are valid under the following conditions: the existence of the log-likelihood function for the parameters, continuous differentiability of the log-likelihood function, boundedness of partial derivatives, positive definiteness and/or non-singularity of covariance matrices, and the finiteness of quadratic forms (Anselin, 1998). Typically, these conditions are satisfied if the spatial interaction structure ($\rho\mathbf{W}$) is non-explosive (i.e., the correlation between y_i and y_{i+d} goes to zero sufficiently "quickly" as $d \to \infty$, where d is graph distance).

Depending on the model form, ML estimation may involve a normalizing constant that is difficult to compute in closed form. For the models discussed above, this involves computing the log-determinant of an $n \times n$ matrix, which requires $O(n^3)$ operations for dense matrices. Research has focused on techniques to make ML estimation more tractable, including pseudolikelihood estimation (Besag, 1975), approximate ML estimation with Markov Chain Monte Carlo (MCMC) methods (Geyer & Thompson, 1992), and closed-form ML methods that avoid direct computation of the determinant (LeSage & Pace, 2001).

Hypothesis tests for ML estimates include the Wald test, the Likelihood Ratio test, and the Lagrange Multiplier test, all of which are based on the optimal asymptotic properties of the ML estimator. The tests are asymptotically equivalent but care must be taken when interpreting the tests on finite samples because some have higher Type I errors and others have higher Type II errors. The relative power of the tests for spatial data is yet to be investigated (Anselin, 1998).

4. Network Models

Spatial models have been applied extensively in the field of social network analysis where data consist of a network of interactions among entities (e.g., people, institutions). Social network datasets are represented as general graphs and differ from temporal and spatial data representations in that they are not restricted to a uniform structure. For example, to model the effects of socio-economic status on voting behavior in a community, income and status would be measured along with friendship ties to other members in the community. A set of nodes representing people and a set of edges representing their friendships forms the network graph. The graph structure varies as each person has a different number of friends. Again equation 1 may be used to model network data. In this case each vector index i indicates a node in the graph.

Spatial autocorrelation models are expressive enough to use as network autocorrelation models. Equation 4 represents a network disturbances model and equation 5 represents a network effects model (Marsden & Friedkin, 1993). In social network models, the weight matrix $\mathbf{W}$ specifies the social influence patterns present in the network and it can affect virtually all of the conclusions drawn from autocorrelation models (Leenders, 2002). Therefore, correct specification of $\mathbf{W}$ is crucial to the utility of the models. In practice, social network analysts do not estimate $\mathbf{W}$. Instead, they specify a $\mathbf{W}$ manually to model specific theories of social influence such as communication and comparison.

Social network models share the same challenges as spatial models—OLS parameter estimates of autocorrelated data will be inefficient and/or biased and inconsistent, and although ML estimates are more robust, they are computationally intensive (Doreian & K. Teuter, 1984). Simulation studies have demonstrated the superiority of ML estimates over a wide range of conditions (Doreian & K. Teuter, 1984). Although social network datasets are not restricted to a uniform structure, unfortunately there appears to be little work in social networks that examines the impact of varying graph structure on parameter estimation and hypothesis tests.

5. Models in Relational Learning

Datasets with more general dependencies than are seen in temporal, spatial, and social network data are commonplace in relational learning. For example, relational data for citation analysis can be represented as a typed, attributed graph, with nodes representing authors, papers and journals, and edges representing citation and published-in relationships. A model of paper *topic* may include attributes of related authors (e.g., speciality) and journals (e.g., prestige). However, an important characteristic of these data is that topic is autocorrelated—the topic of a paper is not independent of the topics of papers that it cites.

Relational data pose a number of additional challenges for model learning and inference. First, relational data often consider more than one type of entity in the same dataset (e.g., papers, authors and references). Second, relational data have complex dependencies, both as a result of direct relations (e.g., research paper references) and through chaining multiple relations together (e.g., papers published in the same journal). Third, relational data have varying structure (e.g., papers have different numbers of authors, references and citations).

Recent research in relational learning has produced several novel types of models to address these issues, including relational Markov network (RMNs) (Taskar et al., 2002), relational Bayesian networks (RBNs) (Friedman et al., 1999), and relational dependency networks (RDNs) (Neville & Jensen, 2004). These three models have the ability to represent and reason with autocorrelation; however, only RMNs and RDNs can reason with arbitrary forms of autocorrelation—RBNs can only reason with acyclic forms of autocorrelation, such as relationships that are structured by temporal constraints (Friedman et al., 1999).

There are two major findings that relate autocorrelation to learning and inference in relational models: that autocorrelation improves joint inference, and that autocorrelation may bias feature selection. We discuss each of these below.

First, several studies have shown that *joint inference* can significantly reduce classification error (Macskassy & Provost, 2003; Chakrabarti et al., 1998; Taskar et al., 2002; Yang et al., 2002; Neville & Jensen, 2003). Joint inference refers to procedures that make simultaneous statistical judgments about the same variables for a set of related data instances. By making inferences about multiple data instances simultaneously, joint inference can exploit autocorrelation in the data—judgments about one instance can be used to improve inferences about related instances. Recent work has shown that the improvement over conditional models, which make inferences in isolation, increases with increased autocorrelation, and in general, a joint inference procedure performs better when higher-order autocorrelation is present or when few labels are known with certainty (Jensen et al., 2004). In conditional models, the utility of modeling autocorrelation depends on whether the values of the autocorrelated attributes are known. Partially labeled datasets are common, but if the known labels do not exhibit autocorrelation, they cannot be used to seed the inferences. Related work shows that the relative advantage of a joint inference procedure over a conditional procedure reduces as the percentage of labeled data increases (Macskassy & Provost, 2003).

Second, recent research has shown that autocorrelation may bias feature selection (Jensen & Neville, 2002). Concentrated linkage and autocorrelation reduce the effective sample size of a data set, thus increasing the variance of parameter estimates (e.g., feature scores) estimated using that set. This reduction in effective sample size parallels the inefficiencies in temporal and spatial estimators. As a consequence, the probability of Type I errors is increased—features formed from objects with high linkage and autocorrelation may be selected as the best feature, even when the features are random. To our knowledge, few current relational learning algorithms adjust for the increased variance in estimation. Specifically, the current instantiation of RDNs use an underlying conditional model which adjusts for this bias, but the current instantiations of RBNs and RMNs do not. Inefficient parameter estimates will impact both selective (e.g., RBN, RDN) and non-selective (e.g., RMN) models. For both types of models, the increased variance may result in overfitting. In addition, the interpretation of feature weights/scores may be more difficult for non-selective models and structure learning may be biased in selective models.

6. Summary and Discussion

Autocorrelation effects have been studied extensively in other fields and it is clear that they cannot be ignored in relational learning. In particular, if the data exhibit autocorrelation, either autocorrelated measurement errors or an autocorrelated response variable, then conventional parameter estimates will be unbiased but will have increased variance. This has implications for (1) model performance, (2) feature rankings, and (3) feature selection. When the model is learned from "small" samples, the increased variance may lead to overfitting and result in lower performance. Although, we typically have "large" datasets in relational learning, as the level of autocorrelation increases so does the variance—the amount of data needed to offset the increased variance may be be larger than we expect. Increased variance will also impact feature rankings (by feature weights/scores), and consequently feature selection. Non-selective models often use feature weights for interpretation (e.g., to identify the most important features), and selective models use feature weights to learn the structure of the model. Both these endeavors will be adversely affected by the increased variance due to autocorrelation.

How can we adjust for autocorrelation in relational models? Below, we summarize past research and discuss options for model representation, learning and inference.

6.1. Representation

The first decision is how to include autocorrelation in the model representation—whether to model autocorrelation directly through variables or indirectly in the error term. This choice corresponds to selection of the effects model, the disturbances model, or some combination of the two, and may be based on the researcher's hypothesis about the dependencies present in the data. Explicit representation may result in more interpretable models, since the influence of an autocorrelated response variable is clear. However, implicit representation in the error term may be more broadly applicable. This approach could allow the use of existing models without a change of representation, but with only an adjustment for the effects of autocorrelation.

The second decision is how to encode the autocorrelation dependencies. This decision corresponds to the functional form of autocorrelation (e.g., first-order). For the spatial and network models discussed above, this refers to the specification of the weight matrix. For relational models, this usually refers to specification of autocorrelation features. For example, to predict the topic of a web page, we may include a feature that encodes the topics of other hyperlinked pages. While considerable attention has been paid to accurate parameter estimation in temporal and spatial autocorrelation models, it appears that researchers are less concerned with model/feature selection. However, one could imagine searching over a space of autocorrelation specifications to learn the correct structure.

6.2. Learning

The effect of autocorrelation on parameter estimation has been studied extensively. Below is a summary of the findings in temporal, spatial and network analysis:

1. If autocorrelation is ignored:
   - Parameter estimates are computationally efficient.
   - Parameter estimates are unbiased but have increased variance.
   - Hypothesis tests and confidence intervals may be biased.
2. If autocorrelation is modeled:
   - Parameter estimates are computationally complex, but more tractable approximate methods exist.
   - Parameter estimates are asymptotically optimal but may be biased in finite samples.
   - Finite sample comparison is limited. More complex estimation techniques may not always be justified.
   - Hypothesis tests are asymptotically unbiased but the relative power of various tests may vary on finite samples.

Some examples of model parameters in relational learning include clique potentials and feature weights. Results for temporal and spatial analysis indicate that there may be a tradeoff between computational efficiency and accurate parameter estimation. Understanding the effect of varying levels of autocorrelation on parameter estimation for finite samples is an important area for research for the relational learning community.

The effects on parameter estimation will also impact structure learning. Structure learning typically involves feature selection, which corresponds to either explicit or implicit hypothesis testing. It has been shown that autocorrelation can lead to increased Type I errors in hypothesis tests, which may lead to a unfair comparison among different features. The impact of these errors has not been fully explored in relational learning. Initial results indicate that they lead to overly complex models with excess structure, and may degrade model performance (Neville et al., 2003).

6.3. Inference

The literature on spatial, temporal, and social network autocorrelation models does not provide much guidance for inference because it focuses on accurate model learning rather than prediction of unobserved variables. There has, however, been preliminary work in relational learning that suggests joint inference can significantly reduce classification error, and that this reduction increases with autocorrelation. Clearly, this is an area with many open questions—e.g., Can autocorrelation be exploited to improve inference efficiency? How does autocorrelation interact with various inference procedures? How does the amount of labeled data interact with the level of autocorrelation in the dataset to determine the improvement in accuracy obtained by joint inference?

7. Conclusions

Autocorrelation is ubiquitous—datasets exhibiting autocorrelation are found in a range of fields including sociology, economics, geography, and physics—and has been studied extensively. In this paper we presented findings from econometrics, spatial statistics, and social network analysis. A common finding is that ignoring autocorrelation may result in unduly complex models, and biased, inconsistent, or inefficient estimators. The effects of autocorrelation are sometimes addressed by modeling the autocorrelation explicitly, and sometimes by using statistical procedures that are robust to these effects.

For reasons we stated earlier, we expect autocorrelation to have greater impact on relational models than on temporal, spatial, and network models. Although the presence of autocorrelation has been widely reported for relational datasets, there has been little focus on the impact of autocorrelation on model learning and inference. The results we discuss here reveal that this is an important area for future research.

Acknowledgments

The authors acknowledge helpful comments provided by the anonymous reviewers. This research is supported under a AT&T Labs Graduate Research Fellowship and by DARPA, NSF and AFRL under contract numbers F30602-00-2-0597, EIA9983215 and F30602-01-2-0566. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA, NSF, AFRL, or the U.S. Government.

References

Anderson, C., Wasserman, S., & Crouch, B. (1999). A p* primer: Logit models for social networks. *Social Networks*, *21*, 37–66.

Anselin, L. (1998). *Spatial econometrics: Methods and models*. The Netherlands: Kluwer Academic Publisher.

Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician*, *24:3*, 179–195.

Chakrabarti, S., Dom, B., & Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. *Proc of ACM SIGMOD98*.

Doreian, P. (1990). Network autocorrelation models: Problems and prospects. In *Spatial statistics: Past, present, and future*, chapter Monograph 12, pp. 369–389. Ann Arbor Institute of Mathematical Geography.

Doreian, P., & K. Teuter, C. W. (1984). Network autocorrelation models: Some monte carlo results. *Sociological Methods and Research*, *13:2*, 155–200.

Engle, R. (Ed.). (1995). *Arch: Selected readings*. Oxford: Oxford University Press.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *IJCAI* (pp. 1300–1309).

Geyer, C., & Thompson, E. (1992). Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society, Series B*, *54:3*, 657–699.

Jensen, D., & Neville, J. (2002). Linkage and autocorrelation cause feature selection bias in relational learning. *Proceedings of the 19th International Conference on Machine Learning*.

Jensen, D., Neville, J., & Gallagher, B. (2004). *Why collective inference improves relational classification* (Technical Report 04-27). University of Massachusetts.

Kennedy, P. (1998). *A guide to econometrics*. Cambridge, Massachusetts: The MIT Press.

Leenders, R. (2002). Modeling social influence through network autocorrelation: Constructing the weight matrix. *Social Networks*, *24*, 21–47.

LeSage, J., & Pace, R. (2001). Spatial dependence in data mining. In *Data mining for scientific and engineering applications*. Kluwer Academic Publishers.

Macskassy, S., & Provost, F. (2003). A simple relational classifier. *2nd Workshop on Multi-Relational Data Mining, KDD-2003*.

Marsden, P., & Friedkin, N. (1993). Network studies of social influence. *Sociological Methods and Research*, *22:1*, 127–151.

McPherson, M., Smith-Lovin, L., & Cook, J. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, *27*, 415–445.

Mirer, T. (1983). *Economic statistics and econometrics*. New York: Macmillan Publishing Co.

Neville, J., & Jensen, D. (2002). Supporting relational knowledge discovery: Lessons in architecture and algorithm design. *Data Mining Lessons Learned Workshop, 19th International Conference on Machine Learning*.

Neville, J., & Jensen, D. (2003). Collective classification with relational dependency networks. *2nd Workshop on Multi-Relational Data Mining, KDD-2003*.

Neville, J., & Jensen, D. (2004). *Dependency networks for relational data* (Technical Report 04-28). University of Massachusetts.

Neville, J., Jensen, D., Friedland, L., & Hay, M. (2003). Learning relational probability trees. *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Tanizaki, H. (2000). Bias correction of olse in the regression model with lagged dependent variables. *Computational Statistics and Data Analysis*, *34*, 495–511.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. *Proceedings of UAI-2002*.

Wooldrige, J. (2003). *Introductory econometrics: A modern approach*. South-Western College Pub.

Yang, Y., Slattery, S., & Ghani, R. (2002). A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*.

Hierarchical Probabilistic Relational Models for Collaborative Filtering

Jack Newton NEWTON@CS.UALBERTA.CA
Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8 Canada

Russell Greiner GREINER@CS.UALBERTA.CA
Department of Computing Science & AICML, University of Alberta, Edmonton, AB T6G 2E8 Canada

Abstract

This paper applies Probabilistic Relational Models (PRMs) to the Collaborative Filtering task, focussing on the EachMovie data set. We first learn a standard PRM, and show that its performance is competitive with the best known techniques. We then define hierarchical PRMs, which extend standard PRMs by dynamically refining classes into hierarchies. This represnetation is more expressive that standard PRMs, and allows greater context sensitivity. Finally, we show that hierarchical PRMs achieve state-of-the-art results on this dataset.

1. Introduction

Personalized recommender systems, which recommend specific products (e.g., books, movies) to individuals, have become very prevalent — see the success of widely used systems like Amazon.com's book recommender and Yahoo!'s LAUNCHcast music recommender system. The challenge faced by these system is predicting what each individual will want.

A pure *content-based* recommender will base this on only facts about the products themselves and about the individual (potential) purchaser. This enables us to express each possible purchase as a simple vector of attributes, some about the product and others about the person. If we also know who previously liked what, we can view this as a standard labelled data sample, and use standard machine learning techniques (Mitchell, 1997) to learn a classifier, which we can later use to determine whether a (novel) person will like some (novel) item.

To make this concrete, consider a movie recommendation system that tries to determine whether a specified person will like a specified movie — e.g., will John like Star Wars (SW)? A content-based system could use a large People×Movies database, where each tuple lists facts about a person, then facts about a movie, together with a vote (e.g., a number between 1 and 5). We could use this dataset to learn a classifier that predicts this vote, based on facts about a person and movie — here about John and about SW. There have been a number of such systems based on clustering (Breese et al., 1998) and Bayesian Models (Chickering et al., 1997), among other technologies.

Notice this prediction does not consider other people (e.g., people "similar" to John) or other movies (similar to SW).

The other main type of recommender system, *collaborative filtering*, addresses this deficiency, by using associations: If person P1 appears similar to person P2 (perhaps based on their previous "liked movies"), and P2 liked X, then perhaps P1 will like X as well. A pure collaboration-only system would use only a matrix, whose $\langle i, j\rangle$ element is the vote that person i gives to movie j, which could be unknown. The challenge, then, is using this matrix effectively, to acquire the patterns that will help us predict future votes. While there are a number of other techniques that have proven effective here, such as clustering, PCA, and K-nearest-neighbor (Ungar & Foster, 1998b; Ungar & Foster, 1998a), notice classical Machine Learning techniques do not work here, as there is no simple way to map this matrix into a simple fixed-size vector of attributes.

Of course, we would like to use *both* content and collaborative information — i.e., include, as training data, facts about the people, facts about the movies, and a set of ⟨P, M, V⟩ records, which specifies that person P gave movie M the vote of V. The challenge is how to use all of this information to predict how John will vote on SW. Here, we want to use facts about John and about SW, and also find and exploit collaborative properties, that deal with people similar to John (in terms of liking similar movies), and movies similar to SoM (in terms of being liked by similar people).

Stepping back, the challenge here is learning a distribution over a set of *databases*, here descriptions of *sets* of people and *sets* of products, as well as their votes. This is quite different from the classical machine learning challenge of learning distributions over tuples (i.e., individual rows of a single relational database), which are iid. That is, while standard techniques seek relationships *within* a row, (e.g., relating `a.vote` to `a.gender` and `a.movieType`), our collaborative system needs to reason *across* rows — e.g., to decide that John (described in one row) is sufficiently like George (described in another row) that we use facts about George to make inferences about John. Another natural inter-row application is based on *sets* of rows: e.g., we might use the fact that the *set* of people with some characteristic (e.g., `Age=teenage`, `gender=male`) typically like members of a *set* movies with some other characteristic (e.g., `Genre=action`).

Probabilistic Relational Models (PRMs) (Koller & Pfeffer, 1198) were designed to address exactly this type of relational learning and inference problem. This paper shows that PRMs can be successfully applied to this learning scenario, in the context of the Recommendation task. We examine the effectiveness of standard PRMs applied to the recommendation task on the EachMovie (EachMovie,) dataset, then evaluate the effectiveness of an extended version of PRMs called "hierarchical PRMS" (hPRMs) (Getoor, 2002). Our empirical results show that standard PRMs can achieve competitive results on the recommendation task, and then that hPRMs can outperform standard PRMs here.

As PRMs can be viewed as a relational extension of Belief Nets, Section 2 describes standard PRMs by showing how they extend Bayesian Networks; in particular, we provide both inference and learning algorithms here. It then presents our application of the PRM framework to the Recommendation task. Section 3 describes some limitations of standard PRMs for this task; addressing these limitations leads naturally to hierarchical PRMs. We introduce our implementation of hierarchical PRMs, and show how an hPRM can provide a more expressive model of the EachMovie dataset. Finally, Section 4 demonstrates the overall effectiveness of PRMs as a recmmendation system, and in particular the superiority of hPRMs over standard PRMs.

2. Probabilistic Relational Models

A PRM can encode *class-level* dependencies that can subsequently be used to make inferences about a particular instance of a class. For example, we might connect (the class of) teenage boys to (the class of) action movies, then use that to infer that the teenage boy `John` will like the action movie `SW`. Of course, we could do something like that in a standard Belief Network, by first transforming this relational information into a non-structured, propositionalized form. However, by performing this transformation we lose the rich relational structure and introduce statistical skews (Getoor, 2002). A PRM can be learned directly on a relational database, thereby retaining and leveraging the rich structure contained therein.

We base our notation and conventions for PRMs on those used in (Getoor, 2002). In general, a PRM is a pair $\langle \mathcal{S}, \theta_{\mathcal{S}} \rangle$ defined over a Relational Schema $\mathcal{R}$, where $\mathcal{S}$ is the qualitative dependency structure of the PRM and $\theta_{\mathcal{S}}$ is the set of associated parameters. The *Relational Schema* $\mathcal{R}$ contains two fundamental elements: a set of *classes*, $\mathcal{X} = \{X_1, \ldots, X_n\}$, and a set of *reference slots* $\{\rho_i\}$ that define the relationships between the classes.

Each class $X \in \mathcal{X}$ is composed of a set of *descriptive attributes* $\mathcal{A}(X)$, which in turn take on a range of values $V(X.A)$. For example, consider a schema describing a domain describing votes on movies. This schema has three classes: `Vote`, `Person`, and `Movie`. For the `Vote` class, the single descriptive attribute is `Score` with values $\{\texttt{0}, \ldots, \texttt{5}\}$; for `Person` the two descriptive attributes are `Age` and `Gender`, which take on values {`young`, `middle-aged`, `old`} and {`Male`, `Female`} respectively; and for `Movie` the single descriptive attribute is `Rating` which takes on values {`G`, `PG`, `M`, `R`}. Furthermore, a class can be associated with a set of *reference slots*, $\mathcal{R}(X) = \{\rho_1, \ldots, \rho_k\}$. A particular reference slot, $X.\rho$, describes how objects of class X are related to objects in other classes in the relational schema. Continuing our example, the `Vote` class would be associated with two reference slots: `Vote.ofPerson`, which describes how to link `Vote` objects to a specific `Person`; and `Vote.ofMovie`, which describes how to link `Vote` objects to a specific `Movie` object. A sequence of one or more reference slots can composed to form a *reference slot chain*, $\tau = \rho_1 \circ \cdots \circ \rho_\ell$, and attributes of related objects can be denoted by using the shorthand $X.\tau.A$, where A is a descriptive attribute of the related class. For example, `Vote.ofPerson.Gender` refers to the gender attribute of the `Person` associated with a given `Vote`.

The dependency structure for a PRM defines the parents $Pa(X.A)$ for each attribute $X.A$. The parent for an attribute $X.A$ is a descriptive attribute, which can be within the class X, or within another class Y that is reachable through a reference slot chain. For instance, in the above example, `Vote.Score` could have the parent `Vote.ofPerson.Gender`.

In many cases, the parent of a given attribute will take on a *multiset* of values $\mathcal{S}$ in $V(X.\tau.A)$. For example, there could be discover a dependency of a `Person`'s age on their rating of movies in the PRMclassChildren genre. However, we cannot directly model this dependency since the user's ratings on `Children`'s movies is a multiset of values, say {`4`, `5`, `3`, `5`, `4`}. For such a numeric attribute, we may choose to use the `Median` database aggregate operator to reduce this multiset to a single value, in this case 4. In this paper we reduce $\mathcal{S}$ to a single value using various types of aggregation functions.

The following definition summarizes the key elements of a PRM:

Definition 1 ((Getoor, 2002)) *A* probabilistic relational model (PRM) $\Pi = \langle \mathcal{S}, \theta_{\mathcal{S}} \rangle$ *for a relational schema* $\mathcal{R} = \langle \mathcal{X}, \mathcal{A} \rangle$ *is defined as follows. For each class* $X \in \mathcal{X}$ *and each descriptive attribute* $A \in \mathcal{A}(X)$*, we have a set of* parents Pa($X.A$)*, and a* conditional probability distribution (CPD) *that represents* $P_{\Pi}(X.A|Pa(X.A))$*,*

2.1. Applying Standard PRMs to the EachMovie Dataset

PRMs provide an ideal framework for capturing the kinds of dependencies a recommender system needs to exploit. In general, model-based collaborative filtering algorithms try to capture high-level patterns in data that provide some amount of predictive accuracy. For example, in the EachMovie dataset, one may want to capture the pattern that teenage males tend to rate Action movies quite highly, and subsequently use this dependency to make inferences about unknown votes. PRMs are able to model such patterns as class-level dependencies, which can subsequently be used at an instance level to make predictions on unknown ratings — i.e., how will John vote on `SW`.

In order to use a PRM to make predictions about an unknown rating, we must first learn the PRM from data. In our experiments we use the PRM learning produce described in (Friedman et al., 1999), which provides an algorithm for both learning a legal structure for a PRM and estimating the parameters associated with that PRM. Figure 1(a) shows a sample PRM structure learned from the EachMovie dataset.

With the learned PRM in hand, we are left with the task of making an inference about a new, previously unseen `Vote.score`. To accomplish this task, we leverage the *ground Bayesian Network* (Getoor, 2002) induced by a PRM. Briefly, a Bayesian Network is constructed from a database using the link structure of the associated PRM's dependency graph, together with the parameters that are associated with that dependency graph. For example, for the PRM in Figure 1(a), if we needed to infer the Score value for a new `Vote` object, we simply construct a ground Bayesian Network using the appropriate attributes retrieved from the associated `Person` and `Movie` objects; see Figure 1(b). The PRM's class-level parameters for the various attributes are then tied to the ground Bayesian Network's parameters, and standard Bayesian Network inference procedures can be used on the resulting network (Getoor, 2002).

3. Hierarchical Probabilistic Relational Models

3.1. Motivation

The collaborative filtering problem reveals two major limitations of PRMS, which in turn motivate hPRMs. First, in the above model, `Vote.Score` can depend on attributes of related objects, such as `Person.Age`, but it is not possible to have `Vote.Score` depend on itself in any way. This is because the class-level PRM's dependency structure must be a directed acyclic graph (DAG) in order to guarantee that the instance-level ground Bayesian Network forms a DAG (Friedman et al., 1999), and thus quality as a well-formed probability distribution. Without the ability to have `Vote.Score` depend probabilistically on itself, we lose the ability to have a user's rating of an item depend on his rating of other items or on other user's ratings on this movie, which is critical for a collaborative system. For example, we may wish to have the user's ratings of `Comedies` influence his rating of `Action` movies, or his rating of a specific `Comedy` movie influence his ratings of other `Comedy` movies; or Collaborative Filtering: use person `A`'s rating on a movie to predict person `B`'s rating. Second, in the above model, we are restricted to one dependency graph for `Vote.Score`; however, depending on the type of object the rating is for, we may wish to have a specialized dependency graph to better model the dependencies. For example, the dependency graph for an `Action` movie may have `Vote.Score` depend on `Vote.PersonOf.Gender`, whereas a `Documentary` may depend on `Vote.PersonOf.Age`.

3.2. Overview

To address the problems described above, we introduce a class hierarchy that applies to our dataset, and modify the PRM learning procedure to leverage this class hierarchy in making predictions. In general, the class hierarchy can either be provided as input, or can be learned directly from the data. We refer to the class hierarchy for class X as $H[X]$. Figure 2 shows a sample class hierarchy for the EachMovie domain. $H[X]$ is a DAG that defines an IS-A hierarchy using the subclass relation $\prec$ over a finite set of subclasses $\mathcal{C}[X]$ (Getoor, 2002). For a given $c, d \in \mathcal{C}[X]$, $c \prec d$ indicates that X_c is a *direct subclass* of X_d (and X_d is a *direct superclass* of X_c). The leaf nodes of $H[X]$ represent the *basic subclasses* of the hierarchy, denoted *basic*($H[X]$). In this paper we assume all objects are members of a basic subclass, although this is not a fundamental restriction of hPRMs. Each object of class X has a subclass indicator $X.Class \in basic(H[X])$, which can either be specified manually or learned automatically by a supplementary algorithm. By defining a hierarchy for a class X in a PRM, we also implicitly specialize the classes

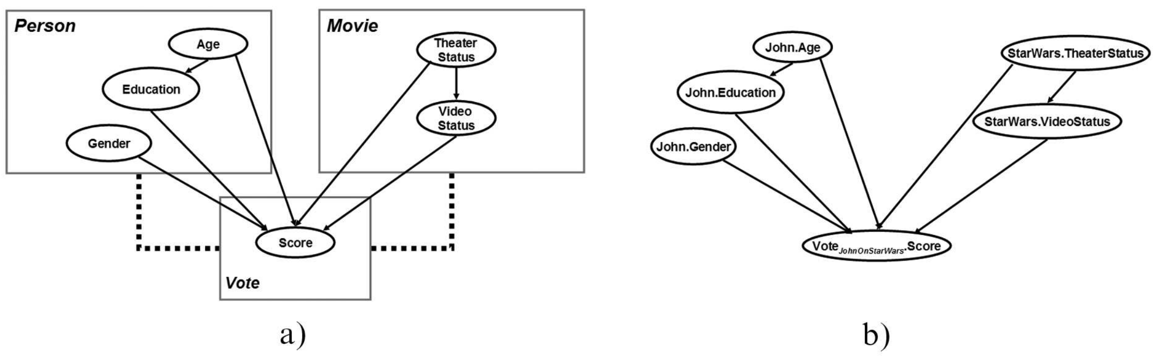


Figure 1. (a) Standard PRM learned on EachMovie dataset (b) Ground Bayesian Network for one `Vote` object

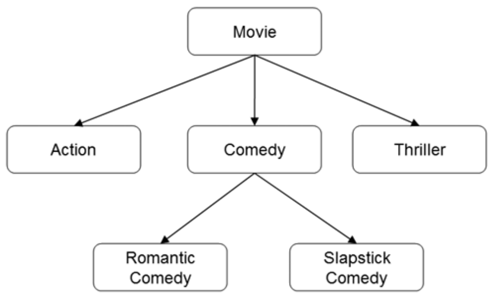


Figure 2. Sample class hierarchy

that are reachable from X via one or more reference slots. For example, if we specialize the `Movie` class, we implicitly specialize the related `Vote` table into a hierarchy as well. For example, in Figure 3, the `Vote` class is refined into four leaf classes, each associated with one of the hierarchy elements in $basic(H[X])$.

Definition 2 *The components of a* Hierarchical Probabilistic Relational Model (hPRM) Π_H *are:*

- *A class hierarchy* $H[X] = (\mathcal{C}[X], \prec)$
- *A set of basic, leaf-node elements* $basic(H[X]) \subset H[X]$
- *A subclass indicator attribute* $X.Class \in basic(H[X])$
- *For each subclass* $c \in \mathcal{C}[X]$ *and attribute* $A \in \mathcal{A}(X)$ *we have a specialized CPD for* c *denoted* $P(X^c.A|Pa^c(X.A))$
- *For every class* Y *reachable via a reference slot chain from* X *we have a specialized CPD for* c *denoted* $P(Y^c.A|Pa^c(Y.A))$

The algorithm for learning an hPRM is very similar to the algorithm for learning a standard PRM. Instead of dealing with the standard set of classes $\mathcal{X}$ when evaluating structure quality and estimating parameters, our hPRM algorithm dynamically partitions the dataset into the subclasses defined by $H[X]$. For inference, a similar technique is used, as for any given instance i of a class, i's place in the hierarchy is flagged through `X.Class`; using this flag it is possible to associate the proper CPD with a given class instance.

3.3. Applying hPRMs to the EachMovie Dataset

Applying the hPRM framework to the EachMovie dataset first requires a hierarchy to be defined, which is then used to build an hPRM that is ultimately used to make predictions for unknown votes.

In our experiments we automatically learn a hierarchy to be used in the learning procedure. In the EachMovie database, a movie can belong to zero or more of the following genre categories: { `action`, `animation`, `art_foreign`, `classic`, `comedy`, `drama`, `family`, `horror`, `romance`, `thriller` }.

We let $G(\chi)$ denote the set of genres that the movie χ belongs to. For example, $G($`WhenHarryMetSally`$) = \{$`comedy`, `drama`, `romance`$\}$. To build our hierarchy dynamically, we first enumerate all combinations of genres that appear in the EachMovie database, and denote this set $\mathcal{G}$. Of course, this set is significantly smaller than the entire 2^ℓ power-set of all possible subsets of the ℓ genres. We also store the number of movies associated with each element of $\mathcal{G}$. We then proceed to greedily partition $\mathcal{G}$ based on this quantity, until reaching a predefined limit of k partitions. (Here, we used $k = 11$.) We define one additional partition that is used for movies that do not fall into one of the predefined partitions. This partition, together with the other k partitions, are used to create a $k + 1$-element hierarchy.

Given this hierarchy, the hPRM learning algorithm is applied to the EachMovie dataset, using the same algorithm used for learning standard PRMs (Section 2.1), with the exception that the learning procedure is modified as outlined

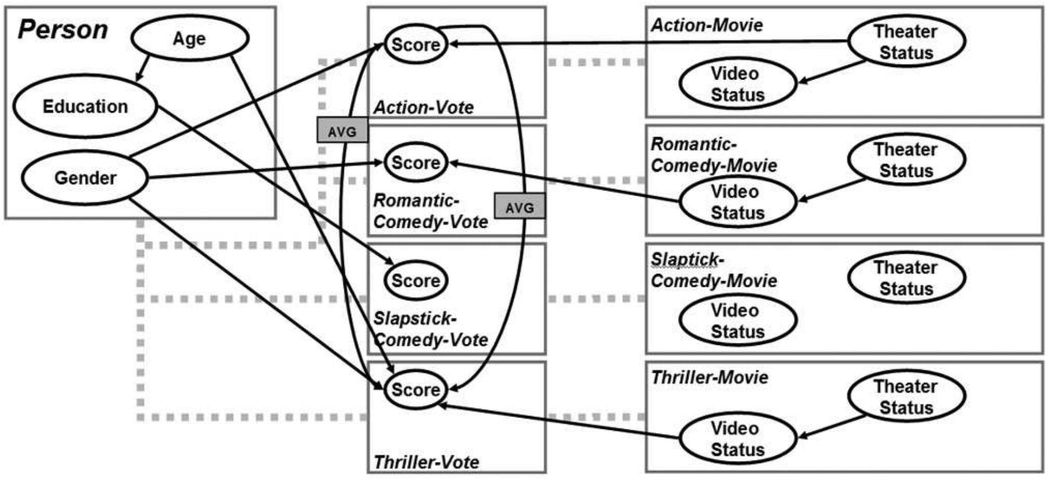


Figure 3. Example hPrm for EachMovie dataset

above.

4. Experimental Results

This section outlines our results in applying both standard PRMs and hPRMs to the recommendation task for the EachMovie dataset. We also compare our results to other recommendation algorithms.

4.1. Experimental Design

One of the main challenges in designing an experiment to test the predictive accuracy of a PRM model is in avoiding resubstitution error. If a PRM is learned on the entire EachMovie database, and subsequently used to make predictions on objects from the same database, we are using the same data for testing as we used for training.

The standard learning paradigm typically splits the data into n subsets, and trains on $(n-1)/n$ of the data, and test of the remaining $1/n$ subset. This simple approach does not apply here, as it is not trivial to divide the data into "independent" compartments, as the movies and people are intertwined

We address this issue by applying a modified cross-validation procedure to the dataset. While the traditional method of dividing data into cross-validation folds cannot be applied directly to a relational database, we extend the basic idea to a relational setting as follows. For n-fold cross validation, we first create n new datasets $\{D_1, \ldots, D_n\}$ with the EachMovie data schema. We then iterate over all the objects in the `Person` table, and randomly allocate the individual to one of $D_i \in \{D_1 \ldots D_n\}$. Finally, we add all the `Vote` objects linked to that individual, and all the `Movie` objects linked to those $Vote$ objects, to D_i. This procedure, when complete, creates n datasets with roughly balanced properties, in terms of number of individuals, number of votes per person, etc. In our experiments we use 5-fold cross validation.

Algorithm	Absolute Deviation
CR	1.257
BC	1.127
BN	1.143
VSIM	2.113
PRM	**1.26**

Table 1. Absolute Deviation scoring results for EachMovie dataset, using "Two Non-Active Votes" (Breese et al., 1998)

4.2. Evaluation Criteria

In this paper we adopt the *Absolute Deviation* metric (Miller et al., 1997; Breese et al., 1998) to assess the quality of our recommendation algorithms. We divide the data into a training and test set using the method described above, and build a PRM (resp., hPRM) using the training data. We then iterate over each user in the test set, allowing each user to become the *active user*. For the active user we then iterate over his set of votes, P_a, allowing each vote to become the *active vote*; each of the remaining votes are used in the PRM model. We let $p_{a,j}$ denote the predicted vote for the active user a on movie j, and $v_{a,j}$ denote the actual vote. The average absolute deviation, over the m_a vote predictions made, is:

$$S_a = \frac{1}{m_a} \sum_{j \in P_a} |p_{a,j} - v_{a,j}| \tag{1}$$

The absolute deviation for the dataset as a whole is the average of this score over all the users in the test set of users.

Algorithm	Absolute Deviation
CR	0.994
BC	1.103
BN	1.066
VSIM	2.136
hPRM	**1.060**

Table 2. Absolute Deviation scoring results for EachMovie dataset, using “All-But-One Votes” (Breese et al., 1998)

4.3. Standard PRMs

In our experiments, we were able to achieve an absolute deviation error of 1.26. For comparison, Table 1 includes the results from (Breese et al., 1998): correlation (CR), Bayesian Clustering (BC), a Bayesian Network model (BN), and Vector Similarity (VSIM). We have elected to include the results from (Breese et al., 1998) where algorithms were given two votes out of the non-active votes to use in making the prediction, since the standard PRM model does not have any direct dependency on other `Votes`.

In this experiment, standard PRMs are able to outperform the VSIM algorithm, and is competitive with the correlation-based algorithm. However, both Bayesian Clustering and the Bayesian Network model have superior results in this context.

4.4. Hierarchical PRMs

The first part of the experiment for hPRMs was constructing a class hierarchy from the EachMovie dataset. In our experiment we set the size of the hierarchy to be 12. Our greedy partitioning algorithm arrived at the following basic classes: { `drama`, `comedy`, `classic`, `action`, `art-foreignDrama`, `thriller`, `romance-comedy`, `none`, `family`, `horror`, `actionThriller`, `other` }.

By applying hPRMs to the EachMovie dataset, we are able to reduce the absolute deviation error from 1.26 (with standard PRMs) to 1.06. Again, for comparison Table 2 includes results from (Breese et al., 1998); however, since hPRMs are able to leverage other votes the user has made in making predictions, we use the *All-But-One* results presented in (Breese et al., 1998), where the prediction algorithm is able to use all of the active user’s votes (except for the current active vote) in making a prediction. Comparing Table 1 to Table 2, We see that including the additional voting information results in a substantial reduction in error rate for most of the other four algorithms.

hPRMs not only provide a significant performance advantage over standard PRMs, but are also able to outperform all but one of the other four algorithms.

5. Conclusion

In this paper we outlined a framework for using PRMs to model the recommendation task. We first use a standard PRM, then extend this representation to hPRMs, to account for hierarchical relationships that are present in the data. hPRMs improve the expressiveness and context-sensitivity of standard PRMs, and also realize real-world performance benefits.

ACKNOWLEDGEMENTS

We thank Lise Getoor for useful discussions and encouragement (as well as access to her software), Dale Schuurmans for his advice on this project, and thank NSERC, iCORE and the Alberta Ingenuity Centre for Machine Learning for funding.

References

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *UAI98* (pp. 43–52).

Chickering, D. M., Heckerman, D., & Meek, C. (1997). A Bayesian approach to learning Bayesian networks with local structure. *UAI97* (pp. 80–89).

EachMovie. http://research.compaq.com/SRC/eachmovie/.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *IJCAI-99* (pp. 1300–1309).

Getoor, L. (2002). *Learning statistical models from relational data*. Doctoral dissertation, Stanford University.

Koller, D., & Pfeffer, A. (1198). Probabilistic frame-based systems. *Proc. of the Fifteenth National Conference on Artificial Intelligence* (pp. 580–587). Madison, WI.

Miller, B. N., Riedl, J. T., & Konstan, J. A. (1997). Experience with GroupLens: Making Usenet useful again. *USENIX* (pp. 219–233).

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

Ungar, L., & Foster, D. (1998a). Clustering methods for collaborative filtering. *Proceedings of the Workshop on Recommendation Systems*.

Ungar, L., & Foster, D. (1998b). A formal statistical approach to collaborative filtering. *CONALD’98*.

Learning Complex Motion Structures

Fabio Tozeto Ramos F.RAMOS@ACFR.USYD.EDU.AU
Hugh F. Durrant-Whyte HUGH@ACFR.USYD.EDU.AU
ARC Centre of Excellence in Autonomous Systems (CAS), Australian Centre for Field Robotics, The University of Sydney, NSW 2006, Sydney, Australia.

Abstract

This paper presents a general methodology for learning complex motions that, despite the fact have non-linear correlations, are cyclical and consequently have a defined pattern of behaviour. Using conventional algorithms to extract features from images, a Bayesian classifier is applied to cluster and classify those features. Clusters are then associated in different frames and structure learning algorithms for Bayesian networks are used to recover the structure of the motion. Applications of these techniques can be from human motion to multi-robots behaviour analysis.

1. Introduction

A key challenge in robotics is how to learn a representation of an unstructured world given only a set of sequential measurements. As the robot moves, an object is seen from different perspectives and parts may be occluded by other objects. In addition, sensor measurements may be erroneous, so requiring a representation able to handle uncertainty. A possible approach to these problems is to employ a state estimator such as a Kalman filter (KF) or Extended Kalman Filter (EKF). These estimators describe the process of state transition and observation, and generate an estimate that minimises estimated mean square error. However, most applications of KFs consider only point targets or objects represented by a group of points with the same dynamic model. In this paper, we are interested in tracking the motion of complex structures, with correlations between parts of the same structure which may, nevertheless, execute separate but correlated motion. The techniques developed are applicable to problems such as tracking human motion or the coordinated motions of a set of robots.

The human tracking problem has been widely studied, especially in the computer vision community. It can be formulated in a probabilistic manner with two different approaches, one based on point features and another based on intensity. Feature-based approaches have the advantage of being able to employ many different algorithms for feature extraction and are generally more amenable to real-time implementation. However, they have additional problems in associating features from different image frames. In (Song et al., 2003) a probabilistic framework is used to identify joints in the human body. Triangulated graphs are used to represent the structure of the body which can be learnt with an EM-like algorithm. Labelling and classification of features is achieved through maximising the likelihood of the data given the decomposition represented by the triangulated graphs. In our approach, rather than labelling each feature from an existent structural model, we first cluster features using the EM algorithm and then learn the structural model by finding correlations between clusters. Features are extracted from a stream of frames with the KLT algorithm (Tomasi & Kanade, 1991) and contain positions and velocities. Then, EM is used to cluster these features under the assumption that positions and velocities are independent given the class. In other words, a Naive Bayes classifier (Friedman et al., 1997), represented as a Bayesian network, is learnt with the class variable being hidden. Once the parameters are learnt, the classifier can then be applied in features in different frames, making the association task straightforward.

With features labelled in every frame, it is then possible to learn dependencies among clusters, so building a Bayesian network model of the motion. In complex structures, dependencies can be non-linear, i.e. variables may be function of a non-linear combination of its descendants. Unfortunately, learning a Bayesian network with continuous nodes and non-linear relations between variables, even assuming these to be Gaussian distributed, is a cumbersome task where Monte Carlo algorithms must generally be applied

(Doucet et al., 2001). It is shown how to tackle this problem by representing non-linear dependencies as a set of net structures, with linear Gaussians distributions. For each frame, a network structure is learnt along with its correlations with the previous frame. As motions are usually periodic, the learning process can stop when the structures have the same dependencies as those previously learnt.

This paper is organised as follows: in Section 2 we present formal definitions and a brief review of Bayesian networks. Section 3 shows how to cluster features in an unsupervised fashion using the EM algorithm. Section 4 presents the structure and parameters learning algorithms along with some experimental results. We conclude in Section 5 and present some ideas for future work.

2. Preliminaries

This section briefly reviews Bayesian networks and introduces necessary notation. Capital letters (X, Y, Z) are used to denote names of random variables, lowercase letters (x, y, z) to denote specific values taken by those variables, boldface capital letters $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ to denote sets of random variables and boldface lowercase variables $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ to denote values taken by those sets. A joint probability over a set $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ is denote by $P(\mathbf{X})$.

A Bayesian network is defined as a tuple $\mathcal{B} = \langle \mathcal{G}, \Theta \rangle$ where $\mathcal{G}$ is a directed acyclic graph whose vertices represent random variables and Θ are the parameters that define the distributions. The main assumption encoded by a BN is that each variable X_i is conditionally independent of its non-parents given its parents. The joint probability is defined by:

$$P(\mathbf{X}) = \prod_i P(X_i|\mathbf{Pa}(X_i)),$$

where $\mathbf{Pa}(X_i)$ represent the parents of the variable X_i.

In this paper, we use Bayesian networks for two different tasks: 1) unsupervised classification of features and 2) learning and representation of the structure of the motion. Except for the class variable of the classifier, all other variables are assumed to have a normal (Gaussian) distribution with parameters μ and σ^2, with the distribution denoted by $\mathcal{N}\left(X; \mu, \sigma^2\right)$. Then, assuming linear relation between Gaussians and an order $X_1, \ldots, X_n$ of variables, it is possible to define linear conditional Gaussian distributions as:

$$P\left(X_i|X_1, \ldots, X_{i-1}\right) = \mathcal{N}\left(X_i; \beta_{i,0} + \sum_{j=1}^{i-1} \beta_{i,j} X_j, \sigma_i^2\right),$$

where $\beta_{i,0}$ and $\beta_{i,j}$ describe the linear combination of the variable X_i given its parents $X_1, \ldots, X_{i-1}$. When $\beta_{i,j} \neq 0$, there is an edge from X_j to X_i forming a graph. Thus, this definition brings linear Gaussian distributions into Bayesian networks. If $\beta_{i,j} = 0$ for every i and j, the variable X_i is a root node with a univariate Gaussian distribution. The joint probability distribution with all variables being Gaussian is then $\mathcal{N}(\mathbf{X}; \mu, \Sigma)$, defined as:

$$P(\mathbf{X}) = \tfrac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right),$$

where μ is a vector of size n and Σ is a symmetric positive-definite matrix of size $n \times n$.

Making inferences in a Bayesian network is the task of computing posterior probabilities given some observed values. That is, given a set of *query* variables $\mathbf{X}_q$ and a set of *evidence* $\mathbf{X}_e = \mathbf{x}_e$, we compute $P\left(\mathbf{X}_q|\mathbf{X}_e = \mathbf{x}_e\right)$ which, with continuous distributions, is proportional to the marginalisation of the joint probability over variables $\mathbf{X}_z$, where $\mathbf{X}_z = \mathbf{X} \setminus (\mathbf{X}_q \cup \mathbf{X}_e)$:

$$P\left(\mathbf{x}_q|\mathbf{x}_e\right) \propto \int \prod_i P(x_i|\mathbf{Pa}(x_i))\, d\mathbf{x}_z.$$

Algorithms for inference in these models are discussed in (Lauritzen, 1992; Murphy, 1998; Lerner, 2002). These algorithms describe Gaussian distributions using *canonical characteristics* and perform message-propagation in a *juction tree* (Huang & Darwiche, 1996) to calculate marginal distributions. A limitation exists when there are deterministic relations between variables since the covariance matrix Σ is not invertible, and the canonical form needs to invert the covariance matrix to calculate one of its terms. To overcome this problem, it is possible to use conditional forms (Lauritzen & Jensen, 1999) which are also more numerically stable than canonical forms when the net has both discrete and continuous variables. A deeper discussion of inference with linear Gaussian distributions is beyond the scope of this paper.

In a frequentist approach, the parameters of linear Gaussian models can be learnt using Maximum-Likelihood techniques. See (Murphy, 2002; Lauritzen, 1996) for details.

3. Unsupervised Feature Classification

We start our discussion about learning motion structures by analysing the problem of feature association. Given a set of features extracted by an algorithm like KLT, the first step towards structure reconstruction is to associate features from different frames. In a complex environment with changes in luminosity, occlu-

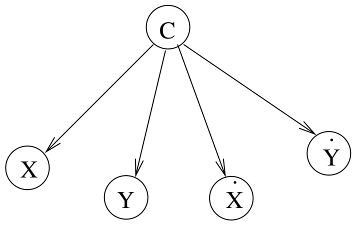


Figure 1. The Naive Bayes classifier to cluster features.

sions, rotations and translations of objects, features can appear and disappear from frame to frame. If there is no predefined dynamic model describing the behaviour of such features, the problem of predicting the position of a particular occluded feature becomes very complex. In the same way, association of that feature fails due to lack of observability. In this work, instead of trying to track individual features fixed in an object, features are clustered using probabilistic methods and only the created clusters are tracked. We advocate that this method is more robust in dealing with occlusions and inaccurate information from the feature extraction algorithm than methods that try associate features individually.

To classify and cluster features we use the well-known Naive Bayes classifier. The Naive Bayes classifier (Friedman et al., 1997) assumes that the attributes are conditionally independent given the class. This assumption is quite reasonable in our problem whose attributes are positions and velocities for the features extracted. Note that at this point, there is no association between features in consecutive frames so that velocities and positions are independent. In the Naive Bayes model, the probability of a specific label c given the observed attributes is given by:

$$P(c|x, y, \dot{x}, \dot{y}) = P(x|c)\, P(y|c)\, P(\dot{x}|c)\, P(\dot{y}|c)\, P(c)\,.$$

A feature will belong to the label that maximises the posterior probability. Figure 1 shows the Bayesian network representing the Naive Bayes classifier used to cluster features.

An alternative way to classify features is through a dynamic Naive Bayes classifier. In this case it is assumed that the class describes a stochastic process $\{C(t)\,, t \in T\}$ where t is a time slice - or a frame - in the stream. If assumed that this process is stationary, the dynamic classifier can be represented as a dynamic Bayesian network with transitions given by $P(C(t)\,|C(t-1))$.

In the unsupervised approach, a Naive Bayes classifier can be learnt using maximum-likelihood techniques

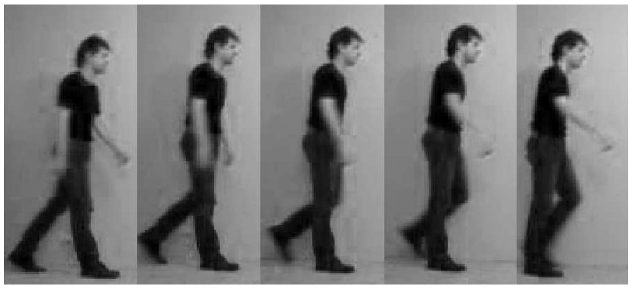


Figure 2. An example of a sequence of frames from a person walking from the right to the left side of the scene.

such as the EM algorithm (Dempster & Rubin, 1977; Neal & Hinton, 1993). The main idea of the EM algorithm is to apply the Jensen's inequality (Cover & Thomas, 1991) to simplify the computation of the log-likelihood. At each interaction, the EM computes the expected value of the hidden variables given the current data and parameters (E-Step). Then, it finds new values for the parameters that maximise the likelihood (M-Step). The only parameter that has to be defined *a priori* is the number of categories that the class variable can have. This value is equal or larger than the number of clusters identified with EM - it is larger if EM finds no feature for a particular cluster. 10 categories are used in our experiments.

Using EM the classifier can be trained with the features extracted by the KLT algorithm whose attributes are positions and velocities for all features detected, regardless of which frame they come from. To do so, it is necessary to remove possible translations from the position variables. For example, suppose that the motion recorded in a video is of a person walking from the left to the right side of the screen, with the camera remaining fixed during the whole video acquisition. Figure 2 shows five frames of this example grabbed with a camera of 320 x 240 pixel resolution. As the person walks, the x position of the detected features changes accompanying the body motion. In order to make the Gaussian assumption reasonable, the translation is removed by subtracting the mean of the x positions of all features detected in a particular frame from the x position of each feature in that frame. For each feature i detected in the frame t its corrected $x_i(t)$ position is given by:

$$x_i(t) = x_i(t) - \mu_x(t),$$

where $\mu_x(t)$ is the mean of the x position of all features detected in frame t.

The data set for the Naive Bayes classifier is thus a set $\mathcal{D} = \{\mathbf{d}_{1,1}, \mathbf{d}_{1,2}, \ldots, \mathbf{d}_{1,N_1}, \ldots, \mathbf{d}_{T,1}, \mathbf{d}_{T,2}, \ldots, \mathbf{d}_{T,N_T}\}$,

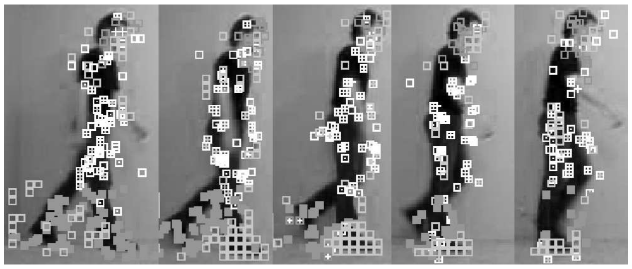


Figure 3. Features clustered using the learnt Naive Bayes. Features represented with the same symbol belong to the same cluster.

where T is the number of frames in the stream and N_i is the number of features detected in the frame i with each sample $\mathbf{d}_{i,j} \in \mathbb{R}^4$ and $\mathbf{d}_{i,j} = \{x, y, \dot{x}, \dot{y}\}^T$.

With all parameters determined, features are clustered by making inferences on the Bayes net of Figure 1. Resulting clustered features are classified with a unique label. Figure 3 shows the result of this process on the sequence of Figure 2. Note that features with velocities close to nil are represented with light gray unfilled squares. They move to the foot in contact with the ground since velocities in this region are zero. Another interesting cluster is the one represented by dark gray unfilled squares. Features of this cluster are associated with the movement of the head and remain accompanying it during the whole sequence.

By making inferences with evidence from features detected across frames, it is possible to associate clusters in the whole stream. The samples in the data set are then modified to incorporate one more dimension representing their labels. Thus, $\mathbf{d}_{i,j} \in \mathbb{R}^5$ and $\mathbf{d}_{i,j} = \{x, y, \dot{x}, \dot{y}, c\}^T$where c is the label or cluster that the feature belongs.

4. Learning the Motion Structure

With a group of samples for each cluster, in each frame (time slice) the motion structure can be learnt using structure learning algorithms for Bayesian networks. However, complex motions may have non-linear dependencies, and a linear Gaussian network may not be directly applicable. Our strategy to tackle this problem is to approximate non-linear relations to linear relations, learning a different structure for each time slice, until structures start repeating. Our assumption is that, even in complex motions like a human body walking, there exists a pattern that is repeated over some (unknown) time interval. The idea is to try to learn this pattern and then construct a Bayesian network to describe it. As conditional probabilities and dependencies change with time, a dynamic Bayesian network, as it is normally defined, would not be appropriate to represent the model. Nonetheless, it is possible to consider the whole motion pattern learnt with a Bayesian network as the structure repeated in a dynamic Bayesian network. Thus, with a slight change in the definition of DBNs, the problem can be described in the form of DBN structure learning.

Given the data set with labelled samples, the algorithm works as follows. Suppose that in the frame t there are n clusters, $C_t^1, \dots C_t^n$, identified with at least m features per cluster, using the procedure described in Section 3. In the next frame $t+1$, the same n clusters are identified, $C_{t+1}^1, \dots, C_{t+1}^n$, with m samples per cluster[1]. The first step of the algorithm is to learn the structure represented by the clusters in the first frame. Structure learning in a Bayes net involves a search over the set of all possible directed acyclic graphs, scored by a determined scoring function. As the number of possible graphs grows super-exponentially with the number of variables, a heuristic strategy must be used. In this work, the scoring function used is the well known Bayesian Information Criterion (BIC) (Heckerman, 1996) which is equivalent to the Minimum Description Length (MDL) approach (Suzuki, 1998). Essentially the BIC has one term that is exactly the likelihood, measuring how well the model predicts the data, and one term to penalise the complexity of the model:

$$\mathrm{BIC}(\mathcal{G}) = \sum_i \sum_n \log P\left(X_i | Pa(X_i), \hat{\theta}_i, D^n\right) - \frac{np_i}{2} \log N,$$

where np_i is the number of parameters in the distribution of X_i and N is the number of samples. A greedy search is used to find the graph that maximises the scoring function. The search starts with a fully connected graph and operations of adding, removing and reverting edges are performed until a local maximum is obtained.

Having learnt the structure for frame t, the same procedure is repeated for frame $t+1$, and another structure is learnt. With two consecutive structures, correlations between clusters in different frames are discovered. This can be done using the same greedy search heuristic, under the constraint that clusters in frame $t+1$ cannot be parents of clusters in frame t. This ensures a Markov assumption where variables are independent of the past given the present.

[1]In the case that more than m features were identified, some of them can be excluded by selecting the m features that have higher probability of belonging to that particular cluster.

Algorithm 1 A pseudo-code for Motion Structure Algorithm.

```
Inputs: A set of labelled features $\mathcal{D}$;
        KL threshold, $k$.
Output: Learnt BN encoding the motion pattern, $\mathcal{B}$.
While $stop > k$ do
    $B_t \leftarrow$greedy_search$(D, t)$
    $B_{t+1} \leftarrow$greedy_search$(D, t+1)$
    $B_t^{t+1} \leftarrow$greedy_search$(D, t, t+1)$ //inter dep.
    $\mathcal{B} \leftarrow \mathcal{B} + \langle B_t, B_{t+1}, B_t^{t+1} \rangle$
    $t \leftarrow t+1$
    $stop \leftarrow KL(B_t; B_{t_0})$
End
```

Figure 4 shows an example of a learnt structure for two consecutive frames. Note that in contrast to a Dynamic Bayesian Network where the net has the same structure for every time step t with $t \neq t_0$, the network structure of Figure 4 differs in the two consecutive frames. The algorithm continues learning structures and inter-frame dependencies until the new learnt structures start being *similar* to those previously learnt, indicating that the cycle has finished. *Similar* in this case is understood in terms of relative entropy or Kullback-Leibler divergence (Cover & Thomas, 1991). Thus, the learning process stops when a defined threshold of KL divergence is achieved. A sketch of the algorithm is shown in Algorithm 1.

Results from the complete algorithm are presented in Figure 5 for the first five frames of a motion pattern. The motion pattern has 19 inter-connected structures representing the whole cycle of a typical human gait. Edges represented with solid lines indicate conditional dependencies between clusters in the same frame, while edges represented with dash lines show the inter-frame correlation. From this figure it is possible to note that clusters associated with the trunk, such as $C4$, $C5$ and $C6$, are normally the parents of other clusters in inter-frame correlations. They represent the centre of the body where the movement of other parts are based on and therefore, tend to have more correlations. Besides, the trunk has a movement closer to linear than other parts such as the limbs. Thus, it is expected that they have similar inter-frame correlations between themselves.

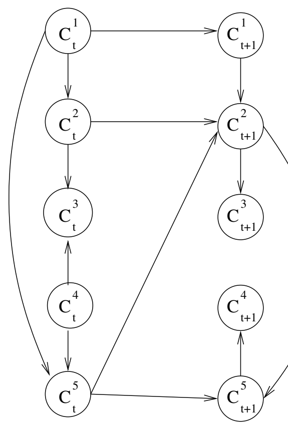


Figure 4. Structure learnt from samples of 10 clusters into 2 consecutive frames.

5. Conclusions and Future Work

The algorithm described in this paper provides a general methodology to learn complex motion structures that have specific patterns. With a set of features extracted from a video, clusters are identified and tracked. These represent characteristics of the object being tracked whose dependencies can be analysed and learnt. Once a sequence of structures and their correlations are obtained, the built network can be used to predict positions and velocities or the general behaviour of the model. Experiments were undertaken using a video of a walking human, however the techniques presented here can be used for more general proposes such as recovering the behaviour of a group of robots whose actions have some coordination. The learning algorithm can be implemented online and can incorporate techniques to select samples - similar to that presented in Section 3.

One of the drawbacks of the proposed algorithm is that it is necessary to store the whole BN encoding the pattern. If the cycle of the motion is long, then the network will grow, possibly becoming intractable for exact inference algorithms. Alternatives to tackle this problem are non-linear regression methods that, by learning non-linear correlations, can incorporate sequences of motions in one structure.

Acknowledgements

This work is supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

References

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. New York: John Wiley & Sons, Inc.

Dempster, A. P., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM al-

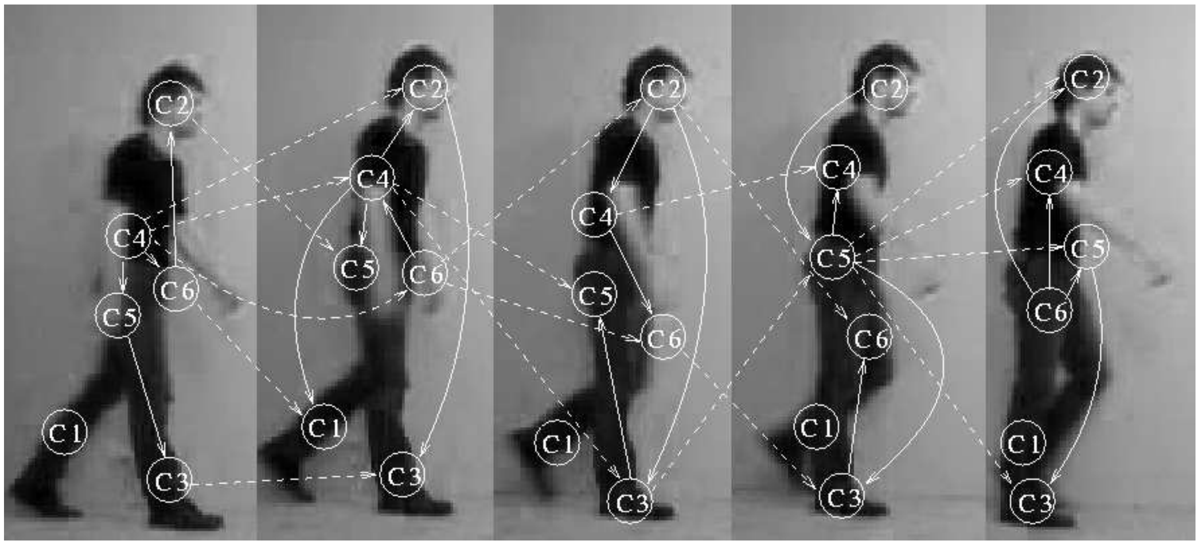


Figure 5. This picture shows the first five frames of a typical human gait and the learnt structures. The positions of the nodes - representing clusters - were calculated by taking the average of the labelled feature positions. Edges in the same frame are represented with solid lines while inter-frame correlations are represented with dash lines.

gorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*, 1–38.

Doucet, A., de Freitas, N., & Gordon, N. (2001). *Sequential Monte Carlo methods in practice.* New York: Springer-Verlag.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*, 131–163.

Heckerman, D. (1996). *A tutorial on learning with Bayesian networks* (Technical Report MSR-TR-95-06). Advanced Technology Division, Microsoft Corporation, Redmond, WA 98052.

Huang, C., & Darwiche, A. (1996). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, *15(3)*, 225–263.

Lauritzen, S. L. (1992). Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, *87*, 1098–1108.

Lauritzen, S. L. (1996). *Graphical models.* Oxford: Clarendon Press.

Lauritzen, S. L., & Jensen, F. (1999). *Stable local computation with conditional Gaussian distributions* (Technical Report R-99-2014). Department of Mathematical Sciences, Aalborg University, Aalborg, Denmark.

Lerner, U. N. (2002). *Hybrid Bayesian networks for reasoning about complex systems.* Doctoral dissertation, Department of Computer Science, Stanford University.

Murphy, K. P. (1998). *Inference and learning in hybrid Bayesian networks* (Technical Report UCB/CSD-98-990). Computer Science Division, University of California, Berkeley, CA 94720.

Murphy, K. P. (2002). *Dynamic Bayesian networks: Representation, inference and learning.* Doctoral dissertation, Computer Science Division, University of California, Berkeley.

Neal, R. M., & Hinton, G. E. (1993). A new view of the EM algorithm that justifies incremental and other variants. *Submitted to Biometrika.*

Song, Y., Goncalves, L., & Perona, P. (2003). Unsupervised learning of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 814–827.

Suzuki, J. (1998). Learning Bayesian belief networks based on the MDL principle: An efficient algorithm using the branch and bound technique. *IEICE Transactions on Information and Systems*, *E81-D*.

Tomasi, C., & Kanade, T. (1991). *Detection and tracking of point features* (Technical Report CMU-CS-91-132). School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

A Dynamic Programming Approach to Parameter Learning of Generative Models with Failure

Taisuke Sato SATO@MI.CS.TITECH.AC.JP
Yoshitaka Kameya KAMEYA@MI.CS.TITECH.AC.JP
Tokyo Institute of Technology / CREST

Abstract

We propose to apply dynamic programming to compute probabilities of failed goals for EM learning of generative models with failure described by symbolic-statistical modeling language PRISM. Programs for failed goals are synthesized deterministically by program transformation.

1. Introduction

The recent surge of interest in first-order statistical learning is motivated by the need of highly expressive language for modeling complex systems (De Raedt & Kersting, 2003). Logical variables and predicates in first-order logic together with probabilistic semantics enable us to define distributions over complex relational structures incorporating domain knowledge. PRISM[1] is a symbolic-statistical programming language designed for this purpose. It is a probabilistic extension of Prolog with a general EM learning routine based on formal semantics (Sato & Kameya, 1997; Sato & Kameya, 2001).

One of the most beneficial features of PRISM to the user is that parameter learning is for free. Distributions are defined by programs which consist of definite clauses and probabilistic built-in atoms. All we need for ML (maximum likelihood) estimation of parameters associated with probabilistic built-in atoms is just writing programs. The rest of the task is taken care of by the Prolog (tabled) search engine and the built-in general EM algorithm called *gEM* (graphical EM) algorithm (Kameya & Sato, 2000). Moreover thanks to dynamic programming nature of probability computation by the gEM algorithm, our EM learning is expected to be efficient. Actually adequately written PRISM programs for singly connected Bayesian networks, HMMs (hidden Markov models) and PCFGs (probabilistic context free grammars) are equivalent, time-complexity wise, to their specialized counterparts, i.e. EM learning by Pearl's belief propagation, the Baum-Welch algorithm and the Inside-Outside algorithm[2] respectively[3].

In PRISM semantics is well-defined for arbitrary programs. There is no restriction to Datalog programs or to range-restricted programs which are often imposed[4]. That being said however, we must add that EM learning is not without restrictions. PRISM excludes programs that do not satisfy '*no failure condition*' which states that computation must not eventually fail once probabilistic choices are made.

The objective of this paper is to remove this no failure condition thereby expanding the class of programmable models by PRISM to a wider class of models such as log-linear models. We allow programs to fail for example by conflicting constraints. Correspondingly when a program *DB* that may fail is given, we augment it with an auxiliary program that simulates failed computations caused by *DB* (sometime this is not possible though). The augmented program can be run on PRISM to infer parameters of *DB* by a new EM algorithm described in the **Appendix**.

Log-linear models cover a very wide class of probabilistic models and researchers have been seeking efficient EM algorithms (Abney, 1997; Riezler, 1998; Johnson

[1]URL = `http://sato-www.cs.titech.ac.jp/prism/`

[2]In the case of PCFGs, it is experimentally confirmed that the gEM algorithm runs faster than the Inside-Outside algorithm by orders of magnitude (Sato & Kameya, 2001).

[3]We also tested a variety of EM learning other than these popular models. The list includes Naive Bayes, linkage analysis, more sophisticated stochastic CFG grammars and stochastic graph grammars.

[4]Programs in which every variable in the head of a clause occurs in the body. Unit clauses must be ground.

et al., 1999) the latest one of which is the FAM algorithm, a specialized EM algorithm for SLPs (stochastic logic programs) proposed by Cussens (Cussens, 2001). He elegantly formulates the EM algorithm in the presence of failures by regarding observations as those with the failures truncated. FAM requires, however, to compute expected occurrences of clauses in the failed computations for which naive computation would obviously cause combinatorial explosion. Our approach opens a way to circumvent this problem by PRISM's dynamic programming.

Our contributions are as follows. We present a new EM algorithm which amalgamates the gEM algorithm and the FAM algorithm to perform ML estimation in a dynamic programming manner even in the presence of failed computations. An auxiliary PRISM program required by the new EM algorithm that simulates failed computations of the original program *DB* is synthesized from *DB* by deterministic program transformation. To our knowledge, this is the first attempt of program transformation in the logical-statistical setting. Also we present a novel class of constrained HMMs described by PRISM programs.

Our approach to probability computation is exact computation. We do not use sampling or other types of approximations. We take this approach because we believe that the dramatic increase of computation power nowadays makes exact computation more and more feasible and attractive as a means for fast probability computation. Also we would like to emphasize that our approach to modeling is generative. We describe a probabilistic model as a process that generates observable output like stochastic derivation of strings by PCFGs. Generative models are generally easy to understand and sampling is straightforward, but not necessarily outperform non-generative ones when we have poor knowledge of the generative mechanism of observations.

In the following, after providing preliminaries for logic programming and PRISM, we look at a motivating example in Section 3, and describe our compilation approach to failure in Section 4. We present the new EM algorithm in Section 5. Section 6 contains a modeling example of constrained HMM. Section 7 is the conclusion. The reader is assumed to be familiar with basics of logic programming (Sterling & Shapiro, 1986) and the EM algorithm (McLachlan & Krishnan, 1997).

2. Preliminaries

A logic program *DB* is a set of definite clause C of the form H `:-` $B_1, \ldots, B_n$ $(n \geq 0)$ where H (head) and B_i $(1 \leq i \leq n)$ (goal) are atoms. Every variable in C is universally quantified at the front of C. Clauses in *DB* are called program clauses. Hereafter we follow Prolog conventions and use strings beginning with upper case letters as variables[5]. So program clause `ancestor(X,Z) :- parent(X,Y), ancestor(Y,Z)` is read for example that for all `X`, `Y` and `Z` if `X` is a parent of `Y` and `Y` is an ancestor of `Z`, then `X` is an ancestor of `Z`. C is called a unit clause when $n = 0$. A clause or a term is said to be ground when no variable appears. The Herbrand universe (resp. the Herbrand base) of *DB* is the set of all ground terms (resp. ground atoms) whose function symbols (resp. function symbols and predicate symbols) occur in *DB*. A Herbrand interpretation is an assignment of truth values to each ground atom in the Herbrand base.

Computation by *DB* is nothing but search for a refutation of *DB* augmented with a query `:-` G such that $DB \vdash G\theta$ where θ is an answer substitution (variable bindings) for variables in G. Search is done by an SLD interpreter (SLD refutation procedure) which nondeterministically reduces a goal in a query to subgoals by a program clause. Define a set of ground atoms $I = \{A \mid DB \vdash A, A \,\text{ground atom}\}$. We identify I with a Herbrand interpretation M_{DB} such that $A \in I$ if-and-only if $M_{DB} \models A$. Then it is well-known that M_{DB} is a model of *DB*, i.e. satisfies every clause in *DB* and M_{DB} is the smallest as a set among Herbrand models of *DB*. M_{DB} is defined to be the denotation of *DB* as a program (*least model semantics*) (Doets, 1994).

PRISM generalizes this least model semantics probabilistically. A PRISM program DB' is the union of a set of definite clauses R and a set of ground atoms F representing probabilistic choices. DB' can be infinite. We give a probability measure P_F (*basic distribution*[6]) over the set of Herbrand interpretations of F and extends P_F to a probability measure $P_{DB'}$ over the set of Herbrand interpretations of DB' using R by way of least model semantics. We define $P_{DB'}$ as the denotation of DB' (*distribution semantics*) which regards ground atoms as binary random variables (Sato & Kameya, 2001). This extension is always possible for whatever P_F, but practical consideration restricts P_F to a (-n infinite) product of multinomial distributions.

Since distribution semantics is a generalization of least

[5]In Prolog, '`,`' (resp. '`;`') stands for conjunction (resp. disjunction) and '`=`' denotes unification.

[6]In this paper, we interchangeably use probability measure and probability distribution for the sake of familiarity.

model semantics, PRISM programs subsume logic programs. It also allows us to use arbitrary logic programs, arbitrary programming constructs like if-then, composition and recursion to define distributions and discrete stochastic processes. More importantly from a viewpoint of statistical learning, an EM algorithm is derived from this semantics for ML estimation of parameters specifying the multinomial distributions in P_F. The derived EM algorithm was general but too naive for real use. So we refined it to the gEM algorithm by incorporating the idea of dynamic programming (Kameya & Sato, 2000). The input of gEM is finite acyclic AND-OR graphs called *explanation (support) graph*s which encode statistical dependency among probabilistic atoms. An explanation graph for a goal G w.r.t. program DB' is obtained from tabled search for all refutations for `:-` G w.r.t. DB' by the SLD interpreter. Tabled search keeps the record of search in a table to prevent redundant search. PRISM adopts *linear tabling* as a tabling strategy (Zhou & Sato, 2003).

Here is an example of PRISM program. It defines a distribution over ground atoms of the form `bernoulli(`n`,`l`)` such that l is a list of outcomes of n coin tosses.

```
target(bernoulli,2).
values(coin,[heads,tails]).
:- set_sw(coin,0.6+0.4).

bernoulli(N,[R|Y]):-
    N>0,
    msw(coin,R),      % probabilistic choice
    N1 is N-1,
    bernoulli(N1,Y). % recursion
bernoulli(0,[]).
```

Figure 1. An example of PRISM program

Here `target(bernoulli,2)` is a declaration specifying the distribution of `bernoulli/2` atoms as a modeling target[7]. `values(coin,[heads,tails])` declares a discrete random variable named `coin` whose range is {`heads`,`tails`} which is implemented by atoms `msw(coin,`v`)` where v is either `heads` or `tails`[8].

`:- set_sw(coin,0.6+0.4)` is a directive on loading this program. It sets *parameter*s of `msw(coin,·)`, i.e. the probability of `msw(coin,heads)` to 0.6 and that of `msw(coin,tails)` to 0.4, respectively[9].

The next two clauses about `bernoulli/2` should be self-explanatory. They behave just like Prolog clauses except that `R` works as a random variable such that $P(\texttt{R} = \texttt{heads}) = 0.6$ and $P(\texttt{R} = \texttt{tails}) = 0.4$. The query `:- bernoulli(3,L)` will return for instance `L =[heads,heads,tails]`.

3. Loss of probability mass

PRISM programs fail just as Prolog programs do. Failure affects distributions and parameter learning, which can be seen by the following program.

```
target(test,1).
values(sw(0),[a,b,c]).
values(sw(1),[a,b,c]).
:- set_sw(sw0,0.5+0.3+0.2).
:- set_sw(sw1,0.5+0.3+0.2).

test(A):-
    msw(sw(0),A),
    msw(sw(1),B),
    A=B.          % A=B may fail
```

Figure 2. PRISM program causing failure

This program defines a distribution over `bernoulli/2`. It uses two probabilistic switches `sw(0)` and `sw(1)` to randomly choose one of {`a`,`b`,`c`}. They are i.i.d.s and represented by `msw` atoms whose parameters are set as $P(\texttt{msw(sw(0),a)}) = P(\texttt{msw(sw(1),a)}) = 0.5$, $P(\texttt{msw(sw(0),b)}) = P(\texttt{msw(sw(1),b)}) = 0.3$ and $P(\texttt{msw(sw(0),c)}) = P(\texttt{msw(sw(1),c)}) = 0.2$, respectively.

Suppose we call `:- test(X)` for sampling with `X` being variable. The clause head `test(A)` is unified and the leftmost goal `msw(sw(0),A)` is executed by randomly choosing `A`'s value from {`a`,`b`,`c`} according to probabilities set by `set_sw/2` directives. Suppose `a` is chosen. Next the second goal `msw(sw(1),B)` is executed similarly but independently. So it probabilistically happens that the sampled value of `B` is `b`. If this happens, the third goal `A=B`, the unification of `A` and `B`, fails and so does `:- test(X)`, which means we have no output, no observation despite sampling; we lost probability mass placed on `msw(sw(0),a)` and `msw(sw(0),b)`.

[7]`p/n` means that a predicate '`p`' has '`n`' arguments. We call an atom A `p` atom if the predicate symbol of A is `p`.

[8]`msw` atoms are most basic primitives in PRISM to make a probabilistic choice. They form constituents of the basic distribution and their probabilities are called *parameter*s.

[9]Parameters are inferred from observed goals consisting of `bernoulli/2` atoms by the gEM algorithm using `learn` command of PRISM.

Because failed computation yields no output, our observations should be interpreted as results of successful computations in our model. Consequently when we apply ML estimation to our observations, what must be maximized are conditional probabilities such as

$$P(\mathtt{test(a)} \mid \exists \mathtt{X}\, \mathtt{test(X)}) = P(\mathtt{test(a)})/P(\exists \mathtt{X}\, \mathtt{test(X)}).$$

$P(\exists \mathtt{X}\, \mathtt{test(X)})$ is the sum of probabilities of all successful computations for `:- test(X)` which is less than one[10]. In other words the current gEM algorithm is inapplicable as it merely maximizes unconditional probabilities of observations such as $P(\mathtt{test(a)})$.

Instead of the gEM algorithm however, we can use the FAM algorithm (Cussens, 2001) to infer parameters. Unfortunately it requires to compute *unnormalized* expected occurrence $\mathbf{E}_{\mathtt{fail}}[\eta(A)]$[11] of `msw` atoms A in all *failed* computations, which raises a question of how, generally, to capture all failed computations of a given program *DB*. We answer this question by synthesizing another program $DB^{\mathtt{fail}}$ that can simulate failed computations of *DB*. Since failed computations of *DB* are exactly successful computations of $DB^{\mathtt{fail}}$, $\mathbf{E}_{\mathtt{fail}}[\eta(A)]$ is computed from $DB^{\mathtt{fail}}$ efficiently by PRISM's dynamic programming. We next discuss how to obtain $DB^{\mathtt{fail}}$.

4. Simulating failed computation

4.1. Negation elimination in non-probabilistic case by First-Order Compiler

We propose to synthesize $DB^{\mathtt{fail}}$ by FOC (first-order compiler) (Sato, 1989). FOC is a deterministic program transformation algorithm that can compile negation, or more generally universally quantified implications[12] in a source program into an of executable logic program. Given a query `:- q(X)` and a logic program *DB* computing `q/1`, FOC eliminates negation automatically from *DB* ∪ { `failure :- not(exist([X],q(X)))` }. The resulting program $DB^{\mathtt{fail}}$ faithfully simulates failed computations caused by query `:- q(X)` given to *DB*.

Figure 3 is an example of negation elimination by FOC. As can be seen, '`not`' in the source program is

```
          *** source program ***

all_non_zero(L):- not(zero(L)).
                              % list L has no 0
zero(L):- mem(X,L),X=0.  % some X in L is 0
mem(X,[X|Y]).
mem(X,[H|Y]):-mem(X,Y).

         *** compiled program ***

all_non_zero(L):- closure_zero0(L,f0).
closure_zero0(L,C):-
   closure_mem0(L,f1(C)).
closure_mem0([],_).
closure_mem0([X|B],C):-
   cont(X,C),closure_mem0(B,C).
cont(X,f1(_)):- \+X=0.
```

Figure 3. Compilation example by FOC

compiled away and `all_non_zero(L)` in the compiled program computes `not(zero(L))` by definite clauses. The disunification `\+X=0` succeeds if-and-only-if the unification of `X` and `0` fails (negation-as-failure). The compiled program traces failed computations of the source program[13]. The (partial) correctness of FOC compilation is guaranteed by

Theorem 1 *(Sato, 1989) Suppose S, a source program is a set of clauses whose body include universally quantified implications, is compiled into S' by FOC. Then, iff(S)*[14] $\vdash A$ (*resp.* $\neg A$) *if* $S' \vdash A$ (*resp.* $\neg A$) *where A is a ground atom.*

This theorem roughly says that any computed result by the compiled program S' is a logical consequence of the source program S.

4.2. Probabilistic case

FOC was developed for non-probabilistic logic programs. Even when a program contains `msw` atoms and hence probabilistic however, compilation is possible.

[10]
$$\begin{aligned} & P(\exists \mathtt{X}\, \mathtt{test(X)}) \\ &= P(\mathtt{test(a)}) + P(\mathtt{test(b)}) + P(\mathtt{test(c)}) \\ &= P(\mathtt{msw(sw(0),a)}) \times P(\mathtt{msw(sw(1),a)}) \\ &\quad + P(\mathtt{msw(sw(0),b)}) \times P(\mathtt{msw(sw(1),b)}) \\ &\quad + P(\mathtt{msw(sw(0),c)}) \times P(\mathtt{msw(sw(1),c)}) \\ &= 0.38 < 1 \end{aligned}$$

[11] $\eta(A)$ is the number of occurrences of A. $\mathbf{E}_{\mathtt{fail}}[\eta(A)]$ is not a conditional expectation : $\mathbf{E}_{\mathtt{fail}}[\eta(A)] = \mathbf{E}[\eta(A) \mid \mathtt{fail}]P(\mathtt{fail})$ where `fail` denotes an occurrence of failure.

[12] Formulas of the form $\forall x(F \rightarrow G)$. Negation $\neg F$ is a special case because $\neg F$ is equivalent to $F \rightarrow \mathtt{false}$.

[13] `closure_mem0([],_)` for example simulates the unification failure of `mem(X,L)` in case of `L=[]` with `mem(·,[·|·])`, the head of a `mem` clause.

[14] iff(S) is the union of S and some additional formulas reflecting Prolog's top-down proof procedure (Doets, 1994).

Suppose for example that we are asked to compile an implication $(\mathtt{msw}(s,t) \rightarrow \phi)$ into an executable formula where s and t are terms and ϕ is some formula. Although `msw` is a probabilistic predicate, distribution semantics tells us to treat it as a normal predicate in compilation defined by some sampled atom, say $\mathtt{msw}(s,v)$. We therefore compile the above formula into $(\mathtt{msw}(s,\mathtt{W}),(\mathtt{W\backslash==}t;\ (\mathtt{W=}t,\phi)))$ where `W` is a new variable[15]. We can prove the correctness of this compilation under the condition that the compiled program terminates for any ground query regardless of sampling of msw atoms.

5. gEM for failure

5.1. Problem revisited

Suppose we write a PRISM program $DB_{\mathtt{q}}$ to define a distribution of a target predicate `q/1` which may fail[16]. Let $\mathtt{q}(s_1),\ldots,\mathtt{q}(s_T)$ be a random sample of length T. As pointed out in Section 3, if there is a loss of probability mass to failure, ML estimation should maximize $L(\boldsymbol{\theta}) = \prod_{t=1}^{T} P(\mathtt{q}(s_t) \mid \exists \mathtt{X}\,\mathtt{q}(\mathtt{X})) = \prod_{t=1}^{T} P(\mathtt{q}(s_t))/P(\exists \mathtt{X}\,\mathtt{q}(\mathtt{X}))$ where $\boldsymbol{\theta}$ collectively represents parameters to be estimated.

$L(\boldsymbol{\theta})$ can be maximized by the FAM algorithm (Cussens, 2001). It additionally computes in the E step, compared with non-failure case, $T * \mathbf{E}_{\mathtt{fail}}[\eta(A)]/P(\exists \mathtt{X}\,\mathtt{q}(\mathtt{X}))$ under the current $\boldsymbol{\theta}$ where $\mathbf{E}_{\mathtt{fail}}[\eta(A)]$ is the unnormalized expected occurrence of `msw` atom A in all failed computations for `:- q(X)` w.r.t. $DB_{\mathtt{q}}$. However the exact computation of $\mathbf{E}_{\mathtt{fail}}[\eta(A)]$ as well as that of $P(\exists \mathtt{X}\,\mathtt{q}(\mathtt{X}))$ is usually intractable if not impossible because of combinatorial explosion of computation paths. We solve this problem by using a combination of program transformation and dynamic programming

5.2. Augmentation with a compiled program for '`failure`'

We extend PRISM's dynamic programming approach to the computation of $\mathbf{E}_{\mathtt{fail}}[\eta(A)]$ by synthesizing a negation-free PRISM program using FOC that simulates the failed computations for `:- q(X)`.

Add a clause `failure :- not(exist([X],q(X)))` to $DB_{\mathtt{q}}$ and let the augmented program be $\overline{DB}_{\mathtt{q}}$. We remove negation from $\overline{DB}_{\mathtt{q}}$ by FOC as described in Section 4 to obtain $\overline{DB}_{\mathtt{q}}^{\mathtt{fail}}$. $\overline{DB}_{\mathtt{q}}^{\mathtt{fail}}$ is a normal PRISM program without negation and dynamic programming is applicable to compute $P(\mathtt{failure})$ and $\mathbf{E}_{\mathtt{succ}}[\eta(A)]$, the unnormalized expected occurrence of `msw` atom A in the *successful* computations for `:- failure` w.r.t. $\overline{DB}_{\mathtt{q}}^{\mathtt{fail}}$.

We henceforth assume that $DB_{\mathtt{q}}$ and $\overline{DB}_{\mathtt{q}}^{\mathtt{fail}}$ terminate with success or failure for query `:- q(X)` regardless of sampled values of `msw` atoms (*terminating condition*)[17].

We also assume that in a failed SLD derivation for `:- q(X)` w.r.t. $DB_{\mathtt{q}}$ and in a successful derivation for `:- failure` w.r.t. $\overline{DB}_{\mathtt{q}}^{\mathtt{fail}}$, a goal has multiple callees only when it is an `msw` atom[18].

Then we can prove with one more assumption not mentioned here that $\mathbf{E}_{\mathtt{fail}}[\eta(A)] = \mathbf{E}_{\mathtt{succ}}[\eta(A)]$ (details omitted). Also we have $P(\exists \mathtt{X}\,\mathtt{q}(\mathtt{X}) \mid \boldsymbol{\theta}) + P(\mathtt{failure} \mid \boldsymbol{\theta}) = 1$ for any $\boldsymbol{\theta}$. So we can replace $\mathbf{E}_{\mathtt{fail}}[\eta(A)]/P(\exists \mathtt{X}\,\mathtt{q}(\mathtt{X}))$ in the E step by $\mathbf{E}_{\mathtt{succ}}[\eta(A)]/(1 - P(\mathtt{failure}))$, which is computable from $\overline{DB}_{\mathtt{q}}^{\mathtt{fail}}$ alone, just by treating '`failure`' as a user-defined atom. We accordingly modify the gEM algorithm so that it additionally computes $\mathbf{E}_{\mathtt{succ}}[\eta(A)]/(1 - P(\mathtt{failure}))$ for every `msw` atom A in the E step.

5.3. New gEM algorithm

The modified new gEM algorithm is shown in the **Appendix**. Assuming appropriate conditions, it can perform EM learning in the presence of failure efficiently by dynamic programming. Modifications to the gEM algorithm are underlined. A brief explanation is in order (see (Sato & Kameya, 2001) for details of the gEM algorithm).

There $\overline{DB}^{\mathtt{fail}}$ is, as in the previous subsection, the original program DB augmented with compiled clauses for '`failure`' to simulate failed computations for the target predicate.

$\mathcal{G}' = G_0, G_1, \ldots, G_T$ is a list of observations $G_1, \ldots, G_T$ with special goal $G_0 = \mathtt{failure}$. Each G_t $(0 \leq t \leq T)$ generates by tabled search an hierarchical graph called an explanation graph which is represented as an ordered list $\widetilde{\psi}_{DB}(\tau_k^t) = \{\tau_1^t, \ldots, \tau_{K_t}^t\}$. Each τ_k^t $(1 \leq k \leq K_t)$ is a disjunction of $\widetilde{S}_{k,j}^t$ $(1 \leq j \leq m_k)$

[15]This is a simplified compilation assuming that $\mathtt{msw}(s,\cdot)$ atom is declared, i.e., the program includes a declaration of the form $\mathtt{value}(s',\cdot)$ such that s is an instance of s'.

[16]We use a unary predicate for explanatory purpose.

[17]Or equivalently an SLD tree for `:-q(X)` w.r.t. $\overline{DB}_{\mathtt{q}}^{\mathtt{fail}}$ is finite regardless of sampled values of `msw` atoms.

[18]This assumption implies that computation proceeds deterministically as far as non-probabilistic predicates are concerned.

such that $\widetilde{S}^t_{k,j}$ is a conjunction of `msw` atoms and probabilistic atoms called table atoms.

The main routine **learn-gEM(** $\overline{DB}^{\texttt{fail}}$,$\mathcal{G}'$) performs EM learning calling two subroutines, **get-inside-probs(**$\overline{DB}^{\texttt{fail}}$, $\mathcal{G}'$) to compute inside probabilities and **get-expectations(** $\overline{DB}^{\texttt{fail}}$,$\mathcal{G}'$) to compute outside probabilities. Four arrays are used to store data, $\mathcal{P}[t,\tau]$ for the inside probability $P_{DB}(\tau \mid \boldsymbol{\theta})$, $\mathcal{Q}[t,\tau]$ for outside probability of τ w.r.t. G_t, $\mathcal{R}[t,\tau,\widetilde{S}]$ for $P_{DB}(\widetilde{S} \mid \boldsymbol{\theta})$ and finally $\eta[t,i,v]$ for the expected occurrence of `msw(`i`,`v`)` in a refutation for `:-` G_t w.r.t. DB.

Learning terminates when an increase of the log-likelihood of conditional probabilities is less than a given threshold ϵ. The time complexity in one iteration and the space complexity of the new gEM algorithm are linear in the size of the total size of the explanation graphs.

6. A learning example: constrained HMM

Here we give a learning example of generative model with failure. The example is small so that the reader can have an overview of our approach at a glance[19]. We introduce a class of HMM models with equality and disequality constraints (inequality constraints are treated similarly). For simplicity we describe an HMM model with a single constraint such that the first output alphabet and the last output alphabet must be identical. This model has five parameters. It seems difficult to express it succinctly by a PCFG with such a small number of parameters.

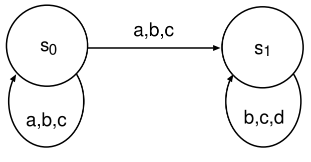


Figure 4. Two state HMM with constraint

Suppose here is a HMM which has two states $\{s_0, s_1\}$ and at s_0 one of $\{a,b,c\}$ is emitted and at s_1 one of $\{b,c,d\}$ is emitted, probabilistically. We place one constraint stated as above on this model.

A program $\overline{DB}_{\rm hmm}$ in Figure 5 specifies our model procedurally. It includes a `failure` clause saying that we have no output string in spite of probabilistic choices made by `msw(obs(`$\cdot$`),`$\cdot$`)` (for alphabet emission) and `msw(tr(`$\cdot$`),`$\cdot$`)` (for state transition).

[19]We have applied our framework to PCFGs and found that EM learning is possible in polynomial time (and space) complexity in some cases. We also have conducted a learning experiment with a PCFG using a real corpus of moderate size, which will be reported elsewhere.

```
failure:-          % there is no output
  not(exist([X],hmm(X))).
string_length(5).  % output length is 5
hmm(Cs):-
  string_length(N),
  msw(obs(s0),C1), % s0 is an initial state
  N1 is N-1,       % C1 is a 1st alphabet
  Cs=[C1|Rest],    % keep C1 till end
  hmm(N1,s0,Rest,C1).
hmm(N,State,[C|Cs],C1):-
  N>1,
  msw(obs(State),C),
  ( State == s0, msw(tr(State),NextS)
  ; State == s1, NextS = s1 ),
  N1 is N-1,
  hmm(N1,NextS,Cs,C1).
hmm(N,State,[C1],C1):-
  N==1,  % the last alphabet must be C1
  msw(obs(State),C1).
```

Figure 5. A constrained HMM program $\overline{DB}_{\rm hmm}$

The subsequent clauses about `hmm/1` and `hmm/4` describe how state transition and alphabet emission at each state are made. For `:- hmm(Cs)`, we probabilistically choose an initial alphabet `C1` by executing `msw(obs(s0),C1)` and enter into recursion by `hmm/4` until the length of a generated string reaches the specified number, 5 in this case.

$\overline{DB}_{\rm hmm}$ is not runnable directly due to 'not' in the `failure` clause. We remove it by FOC and obtain a PRISM program $\overline{DB}^{\texttt{fail}}_{\rm hmm}$. We show part of it in Figure 6 concerning the computation of `failure` where '`==`' (resp. '`\==`') is a built-in predicate for strict equality (resp. strict disequality).

By inspection, we know that the search terminates at most in 5 steps. We also notice that the compiled program is tail recursive on `closure_hmm0/4`. So all refutation search for `:- failure` w.r.t $\overline{DB}^{\texttt{fail}}_{\rm hmm}$ by tabled search generates an explanation graph with trellis structure to which dynamic programming is effectively applicable. The time complexity of probability computations (that of one iteration in EM learning) by $\overline{DB}^{\texttt{fail}}_{\rm hmm}$, is $O(m^2 * n)$ where m is the number of states and n the length of input string, contrary to the exponential order caused by the naive non-dynamic programming approach. Also $O(m^2 * n)$ is equal to the

```
target(failure,0).
values(tr(s0),[s0,s1]).
values(obs(s0),[a,b,c]).
values(obs(s1),[b,c,d]).
:- set_sw(obs(s0),0.2+0.4+0.4).
:- set_sw(obs(s1),0.4+0.4+0.2).
:- set_sw(tr(s0),0.7+0.3).

failure:- closure_hmm0(f0).
closure_hmm0(A):-
   closure_string_length0(f2(A)).
closure_string_length0(A):- cont(5,A).
cont(A,f2(B)):-
   msw(obs(s0),C),
   D is A-1,
   closure_hmm0(D,s0,C,B).
closure_hmm0(A,B,C,D):-
  ( A>1, msw(obs(B),_),
     ( B\==s0
     ; B==s0, msw(tr(B),E), F is A-1,
         closure_hmm0(F,E,C,D) ),
     ( B\==s1
     ; B ==s1, G is A-1,
         closure_hmm0(G,s1,C,D) )
  ; A=<1 ),
  ( A\==1 ; A ==1, msw(obs(B),H), \+H=C ).
```

Figure 6. The compiled program $\overline{DB}^{\texttt{fail}}_{\text{hmm}}$ (part)

space complexity as it is the size of the explanation graphs for the failed and successful computations.

We computed P(`failure`) with parameters set by `set_sw` in $\overline{DB}^{\texttt{fail}}_{\text{hmm}}$ and obtained the following value[20].

```
?- prob(failure).
...
The probability of failure is: 0.66628
```

We also conducted a learning experiment with $\overline{DB}^{\texttt{fail}}_{\text{hmm}}$ using the new gEM algorithm in the **Appendix**. In each trial we randomly sampled 10,000 data by running `hmm/1` with the original parameters indicated in Figure 6 and then let $\overline{DB}^{\texttt{fail}}_{\text{hmm}}$ learn parameters from the generated data by using the new gEM algorithm with randomized initial values. We show in Figure 7 averages (with standard deviations) of parameters learned from 10 trials (threshold is 10^{-4}). For comparison, we add the averages of learned parameters by the gEM algorithm under the same condition.

[20]`prob(G)` is a PRISM built-in to compute the probability of a goal `G`.

msw name	learned parameter (ave. of 10 trials)		
`obs(s0)`	`a`	`b`	`c`
original	0.2	0.4	0.4
new gEM	$0.196(9.0 \times 10^{-4})$	$0.409(1.2 \times 10^{-4})$	$0.394(6.8 \times 10^{-4})$
gEM	$0.142(1.9 \times 10^{-4})$	$0.437(4.7 \times 10^{-4})$	$0.421(5.5 \times 10^{-4})$
`obs(s1)`	`b`	`c`	`d`
original	0.4	0.4	0.2
new gEM	$0.390(1.6 \times 10^{-3})$	$0.396(2.7 \times 10^{-3})$	$0.214(1.7 \times 10^{-3})$
gEM	$0.444(7.8 \times 10^{-4})$	$0.435(5.8 \times 10^{-4})$	$0.120(6.0 \times 10^{-4})$
`tr(s0)`	`s0`	`s1`	
original	0.7	0.3	
new gEM	$0.688(2.1 \times 10^{-3})$	$0.312(2.1 \times 10^{-3})$	
gEM	$0.713(1.2 \times 10^{-3})$	$0.286(1.2 \times 10^{-3})$	

Figure 7. Learned parameters

This table reads for instance P(`msw(obs(s0),a)`) = 0.196 with standard deviation 9.0×10^{-4} is obtained by the new gEM algorithm. We see that the new gEM algorithm infers better parameters than the gEM algorithm.

7. Conclusion

We have proposed a new EM algorithm applicable to generative models with failure described by PRISM programs. It is an amalgamation of two EM algorithms, one the gEM algorithm which is based on dynamic programming (Kameya & Sato, 2000; Sato & Kameya, 2001) and the other the FAM algorithm which considers failure in refutation search for SLPs (Cussens, 2001).

To realize this amalgamation, we also proposed to apply FOC (first-order compiler) (Sato, 1989) to PRISM programs containing '`not`,' logical negation, to synthesize a program which is able to simulate failed computations by the original program.

We also explained how universally quantified logical formula with probabilistic built-ins are deterministically transformed to executable codes in PRISM while preserving distribution semantics.

References

Abney, S. (1997). Stochastic attribute-value grammars. *Computational Linguistics*, *23*, 597–618.

Cussens, J. (2001). Parameter estimation in stochastic logic programs. *Machine Learning*, *44*, 245–271.

De Raedt, L., & Kersting, K. (2003). Probabilistic logic learning. *ACM-SIGKDD Explorations, special issue on Multi-Relational Data Mining*, *5*, 31–48.

Doets, K. (1994). *From logic to logic programming.* The MIT Press.

Johnson, M., Geman, S., Canon, S., Chi, Z., & Riezler, S. (1999). Estimators for stochastic unification-based grammars. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)* (pp. 535–541).

Kameya, Y., & Sato, T. (2000). Efficient EM learning for parameterized logic programs. *Proceedings of the 1st Conference on Computational Logic (CL2000)* (pp. 269–294). Springer.

McLachlan, G. J., & Krishnan, T. (1997). *The EM algorithm and extensions.* Wiley Interscience.

Riezler, S. (1998). *Probabilistic constraint logic programming.* Doctoral dissertation, Universität Tübingen.

Sato, T. (1989). First order compiler: A deterministic logic program synthesis algorithm. *Journal of Symbolic Computation*, *8*, 605–627.

Sato, T., & Kameya, Y. (1997). PRISM: a language for symbolic-statistical modeling. *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)* (pp. 1330–1335).

Sato, T., & Kameya, Y. (2001). Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, *15*, 391–454.

Sterling, L., & Shapiro, E. (1986). *The art of prolog.* The MIT Press.

Zhou, N.-F., & Sato, T. (2003). Efficient Fixpoint Computation in Linear Tabling. *Proceedings of the Fifth ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP2003)* (pp. 275–283).

Appendix: gEM for failure

The followings are the procedures of the new graphical EM algorithm for failure.

```
procedure learn-gEM( $\overline{DB}^{\mathtt{fail}}$,$\mathcal{G}'$)
begin
  Select some $\boldsymbol{\theta}$ as initial parameters;
  get-inside-probs($\overline{DB}^{\mathtt{fail}}$, $\mathcal{G}'$);
  $\lambda^{(0)} := \underline{\sum_{t=1}^{T} \ln\left(\mathcal{P}[t, G_t]/(1 - \mathcal{P}[0, G_0])\right)}$;
  repeat
    get-expectations($\overline{DB}^{\mathtt{fail}}$, $\mathcal{G}'$);
    foreach $i \in I, v \in V_i$ do
      $\eta[i, v] := \underline{T * \eta[0, i, v]/(1 - \mathcal{P}[0, G_0])}$
                    $+ \sum_{t=1}^{T} \eta[t, i, v]/\mathcal{P}[t, G_t]$;
    foreach $i \in I, v \in V_i$ do
      $\theta_{i,v} := \eta[i, v] / \sum_{v' \in V_i} \eta[i, v']$;
    get-inside-probs($\overline{DB}^{\mathtt{fail}}$, $\mathcal{G}'$);
    $m := m + 1$;
    $\lambda^{(m)} := \underline{\sum_{t=1}^{T} \ln\left(\mathcal{P}[t, G_t]/(1 - \mathcal{P}[0, G_0])\right)}$;
  until $\lambda^{(m)} - \lambda^{(m-1)} < \varepsilon$
end.
```

```
procedure get-inside-probs($\overline{DB}^{\mathtt{fail}}$, $\mathcal{G}'$)
begin
  for $t := 0$ to $T$ do begin
    Let $\tau_0^t = G_t$;
    for $k := K_t$ downto $0$ do begin
      $\mathcal{P}[t, \tau_k^t] := 0$;
      foreach $\widetilde{S} \in \widetilde{\psi}_{DB}(\tau_k^t)$ do begin
        Let $\widetilde{S} = \{A_1, A_2, \ldots, A_{|\widetilde{S}|}\}$;
        $\mathcal{R}[t, \tau_k^t, \widetilde{S}] := 1$;
        for $l := 1$ to $|\widetilde{S}|$ do
          if $A_l = \mathtt{msw}(i, \cdot, v)$ then
            $\mathcal{R}[t, \tau_k^t, \widetilde{S}] *= \theta_{i,v}$
          else $\mathcal{R}[t, \tau_k^t, \widetilde{S}] *= \mathcal{P}[t, A_l]$;
        $\mathcal{P}[t, \tau_k^t] += \mathcal{R}[t, \tau_k^t, \widetilde{S}]$
      end /* foreach $\widetilde{S}$ */
    end /* for $k$ */
  end /* for $t$ */
end.
```

```
procedure get-expectations($\overline{DB}^{\mathtt{fail}}$, $\mathcal{G}'$)
begin
  for $t := 0$ to $T$ do  begin
    foreach $i \in I, v \in V_i$ do $\eta[t, i, v] := 0$;
    Let $\tau_0^t = G_t$; $\mathcal{Q}[t, \tau_0^t] := 1.0$;
    for $k := 1$ to $K_t$ do $\mathcal{Q}[t, \tau_k^t] := 0$;
    for $k := 0$ to $K_t$ do
      foreach $\widetilde{S} \in \widetilde{\psi}_{DB}(\tau_k^t)$  do begin
        Let $\widetilde{S} = \{A_1, A_2, \ldots, A_{|\widetilde{S}|}\}$;
        for $l := 1$ to $|\widetilde{S}|$ do
          if $A_l = \mathtt{msw}(i, \cdot, v)$ then
            $\eta[t, i, v] += \mathcal{Q}[t, \tau_k^t] \cdot \mathcal{R}[t, \tau_k^t, \widetilde{S}]$
          else
            $\mathcal{Q}[t, A_l] += \mathcal{Q}[t, \tau_k^t] \cdot \mathcal{R}[t, \tau_k^t, \widetilde{S}]/\mathcal{P}[t, A_l]$
      end /*  foreach $\widetilde{S}$ */
  end /*  for $t$ */
end.
```

Cluster-based Concept Invention for Statistical Relational Learning

Alexandrin Popescul POPESCUL@CIS.UPENN.EDU
Lyle H. Ungar UNGAR@CIS.UPENN.EDU
Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104

Abstract

We use clustering to derive new relations which augment database schema used in automatic generation of predictive features in statistical relational learning. Clustering improves scalability through dimensionality reduction. More importantly, entities derived from clusters increase the expressivity of feature spaces by creating new first-class concepts which contribute to the creation of new features. For example, in CiteSeer, papers can be clustered based on words or citations giving "topics", and authors can be clustered based on documents they co-author giving "communities". Such cluster-derived concepts become part of more complex feature expressions. Out of the large number of generated features, those which improve predictive accuracy are kept in the model, as decided by statistical feature selection criteria. We present results demonstrating improved accuracy and scalability when predicting publication venues using CiteSeer data.

1. Introduction

Statistical relational learning and related methods search a space of database queries or logic expressions to find those which generate new predictive features. A given schema, describing background data, is used to structure a search over database queries. Each query generates a table, which in turn is aggregated to produce scalar feature candidates. The process produces a stream of features, from which statistically significant predictors are selected. The expressivity of the generated features is determined by the set of central relational entities participating in the search.

Considerably more powerful models can be built when the original schema is augmented with new relations which are derived via clustering (*cluster-relations*). Clustering can be used to create first-class relational concepts which are not derivable otherwise from the original relations. The addition of cluster-relations to the schema results in the creation of richer, more expressive, feature spaces, resulting in more accurate models than those built from the original relational concepts. Perhaps surprisingly, this approach can also lead to the more rapid discovery of predictive features. In addition to summarizing information (e.g. "Is this document on a given topic?"), cluster derived concepts participate in more complex relationships (e.g., "Does the database contain another document on the same topic and published in the same conference?"). The creation of these new high-level concepts allows more accurate and robust modeling from complex data sources not simply through information reduction, but, more importantly, through the increased expressivity of the language used to describe patterns in the data (0).

2. Methodology

We use a form of statistical relational learning which integrates regression with feature generation from relational data. In this paper we use logistic regression, giving a method we call Structural Logistic Regression (SLR). SLR combines the strengths of classical statistical modeling with the high expressivity of features automatically generated from a relational database.

Cluster-relations enter the formulation of the search space used to generate predictive features exactly as the original relations. The original database schema is used to decide which entities to cluster and what sources of attributes to use, for example documents clustered by words or by citations create alternative clusterings of the same objects. Once the schema is expanded by adding derived cluster relations to it, the underlying statistical relational learning methodology is repeated, i.e. database queries of the feature generation search space are evaluated, and the resulting tables per observation are aggregated to produce scalar feature columns, Figure 1. The new relations added are treated exactly the same as the original relations.

Section 2.1 briefly presents SLR; the reader is referred to (0) for a more detailed description.

2.1. Structural Logistic Regression

SLR is an extension of logistic regression to modeling relational data. It combines the strengths of classical statistical models with the higher expressivity of features automatically generated from a relational database. SLR dynamically couples two main components: generation of feature candidates from relational data and their selection using statistical model selection criteria. Relational feature generation is a search problem. It requires formulation of the search in the space of, possibly complex, queries to a relational database. At each search node, feature candidates are constructed and considered for model inclusion. Thus, the process incrementally learns predictive data patterns, possibly encoding complex regularities in a domain. The process results in a statistical model where each selected feature is the evaluation of a database query encoding a predictive data pattern.

As mentioned above, relational feature generation is a search problem. We use top-down search of refinement graphs (**?**; **?**) as our main search space specification method. Each node in the refinement graph is a database query. The search starts with simpler queries about learning examples and progresses by refining its nodes, i.e. adding more relation instances and conditions to a parent query. Since we are building statistical models, rather than logic clauses as is the case in inductive logic programming where refinement graphs are used, we are not limited to searching in the space of binary logic-valued clauses. In our case, each node of the graph is a query evaluating into a table of all satisfying solutions. Within each node we apply a number of aggregate operators to produce both boolean and real-valued features. Thus, each node of the refinement graph produces multiple feature candidates. Although there is no limit to the number of aggregate operators one may try, e.g. square root of the sum of column values or logarithm of their product, we find *count*, *ave*, *max*, *min*, and *empty* to be particularly useful. Aggregations can be applied to a whole table or to individual columns, as appropriate given type restrictions, e.g. *ave* cannot be applied to a column of a categorical type.

The use of aggregate operators in feature generation makes pruning of the search space more involved. Currently, we use a hash function of partially evaluated feature columns to avoid fully recomputing equivalent features. In general, determining equivalence among relational expressions is known to be NP-complete. Polynomial algorithms exist for restricted classes of expressions, e.g. (**?**; **?**). Equivalence determination based on the homomorphism theorem for *tableau* query formalism, essentially the class of conjunctive queries we look at before aggregation, is explained in detail in (0), page 115. However, deciding the equivalence of two arbitrary queries is different from the simpler problem we face of avoiding duplicates when we have control over the way we structure the search space.

Top-down search of refinement graphs allows a number of optimizations, e.g. i) the results of queries (prior to applying the aggregations) at a parent node can be reused at the children nodes, ii) a node resulting in an empty table for each observation should not be refined any further as its refinements will also be empty.

3. Task and Data

We explore the task of classifying CiteSeer documents into their publication venues, conferences or journals. The target concept pair is `<Document, Venue>`. The value of the response variable is one if the pair's venue is a true publication venue of the corresponding document and, it is zero otherwise. The search space contains queries based on several relations about documents and publication venues, such as citation information, authorship and word content of the documents. Modeling of latent structure of entities in this domain, such as topics of documents or communities of authors, is capable of producing more accurate predictive models that the original relational representation. Clusters can be derived by clustering entities in the domain based on the variety of alternative sources of attributes.

Publication venues were extracted by matching information with the DBLP database, `http://dblp.uni-trier.de/`. Publication venues are known for 60,646 CiteSeer documents.We use these documents and information about them in the experiments described below. Table XXX shows the basic relations we use,

- `PublishedIn(`*doc*`:Document, `*vn*`:Venue)`. 60,646 (60,646, 1,560)
- `Author(`*doc*`:Document, `*auth*`:Person)`. 131,582 (53,660, 26,740)
- `Citation(`*from*`:Document, `*to*``:Document)`. There are 173,410(42,749,31,603)
- `HasWord(`*doc*`:Document, `*word*`:Word`. 6,894,712 (56,104, 1,000) (The vocabulary was limited to

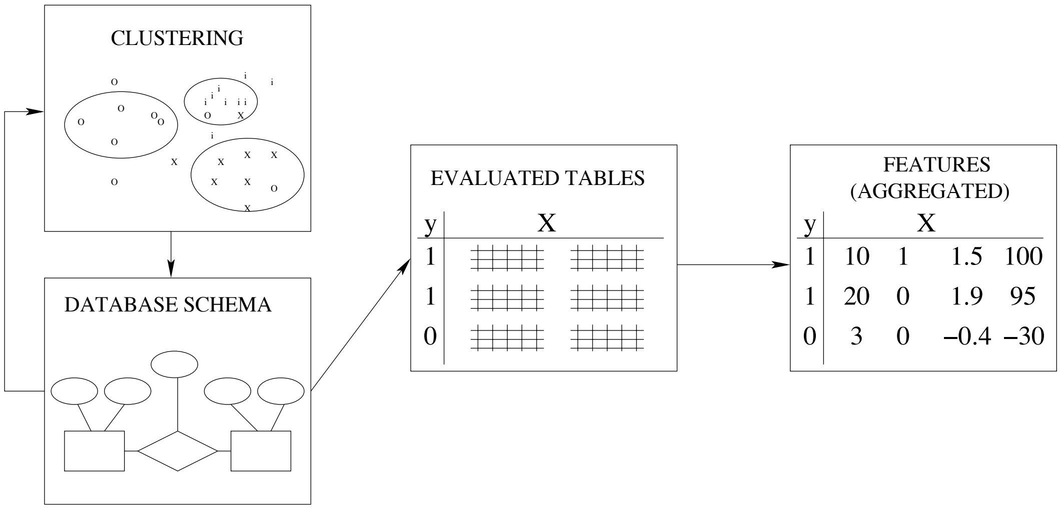


Figure 1. Cluster-relations augment database schema used to produce feature candidates.

the top 1,000 count words in the collection after Porter stemming and stop word removal).

We use k-means to derive cluster relations; any other hard clustering algorithm can be used for this purpose. The results of clustering are represented by binary relations
`<ClusteredEntity, ClusterID>`.

The original database schema contains several entities which can be clustered based on a number of alternative criteria. Each many-to-many relation in the original schema presented above can produce two cluster relations. Three out of four relations are many-to-many (with the exception of `PublishedIn`), this results in six new cluster-relations. The following is the list of these six cluster relations which we add to the relational database schema:[1]

- `Author(`*doc*`:Document,`*auth*`:Person)` produces:

  `ClusterDocsByAuthors(`*doc*`:Document,`*clust*`:Clust0)` 53,660 documents are clustered based on the identity of their 26,740 authors.

  `ClusterAuthorsByDocs(`*auth*`:Person,`*clust*`:Clust1)` 26,740 authors are clustered based on 53,660 documents they wrote.

- `Citation(`*from*`:Document,`*to*`:Document)` produces:

  `ClusterDocsByCitingDocs(`*doc*`:Document,`*clust*`:Clust2)` 31,603 documents are clustered based on 42,749 documents citing them.

  `ClusterDocsByCitedDocs(`*doc*`:Document,`*clust*`:Clust3)` 42,749 documents are clustered based on 31,603 documents cited from them.

- `HasWord(`*doc*`:Document,`*word*`:Word` produces:

  `ClusterDocsByWords(`*doc*`:Document,`*clust*`:Clust4)` 56,104 documents are clustered based on the vocabulary of top 1,000 words they contain.

  `ClusterWordsByDocs(`*word*`:Word,`*clust*`:Clust5)` The vocabulary of 1,000 words is clustered based on their occurrence in this collection of 56,104 documents.

Throughout the experiments in this paper we use k-means clustering algorithm (e.g. (**?**)) with vector-space cosine similarity (0), a widely used similarity measure in text analysis. The search can be extended to include more similarity measures and a search over k, the number of groups in clustering. These simply results in more features tested in the regression. If needed, on-the-fly optimization using subsampling and efficient linear time clustering algorithm could be used, but in this paper we did not find them necessary.

An important aspect of optimizing cluster utility in general, and of the use of cluster relations in our setting in particular, is the choice of k, the number of groups into which the entities are clustered. In our case, for each potential value of k we would ideally

[1]See Future Work section for the discussion of how the expressivity of the cluster types can be further increased when attribute vectors of clustered entities are derived from compound relationships.

compute separate clusters. For simplicity and speed in the experiments presented here we fix k to be equal to 100 in all cluster relations except for the last one, `ClusterWordsByDocs`, where the number of clusters is 10. The latter is clustered into fewer groups because there is roughly an order of magnitude fewer objects, words, to be clustered; we selected the vocabulary of size 1,000 to make the size of `HasWord` relation smaller and more manageable.

4. Results

The results presented below demonstrate the advantages of including cluster relations in the statistical relational learning framework. First, including cluster relations in the search space increases the expressivity of the feature space and achieves higher classification accuracy. Second, cluster-based features are cheaper to generate since cluster relations contain fewer tuples than the original relations from which they were derived.

The training set contains 1,000 observations: 500 positive examples of `<Document, Venue>` target pairs uniformly sampled from the `PublishedIn` relation, and 500 negative examples where document is uniformly sampled from the remaining documents and the venue is uniformly sampled from the domain of all venues, such that the sampled venue is not a true venue of the corresponding document. Sampled positive pairs are removed from the background relation `PublishedIn`, as well as the tuples involving documents sampled for the negative set. The test set contains 2,000 examples: 1,000 positive and 1,000 negative, sampled and removed from `PublishedIn` in the same manner as the training set examples. We know the citation structure, authorship and content of the documents for which we are learning to predict publication venues.

To demonstrate the utility of using cluster relations within our statistical relational learning framework we compare two models. One model is learned from the feature space generated from four original non-cluster relations, `PublishedIn`, `Author`, `Citation` and `HasWord`. The other model is learned from the original four relations plus six derived cluster relations, `DocsByAuthors`, `AuthorsByDocs`, `DocsByCitingDocs`, `DocsByCitedDocs`, `DocsByWords` and `WordsByDocs`. Models are learned with sequential BIC feature selection, i.e. as each feature is generated it is added to the model permanently if the BIC statistic improves, or is permanently dropped otherwise. Sequential feature selection is different from standard step-wise model selection. Standard step-wise model selection is infeasible within this framework because it is not known in

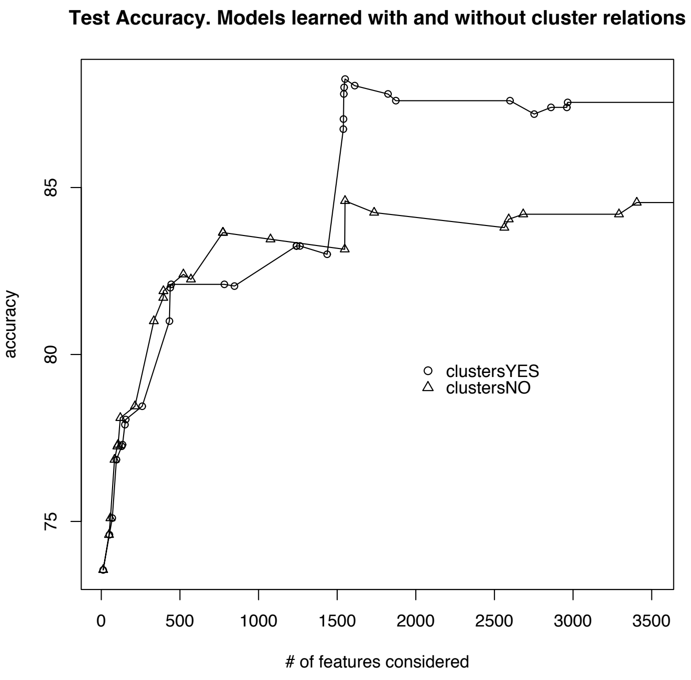


Figure 2. Learning curves show test set (N_{test}=2,000) accuracy changing with the number of features generated and selected from the training set (N_{train}=1,000). Balanced positive/negative priors.

Table 1. Six features in each model which improve test accuracy by at least 1.0 percentage point after being selected. Target: $publishedIn^{'}(D,V)$

Feature	**Model**
$size[publishedIn(\_,V)]$	both
$exists[citation(D,D1), publishedIn(D1,V)]$	both
$exists[citation(D1,D), publishedIn(D1,V)]$	both
$exists[citation(D,D2), citation(D1,D2), publishedIn(D1,V)]$	both
$exists[author(D,A), author(D1,A), publishedIn(D1,V)]$	both
$exists[citation(D,D3), citation(D3,D2), citation(D1,D2), publishedIn(D1,V)]$	`clusterNO`
$exists[publishedIn(D1,V), docsByWords(D,C), docsByWords(D1,C)]$	`clusterYES`

advance which features will be generated next in the feature stream; it is also computationally much more demanding for very large feature streams. Step-wise feature selection needs to check all available feature candidates before adding/dropping one; sequential feature selection re-trains only one additional model per one generated feature.

We learn two models (`clusterNO` and `clusterYES`) using sequential feature selection from the feature streams of size 3,500 of numerically unique features each. A numeric signature of partially evaluated features is maintained to avoid fully generating numerically equivalent (or rather, at least, nearly collinear within hashing error bound) features; note that this is different from avoiding syntactically equivalent nodes of the search space: two different queries can produce numerically equivalent feature columns, e.g. all zeros, which is a common case as feature generation progresses deeper in the search space.

Figure 2 presents learning curves for the two models, learned with and without cluster relations. The curves show test set accuracy changing with the number of features generated and sequentially selected from the training set. The x-axis is the number of features generated from the background relations; points on the curves correspond to the selected features. The model with clusters, `clusterYES`, contains 31 features at the end of the process; 24 features are selected into `clusterNO` from equally many feature candidates (3,500). The number of positive and negative examples is equal in both cases; the curves start from the first added non-intercept feature. Empty models, i.e. only with an intercept, would be 50% accurate. The test set accuracy of the cluster based model after exploring the entire feature stream is 87.55%, which is 3.0 percentage points higher than the accuracy of the model not using cluster relations. The figure suggests that there is no significant overfitting, represented by a decrease in test accuracy when adding a new feature. The largest single drop in accuracy after adding next feature is 0.45 percentage points. Thus, the `clusterYES` model achieves a significant improvement in accuracy and with fewer resources, as the generation of the same number of features for the cluster-based model is cheaper, as the size of cluster relations is smaller that the size of the original relations from which they were derived.

Not all of the cluster relations are equally useful. As Figure 2 shows, the improved accuracy of the cluster-based model comes mostly from a single cluster-based feature, 1540-th in the stream, which made test set accuracy jump by 3.75 percentage points. This feature is a binary feature involving latent document topics, i.e. the cluster relation of documents clustered by their word content. The feature is *ON* for target document/venue pair `<D,V>`, if there exists a document `D1` in the cluster where `D` belongs such that `D1` is published in the same venue as `D`.

Table 1 shows the most significant features learned. *Post factum*, these features are fairly obvious, which is good news for the purposes of validating the methodology and its potential application to less well understood domains. The last item in the table shows the most important cluster-based feature, which translates to English as "if there exists another document which is on the same latent topic as `D` and is published in `V`."

4.1. Cluster-First Search

The cost of database query evaluation used in feature generation dominates the complexity of the SLR methodology. Up to this point we presented models learned when searching the feature space in breadth-first manner. In this section we explore an alternative search strategy which places cluster-based features earlier in the stream. In our venue prediction setting, this strategy achieves the same accuracy as the breadth-first `clusterYES` model with far fewer features tested.

To test the cluster-first search strategy we split the

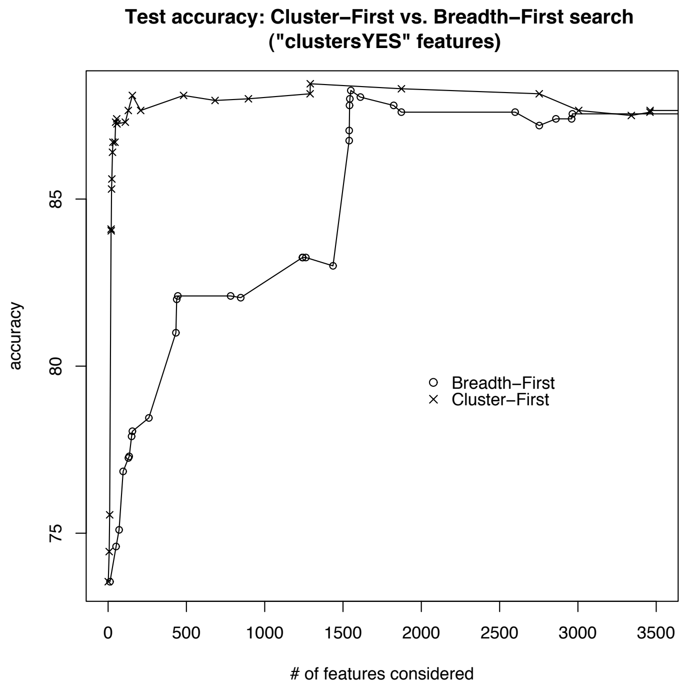


Figure 3. Learning curves show test set (N_{test}=2,000) accuracy changing with the number of features generated and selected from the `clusterYES` training set (N_{train}=1,000) following two search strategies: breadth-first and cluster-first. The latter pushes cluster-based features earlier in the stream. Balanced positive/negative priors.

stream of 3,500 features generated by breadth-first traversal of the `clusterYES` search space into two consecutive substreams. The features of the first "cluster" substream are presented for statistical feature selection first, followed by the features from the second substream. The features in each substream appear in the same relative order as in the original breadth-first stream.

The first stream includes features which involve only cluster-relations and the `PublishedIn` relation. The second stream includes all other features, some of which are based only on `Citation`, `Author` and `HasWord` relations, and others involve cluster relations and `PublishedIn` relation *together* with `Citation`, `Author` and `HasWord` relations. The `PublishedIn` relation is pushed earlier in the stream together with cluster-based relations because of its special status - this relation serves as a *structural core* background relation being of the same type as the response concept. It provides background reference essential for learning, similarly to the role of the `Citation` relation when learning models for link prediction (0; **?**). Because it is not a many-to-many relation there are no "proxy" cluster-relations derived from `PublishedIn`.

Figure 3 presents learning curves for two search strategies, and show test set accuracy a function of the number of features generated and tested in the model.

5. Related Work and Discussion

Clustering and other latent space modeling methods such as PCA often used in propositional predictive modeling as a means for dimensionality reduction. Dimensionality reduction is achieved by replacing the original flat features with the identifiers of clusters they are elements of, or by the coordinates of their projections onto a lower dimensional space. For example, words can be clustered into groups which replace individual words for document classification. structure together with the flat features resulting in more accurate predictive models, for example in the context of maximum entropy modeling (0).

One research direction in relational learning addresses clustering of relational entities with novel distance metrics defined over the interlinked relational representation. Many people have address clustering from relational representation (see e.g. (0)).

It is not our goal to find a single "best" data partitioning. Instead, we identify a number of alternative clusterings which are involved in more complex features improving predictive accuracy of statistical models. Objects may be clustered based on different attributes, usin different similary measures, and with different numbers of clusters found The usefulness of a grouping can be assessed only in relation to a particular set of predictions being made.

Cluster-based concept and relation invention, as described in this paper, differs importantly from using aggregation, in a sense commonly used in databases, as a means of summarization. Aggregation is essential in statistical relational learning and is also used to create new, rich types of features from relational representation (**?**). Using aggregates creates richer features than modeling a boolean, table empty/non-empty feature as is the case in classical logic-based relational learning approaches (0). The need for aggregates in relational learning comes from the fact that the central type of relational representation is a table (set); the data is represented by a number of tables, and database queries result in tables. Statistical models, on the other hand, work with scalar values, real numbers, integers, or categorical variables. Aggregation allows summarizing information in a table per given observation into a scalar value which can be included in a statistical model, for example, *average* of a word count in all cited documents, or a citing document with *max* number of incoming links. Aggregates are essential to our approach; each node in our search space evaluates into a table, which in turn is aggregated to produce a number of scalar feature candidates. The advantage of clusters comes at another level to create central relational entities from which features are generated; aggregates are applied at the next step to the tables resulting from queries which can involve both the cluster relations and the original relations, for example, the number of documents in the same cluster and published in the same conference is a *count* or *size* aggregate of a corresponding derived view.

The idea of augmenting the existing representation with new relations or predicates is, of course, not new. In inductive logic programming it is known as "predicate invention". For example, Statistical Predicate Invention (0) which was proposed for learning in hypertext domains, represents classifications produced by Naive Bayes as a new predicate added to FOIL (**?**). Statistical Predicate Invention preserves FOIL as the central modeling component and calls statistical modeling from within the inner structure navigation loop to supply new predicates. Our approach differs in that we use statistics rather than logic as a modeling component, and more importantly in this context, we advocate the use of cluster-based relation invention as a means to enrich feature spaces by adding to schema many types of clusters, not only those of a response concept, thus creating first-class relational concepts, such as "topics" or "communities", which have a clean "identity" as the world representation entities.

Concept invention could also, in theory be done in other types of relational learning, such as in those using graphical models, e.g. Probabilistic Relational Models (PRMs) (**?**; 0), which are generative models of joint probability distribution capturing probabilistic influences between entities and their attributes in a relational domain. However, such generative models are not condusive to searching for complex features, as is done in ILP and in this paper.

6. Conclusions and Future Work

We presented a framework for learning predictive statistical models from relational data where new concepts and relations are derived by clustering items in the original database schema. Adding these new relations to the database schema (as opposed to just using them as aggregates to derive new features) allows a more efficient search of a richer feature space. Including new relations such as $docsByWords$ allows discovery of new features such as $exists[publishedIn(D1, V), docsByWords(D, C), docsByWords(D1, C$

We applied cluster-relation invention to the task of predicting the publication venue of scientific papers from the CiteSeer database, which contains citations, paper authorship, and word content. We used clustering to derive new first class relational entities reflecting hidden topics of papers, author communities and word groups. New cluster relations included into the feature generation process, in addition to the original relations, resulted in the creation of richer cluster-based features, where clusters enter into more complex relationships with existing background relations rather than only provide dimensionality reduction. Using relation invention gives more accurate models than those built only from the original relational concepts.

Two models, with and without cluster relations, were compared on feature streams of 3,500 unique features. Relation invention gave accuracy improvement of about 3.0 percentage points over the non cluster-based model. Enriching the schema with more compact cluster relations can also lead to a more rapid discovery of predictive features. Cluster relations are smaller than the original relations from which they are derived; this translates into lower costs of generating cluster-based features. Focussing search on relations containing clusters, we get high accuracy in fewer than 100 features, in contrast to the breadth-first strategy which achieved the same accuracy only after considering more than 1,500 features. This is important, as the most computationally demanding process in the SLR methodology is database query evaluation for feature generation.

We envision several improvements to the relation invention methodology. Richer types of clusters can be derived from more complex sets of attributes than those immediately available in a single relation. For example, publication venues and authorship data are in two separate relations which both can be used to cluster publication venues based on the authors who publish in them. Also, clustering can be performed lazily as a corresponding depth in the feature search space is reached by the feature generation process. In contrast to "propositionalization" (?), which implies a decoupling of relational feature generation and modeling, SLR is dynamic and allows for a more natural introduction of this extension.

References

Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

A. V. Aho, Y. Sagiv, and J. D. Ullman. Equivalences among relational expressions. *SIAM Journal of Computing*, 8(2):218–246, 1979.

M. Craven and S. Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, 43(1/2):97–119, 2001.

Saso Dzeroski and Nada Lavrac. An introduction to inductive logic programming. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 48–73. Springer-Verlag, 2001.

Dean Foster and Lyle Ungar. A proposal for learning by ontological leaps. In *Proc. of Snowbird Learning Conference*, Snowbird, Utah, 2002.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. of IJCAI99*, pages 1300–1309, Stockholm, Sweden, 1999.

L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 307–338. Springer-Verlag, 2001.

L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley-Interscience, 1990.

Mathias Kirsten, Stefan Wrobel, and Tamas Horvath. Distance based approaches to relational learning and clustering. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 213–230. Springer-Verlag, 2001.

S. Kramer, N. Lavrac, and P. Flach. Propositionalization approaches to relational data mining. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 262–291. Springer-Verlag, 2001.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

Werner Nutt, Yehoshua Sagiv, and Sara Shurin. Deciding equivalences among aggregate queries. In *Proc. of PODS-98*, pages 214–223, 1998.

Dmitry Pavlov, Alexandrin Popescul, David M. Pennock, and Lyle H. Ungar. Mixtures of conditional maximum entropy models. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, 2003.

Claudia Perlich and Foster Provost. Aggregation-based feature invention and relational concept classes. In *Proc. of KDD-2003*, 2003.

Alexandrin Popescul and Lyle H. Ungar. Statistical relational learning for link prediction. In *IJCAI Workshop on Learning Statistical Models from Relational Data*, 2003.

Alexandrin Popescul and Lyle H. Ungar. Structural logistic regression for link analysis. In *KDD Workshop on Multi-Relational Data Mining*, 2003.

Alexandrin Popescul, Lyle H. Ungar, Steve Lawrence, and David M. Pennock. Statistical relational learning for document mining. In *Proceedings of IEEE International Conference on Data Mining (ICDM-2003)*, 2003.

J.R. Quinlan and R.M. Cameron-Jones. Induction of logic programs: FOIL and related systems. *New Generation Computing*, 13:287–312, 1995.

G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw Hill, 1983.

E. Shapiro. *Algorithmic Program Debugging.* MIT Press, 1983.

A Random Forest Approach to Relational Learning

Anneleen Van Assche ANNELEEN.VANASSCHE@CS.KULEUVEN.AC.BE
Celine Vens CELINE.VENS@CS.KULEUVEN.AC.BE
Hendrik Blockeel HENDRIK.BLOCKEEL@CS.KULEUVEN.AC.BE
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

Sašo Džeroski SASO.DZEROSKI@IJS.SI
Department of Knowledge Technologies, Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Abstract

Random forest induction is an ensemble method that uses a random subset of features to build each node in a decision tree. The method has been shown to work well when many features are available. This certainly is the case in relational learning, especially when aggregate functions, combined with selection conditions on the set to be aggregated, are included in the feature space. This paper presents an initial exploration of the use of random forests in a relational context. We experimentally validated our approach both in a business domain, and on a structurally complex data set.

1. Introduction

The motivation for this paper is based on two observations. First, random forest induction, an ensemble method that builds decision trees using only a random subset of the feature set at each node, has been shown to work well when many features are available (Breiman, 2001). Because relational learning typically deals with large feature sets, it seems worthwhile to investigate whether random forests perform well in this context. Second, using random forests allows an extension of the feature space by including aggregate functions, possibly combined with selection conditions. So far, this combining has been considered a difficult task (Blockeel & Bruynooghe, 2003), because the feature set grows quickly, and because the search space is less well-behaved due to the non-monotonicity problem (Knobbe et al., 2002).

The paper is organized as follows. In Section 2, we discuss random forests. Section 3 illustrates the problem of combining aggregates with selection conditions in relational learning. Our method, which is a random forest approach to relational learning, is presented in Section 4. In Section 5, we experimentally evaluate our method in both a structurally complex domain, and a (highly non-determinate) business domain. Finally, we formulate conclusions and some ideas for future research in Section 6.

2. Random Forests

Random forest induction (Breiman, 2001) is an ensemble method. An ensemble learning algorithm constructs a set of classifiers, and then classifies new data points by taking a vote of the predictions of each classifier. A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members, is that the classifiers are accurate and diverse (Hansen & Salamon, 1990). An accurate classifier does better than random guessing on new examples. Two classifiers are diverse if they make different errors on new data points. There are different ways for constructing ensembles: bagging (Breiman, 1996) and boosting (Freund & Schapire, 1996) for instance, introduce diversity by manipulating the training set. Several other approaches attempt to increase variability by manipulating the input features or the output targets, or by introducing randomness in the learning algorithm (Dietterich, 2000). Random forests try to increase diversity among the classifiers by resampling the data, and by changing the feature sets over the different (tree) model induction processes. The exact procedure is as follows:

```
• for $i = 1$ to $k$ do:
    – build dataset $D_i$ by sampling with replacement from dataset $D$
    – learn a decision tree $T_i$ from $D_i$ using randomly restricted feature sets
```

- make predictions according to the majority vote of the set of k trees

The part of the algorithm where random forests differ from the normal bagging procedure is emphasized. When inducing a decision tree, the best feature is selected from a fixed set of features F in each node. In bagging, this set of features does not vary over the different runs of the induction procedure. In random forests however, a different random subset of size $f(|F|)$ is considered in each node (e.g. $f(x) = 0.1x$ or $f(x) = \sqrt{x}, \ldots$), and the best feature from this subset is chosen. This obviously increases variability. Assume for instance that $f(x) = \sqrt{x}$, and that two tests t_1 and t_2 are both good features for the root of the tree, say t_1 is the best and t_2 is the second best feature. With a regular bagging approach t_1 is consistently selected for the root, whereas with random forests both t_1 and t_2 will occur in the root nodes of the different trees respectively with frequency $1/\sqrt{|F|}$ and $1/\sqrt{|F|} - 1/|F|$. So t_2 will occur only with slightly lower frequency than t_1.

Random forests have some interesting properties (Breiman, 2001). They are more efficient since only a sample of $f(|F|)$ features needs to be tested in each node, instead of all features. They also do not overfit as more trees are added. Furthermore, they are relatively robust to outliers and noise, and they are easily parallellized.

The efficiency gain makes random forests especially interesting for relational data mining, for which it is typical that there is a large number of features, many of which are expensive to compute. On the other hand, relational data mining offers an interesting test suite for random forests, exactly because the advantage of random forests is expected to become more clear for very large feature spaces. In relational data mining, such data sets abound. Moreover, using random forests allows us to even enlarge the feature set by including aggregate functions, possibly refined with selection conditions. We discuss this in the next section.

3. Aggregates and Selection in Relational Learning

When considering multi-relational learning, sets (or more general, bags) need to be handled. They are represented by one-to-many or many-to-many relationships in or between relations. Blockeel and Bruynooghe (2003) categorize current approaches to relational learning with respect to how they handle sets. Whereas ILP (Muggleton, 1992) is biased towards selection of specific elements, other approaches, such as PRM's (Koller, 1999) or propositionalisation approaches (e.g. the one by Krogel and Wrobel (2001)) use aggregate functions, which compute a feature of the set that summarizes the set. The latter methods are optimized for highly non-determinate (e.g. business) domains, whereas the former is geared more towards structurally complex domains, e.g. molecular biology, language learning, etc.

Perlich and Provost (2003) present a hierarchy of relational concepts of increasing complexity, where the complexity depends on that of any aggregate functions used. They argue that ILP is currently the only approach that can explore all concepts in the hierarchy. However, combining aggregates and selection in ILP is not a trivial task. Until now ILP-like feature construction and the use of aggregation features have mostly been combined in relatively restricted ways, e.g., without allowing complex conditions within the aggregation operator.

There are several difficulties that arise. First of all, while approaches such as ILP by themselves already suffer from a large search space, integrating aggregation operators into them further increases the hypothesis space (Blockeel & Bruynooghe, 2003). Moreover, the search space is less well-behaved because of the problem of non-monotonicity (Knobbe et al., 2002), which renders traditional pruning methods in ILP inapplicable. Clearly, the idea of performing a systematic search for a good hypothesis has to be given up; the search will have to be more heuristic.

Let us explain the reduction in pruning opportunities. Suppose refining a hypothesis h consists of adding a condition into its body. Then refining h can only decrease its coverage, so if any h is unsuitable because it has too low coverage, then any refinements of h must be unsuitable as well; hence we can prune the search space at h. If conditions are added within an aggregate, though, the coverage of the hypothesis may increase instead of decrease, hence such pruning cannot safely be applied any more.

We illustrate the above with an example:

```
a(X) :- count(Y, child(X,Y), C), C>2.
```

adding a literal inside the count aggregate gives

```
a(X) :- count(Y, (child(X,Y), female(Y)), C), C>2.
```

which must have at most the same coverage (people with more than two daughters must be a subset of people with more than two children). If we consider

```
a(X) :- count(Y, child(X,Y), C), C<=2.
```

and its refinement

```
a(X) :- count(Y, (child(X,Y), female(Y)), C), C<=2.
```

then the examples covered by the refinement are a superset of the original set (people with at most two children certainly have at most two daughters, but people with at most two daughters may have more than two children).

Knobbe et al. (2002) present an approach to combine aggregates with reasonably complex selection conditions by generalizing the selection graph pattern language. Selection graphs are a graphical description of sets of objects in a multi-relational database.

In this paper we explore an alternative approach which is based on random forests and explores the same feature space as Knobbe et al. (2002).

4. A Random Forest Approach to Relational Learning

Knobbe et al. (2002) provide some definitions that are useful for our explanation. An *aggregate condition* is a triple (f, o, v) with f being an aggregate function, o a comparison operator and v a value of the domain of f. An aggregate condition is *monotone* if, given sets of records S and S', $S' \subseteq S, f(S')\ o\ v \Rightarrow f(S)\ o\ v$. The example in the previous section showed that $(count, >, v)$ is a monotone aggregate condition, while $(count, <, v)$ is not.

As mentioned in Section 3, Knobbe et al. (2002) describe a method for including aggregate functions in selection graphs. If an aggregate condition is monotone, selection conditions on the set to be aggregated are allowed.

We implemented a similar method in Tilde (Blockeel & De Raedt, 1998), which is included in the ACE data mining system (Blockeel et al., 2002). Tilde is a relational top-down induction of decision trees (TDIDT) instantiation, and can be viewed as a first order upgrade of C4.5 (Quinlan, 1993), using logical queries in tree nodes.

We expanded the feature set considered at each node in the tree to consist of the regular features (with aggregate conditions included), augmented with any refinements of aggregate conditions (such as the ones discussed in the example of Section 3) used on the path from the root to the current node, when the "yes"- branch was taken.

Moreover, we built in a filter that allows only a random subset of the tests to be considered at each node[1]. Around this procedure we built a wrapper in order to get bagging. These two mechanisms together result in a random forest induction method.

We want to point out that the use of random forests tackles the difficulties that arise when combining aggregation and selection in ILP. First, the size of the search space is limited because we only consider a subset of the possible features at each node in a tree. Second, using decision trees, the comparison operators "<" and ">" are equivalent up to switching branches. This means that we can restrict ourselves to using monotone aggregate conditions. However, even if we would allow non-monotone aggregate conditions, the refinements of these aggregates wouldn't be chosen to split a node, because they don't add any extra information to the tree.

Another advantage of using random forests is that more randomly generated features can be included in the feature space. This includes the construction of aggregate conditions, where the aggregate clause consists of a number of conjuncts. This way, tests as

```
count(Y, (child(X,Y),female(Y),blonde(Y)), C), C>3
```

could be immediately considered at the root node, while now they can only be found as a refinement of an aggregate already occuring in the tree. Non-monotone aggregate functions could then be refined by deleting a number of conjuncts. This would lead to a bottom-up search strategy included in our top-down approach, and would be very interesting to explore. However this is not yet implemented in our system.

5. Experimental results

The aim of our experiments is to investigate the benefit of random forests in a relational setting where the feature set is expanded with aggregates and even refinements of aggregates, both in a business domain and a structurally complex data set. In the following sections we first describe our experimental setup, and then present results on two well-known data sets: the Mutagenesis data set (Srinivasan et al., 1999) and the Financial data set (Berka, 2000).

5.1. Experimental setup

All experiments were performed using a 5-fold cross-validation and were carried out five times (with the same folds). Afterwards the results were averaged in order to obtain a more reliable estimate of the per-

[1]This actually differs from the definition in (Breiman, 2001) where a random subset of the attributes, instead of the tests, is chosen.

formance of the random forests. Different parameters needed to be set. First of all, the number of random features in each node: we chose to consider random subsets of 100%, 75%, 50%, 25%, 10%, and the square root of the number of tests at each node in the trees. We have also examined the influence of the number of trees in the random forests, experimenting with 3, 11, and 33 trees. To investigate the performance of random forests in the context of aggregation, we have performed experiments with and without the use of aggregates, and also with refinement of aggregates, where conditions on the set to be aggregated are added.

5.2. Mutagenesis

For our first experiment we used the Mutagenesis data set. This ILP benchmark data set, introduced to the ILP community by Srinivasan et al. (1999), consists of structural descriptions of 230 molecules that are to be classified as mutagenic (60%) or not. The description consists of the atoms and the bonds that make up the molecule. For this data set we expect only a slight gain in accuracy using aggregates, since Mutagenesis does not have many numerical attributes nor is it very non-determinate.

Table 1 shows the results obtained on the Mutagenesis data. For each column the best result is in bold. For the experiments with refined aggregates, we can see that the best result is always one where random subsets of tests are used. In fact, we see that applying random forests using random samples with down to 25% of the possible tests gives either an improvement or at least no significant decrease in accuracy over bagging[2], while it certainly results in an efficiency gain. As expected it also clearly outperforms Tilde.

For the results without the use of aggregates we see that random forests don't seem to enhance accuracy over bagging, while still performing better than Tilde. This may show that random forests indeed profit from large feature sets. These results also support Breiman's (2001) statement that random forests do not tend to overfit. The average difference between training and test set error for random forests using randomly 10% of the tests at each node is 6% while for Tilde (using all features) it is 16%. As was expected, we also found that adding more trees to the forest, clearly increases accuracy in all cases. In Figure 1 the results for a random forest with 33 trees either making use of refined aggregates (RA) or aggregates without refinement (WA) or using no aggregates (NA) are compared for different sizes of random subsets in the nodes

[2]In Table 1 bagging is corresponding to random forests with 100% of the tests used in the node.

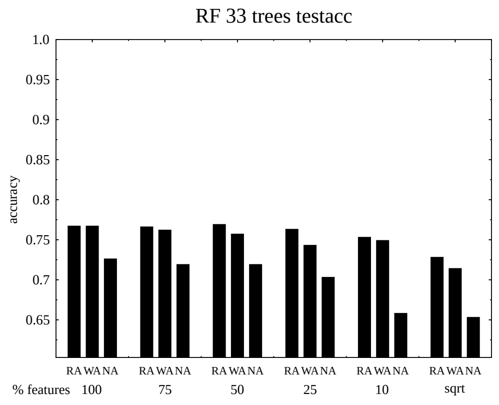


Figure 1. Accuracy for random forests consisting of 33 trees, either not using aggregates (NA), using aggregates (WA) or refined aggregates (RA) on the Mutagenesis data. Results are shown for different numbers of randomly chosen features.

of the trees (100% down to square root of the number of features at the node). As we can see from Figure 1 there is always (no matter which number of features is used) an improvement when aggregates are added. When we allow refinements of the aggregates in the queries, in many cases another slight improvement is observed. An example of a test which was frequently found across the different trees was the following aggregate

```
count(BId, (bond(BId,Mol,At1,At2,Tp)), C), C>28.
```

with refinement

```
count(BId, (bond(BId,Mol,At1,At2,Tp),
           atom(Mol,At1,carbon)), C), C>28.
```

The first part of the example represents the set of all molecules that have at least 28 bonds. This aggregate was also found to be a good test by Knobbe et al. (2002). The refinement of that aggregate describes all molecules that have at least 28 bonds connected to an atom of type carbon.

5.3. Financial Data Set

Our second experiment deals with the Financial data set, from the discovery challenge organized at PKDD '99 and PKDD '00 (Berka, 2000). This data set involves learning to classify bank loans into good (86%) and bad loans. The data set consists of 8 relations and contains for each loan customer information and account information which includes permanent orders and several hundreds of transactions per account. This

	Tilde			RF 3 trees			RF 11 trees			RF 33 trees		
	RA	WA	NA	RA	WA	NA	RA	WA	NA	RA	WA	NA
1	0.717	0.730	0.683	0.721	0.710	**0.704**	0.754	0.745	**0.718**	0.767	**0.767**	**0.726**
0.75	0.723	0.728	**0.687**	0.725	0.730	0.677	0.758	**0.750**	0.704	0.766	0.762	0.719
0.50	**0.728**	0.710	0.678	**0.735**	0.736	0.690	**0.763**	0.743	0.710	**0.769**	0.757	0.719
0.25	0.724	**0.744**	0.670	0.728	**0.737**	0.700	0.753	0.743	0.703	0.763	0.743	0.703
0.10	0.717	0.727	0.653	0.723	0.720	0.657	0.753	0.735	0.651	0.753	0.749	0.658
sqrt	0.701	0.719	0.654	0.693	0.700	0.638	0.707	0.710	0.668	0.728	0.714	0.653

Table 1. Accuracy results on the Mutagenesis data set. The rows describe the different proportions of random test subsets in the nodes. The columns compare accuracies for experiments using refined aggregates (RA), aggregates without refinement (WA) and no aggregates (NA) for Tilde and random forests with different number of trees.

	Tilde			RF 3 trees			RF 11 trees			RF 33 trees		
	RA	WA	NA	RA	WA	NA	RA	WA	NA	RA	WA	NA
1	0.978	0.978	0.746	0.978	0.985	0.839	0.988	0.986	0.849	0.987	0.988	0.853
0.75	0.980	0.984	0.768	0.981	0.980	0.833	0.985	0.984	0.847	0.987	0.988	0.853
0.50	**0.985**	0.981	0.766	0.983	0.983	0.823	0.990	**0.989**	0.854	0.987	0.988	0.862
0.25	0.978	**0.984**	0.752	**0.991**	**0.990**	0.847	**0.990**	0.986	0.855	**0.992**	**0.990**	0.859
sqrt	0.954	0.949	0.766	0.959	0.970	0.847	0.972	0.983	0.865	0.984	0.986	0.864
0.10	0.950	0.972	**0.827**	0.957	0.947	**0.863**	0.964	0.964	**0.866**	0.964	0.957	**0.866**

Table 2. Accuracy results on the Financial data set. The rows describe the different proportions of random test subsets in the nodes. The columns compare accuracies for experiments using refined aggregates (RA), aggregates without refinement (WA) and no aggregates (NA) for Tilde and random forests with different number of trees.

problem is thus a typical business data set which is very non-determinate.

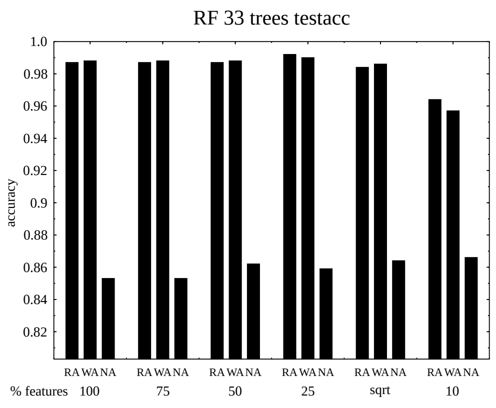


Figure 2. Accuracy for random forests consisting of 33 trees, either not using aggregates (NA), using aggregates (WA) or refined aggregates (RA) on the Financial data. Results are shown for different numbers of randomly chosen features.

Table 2 provides the results obtained on the Financial data. For this data set we want to point out that the data distribution is quite skewed; 86% of the examples are positive. We also note that in this case the square root of the number of tests is on average larger than 10%, so we switched these two rows in Table 2. We observe that for all experiments, using only a random part of the tests to select the best test (certainly down to 25% of the tests) seems to be advisable since there is no drop in accuracy (no significant gain either) and we profit from the efficiency gain by testing fewer features in each node. Contrary to the results on the Mutagenesis data, when no aggregates are used, the accuracy seems to increase while adding more randomness. This is actually due to the skewness of the Financial data: without randomness the predictive accuracy of the forests is below the accuracy of allways predicting the default value (86%). When adding more randomness, some trees tend to allways predict the default value, or at least predict a very small proportion of the examples to be negative. So this might improve accuracy up to the default, but the trees become certainly less informative.

Adding aggregates resulted in much more compact trees, and, as can be seen in Figure 2, yielded a large improvement of accuracy. Allowing refinements in these aggregates didn't seem to improve accuracy further though. This may be due to the fact that the theory that needs to be learned, can be represented very well without complex selection within the aggregates. We also found that among the trees aggregates were rarely refined.

5.4. Experimental conclusions

For the experiments using aggregates, decreasing the number of random tests considered in a node to a certain threshold slightly increases the predictive accuracy. If we go below that threshold the accuracy decreases again. However it seems quite difficult to determine an optimal value for this threshold. For the chosen data sets, sampling randomly around 25% of the tests at each node, gives either an improvement or at least no significant decrease in accuracy. Breiman (2001) on the other hand, obtained improvements with even much smaller proportions of features in the propositional case. Also the average improvement was much higher. This difference might be due to fact that in our approach a random subset of tests is taken while Breiman takes a random subset of the attributes, and selects the best test using these attributes. Or it might show that in relational learning random forests do not work as well as in propositional learning.

We also found that the results without aggregates differ a lot over the two data sets: on Mutagenesis decreasing the random sample of tests decreases accuracy while on the Financial data set the accuracy increases. This last effect was because of the skewness of the data.

As was expected, adding more trees to the forest indeed increases accuracy. Of course more trees mean longer runtimes, so there is always a trade-off between efficiency and accuracy.

Concerning aggregation and selection, allowing aggregate functions in the tests always gave an improvement, whereas additional refinement of the aggregates didn't consistently increase accuracy.

While in general performing at least as well as bagging, random forests are computationally more efficient than bagging. If we compare random forests, consisting of 33 trees and using 25% of the features, with bagging using 33 trees, we found an efficiency gain of factor 2.6 on the Mutagenesis data and a factor 1.3 on the Financial data. Again this factor is smaller than in the propositional case, where there is no need to generate tests at each node, since the set of tests remains the same for all nodes. In the relational case on the other hand constructing a new node in the tree consists of two steps: first all possible refinement queries need to be generated and then secondly those queries are executed on the remaining examples and for each query a heuristic is computed. Afterwards, based on those heuristics the best test is placed in the new node. When using random forests the time for executing the queries and calculating the heuristics is reduced, since only a sample of all generated refinements is considered. But in our current implementation the time for the first step remains the same, because all possible refinements are still generated and only afterwards a random sample of that set is taken.

6. Conclusions and Future Work

In this paper, we have explored a random forest approach to relational learning. Random forests have been shown to work well when many features are available. This is often the case in relational learning. Thus the investigation of random forests in this context seems worthwhile.

Moreover, when using random forests, we can further enlarge the feature space by including aggregate conditions and refining them with selection conditions on the set to be aggregated. This combination of aggregation with selection conditions is not a trivial task, because the feature space grows quickly and the search space is less well-behaved due to the non-monotonicity problem. However, the use of random forests overcomes these problems.

We experimentally validated the strength of random forests and the use of (refined) aggregates in the relational case. Our results show that, if we randomly decrease the feature set at each node in a tree down to a certain level, the classification performance is at least as good as bagging, so we profit from the gain in efficiency. In our chosen data sets, this level turned out to be 25% of the features, below this threshold classification performance decreased. The benefit of including aggregate functions was clear from the results. Refinements of aggregates sometimes yielded another slight performance improvement. However, some effects are still unclear and therefore, more data sets should be explored.

Now we turn to three possible directions for future work. As mentioned in Section 4, we would like to introduce more randomly generated features, e.g. aggregate functions with a number of selection conditions on the set to be aggregated. At this moment, these features can only be found as refinements of other aggregate functions. These randomly generated aggregate functions would increase diversity between features, and could be refined by deleting some selection conditions, if the function is non-monotone. This would lead to research into the benefits of bottom-up search strategies included in a top-down approach.

Alternative search heuristics could also be explored. Instead of always choosing the best feature (out of the features in the randomly chosen subset) at each node in a tree, features could be chosen according to a Boltzman distribution. This way, even less good features would have a chance to be chosen; this chance being proportional to their quality. This would increase diversity among the trees, and probably improve the strength of the random forest.

It would also be interesting to investigate the effect of autocorrelation and degree diversity (Jensen et al., 2003) in the context of random forests, since these may influence the extent to which different trees in the forest are independent.

Acknowledgements

Anneleen Van Assche is supported by the Institute for the Promotion of Innovation by Science and Technology in Flanders (I.W.T.-Vlaanderen). Celine Vens is supported by the GOA/2003/08(B0516) on Inductive Knowledge Bases. Hendrik Blockeel is Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium) (F.W.O.-Vlaanderen).

References

Berka, P. (2000). Guide to the financial data set. *The ECML/PKDD 2000 Discovery Challenge.*

Blockeel, H., & Bruynooghe, M. (2003). Aggregation versus selection bias, and relational neural networks. *IJCAI-2003 Workshop on Learning Statistical Models from Relational Data, SRL-2003, Acapulco, Mexico, August 11, 2003.*

Blockeel, H., & De Raedt, L. (1998). Top-down induction of first order logical decision trees. *Artificial Intelligence*, *101*, 285–297.

Blockeel, H., Dehaspe, L., Demoen, B., Janssens, G., Ramon, J., & Vandecasteele, H. (2002). Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, *16*, 135–166.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Dietterich, T. (2000). Ensemble methods in machine learning. *Proceedings of the 1th International Workshop on Multiple Classifier Systems* (pp. 1–15).

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 148–156). Morgan Kaufmann.

Hansen, L., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Patern Analysis and Machine Intelligence*, *12*, 993–1001.

Jensen, D., Neville, J., & Hay, M. (2003). Avoiding bias when aggregating relational data with degree disparity. *Proceedings of the 20th International Conference on Machine Learning.*

Knobbe, A., Siebes, A., & Marseille, B. (2002). Involving aggregate functions in multi-relational search. *Principles of Data Mining and Knowledge Discovery, Proceedings of the 6th European Conference* (pp. 287–298). Springer-Verlag.

Koller, D. (1999). Probabilistic relational models. *Proceedings of the Ninth International Workshop on Inductive Logic Programming* (pp. 3–13). Springer-Verlag.

Krogel, M.-A., & Wrobel, S. (2001). Transformation-based learning using multi-relational aggregation. *Proceedings of the Eleventh International Conference on Inductive Logic Programming* (pp. 142–155).

Muggleton, S. (Ed.). (1992). *Inductive logic programming.* Academic Press.

Perlich, C., & Provost, F. (2003). Aggregation-based feature invention and relational concept classes. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 167–176). ACM Press.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* Morgan Kaufmann series in Machine Learning. Morgan Kaufmann.

Srinivasan, A., King, R., & Bristol, D. (1999). An assessment of ILP-assisted models for toxicology and the PTE-3 experiment. *Proceedings of the Ninth International Workshop on Inductive Logic Programming* (pp. 291–302). Springer-Verlag.

Playing Multiple Roles: Discovering Overlapping Roles in Social Networks

Alicia P. Wolfe PIPPIN@CS.UMASS.EDU
David Jensen JENSEN@CS.UMASS.EDU
140 Governor's Drive, University of Massachusetts, Amherst, MA 01003-4610 USA

Abstract

Many social networks can be characterized by the roles of the participants in the network. Participants with similar roles exhibit similar patterns of link structure. For example, a dentist has links to patients and other dentists, a PTA member has links to parents and students at a local school. We extend existing methods of finding roles to include multiple role labels (e.g., dentists who are also PTA members), and apply Gibbs sampling methods to find the maximum likelihood role labels and link probabilities on an unlabeled graph. We use synthetic data to evaluate the accuracy of this method compared to methods which assume a single role labeling.

1. Introduction

1.1. Why find roles?

The idea of modeling graph structure in social networks by finding "roles" of nodes in the graph was first introduced by Lorrain and White [13]. The basic idea is that two entities have the same role if they both are related to other entities in the same pattern (see figure 1). For example, a dentist usually has links to patients and hygenists, where a PTA (Parent-Teacher Association) member generally has links to parents and students at a local school. Ideally, good role labels on nodes in the graph would provide sufficient information to predict the link structure of the graph.

Many algorithms look for specific type of structure in the graph – highly connected subcomponents, or clusters. While clusters are interesting, we will focus instead on the *role* of a node. In some cases, the label being found does not cluster, for example, dentists are in general more connected to their patients and hygenists than other dentists. Note, however, that a cluster can be represented as a role with high probability of linking to itself and low probability of linking to other roles.

There are many precise mathematical definitions of what it means to have the same role, stochastic equivalence is the version used here:

> Let X be a random adjacency array. We say two nodes i and i' are stochastically equivalent if and only if the probability of any event about X is unchanged by interchanging nodes i and i'. [9]

In the simplest case "events" consist only of the links between nodes. In this case, two nodes are equivalent if they have the same probability of linking to nodes of each role label.

While attribute information can be added to this model, in its simplest form it only attempts to capture the link structure of the graph. We will use this fact to explore the effects of changes in the link generation model on role label prediction, in order to carefully isolate and analyse the effects of modifying the model to include multiple role labels.

Role induction allows the creation of a small abstract model of our graph. The model is a homomorphic image of the original graph, which contains summary information about the role labels and their relationships [17], as shown in figure 2. This model can be used to classify nodes into role types, predict links between nodes based on role label, summarize the properties of relationships among nodes, make predictions about similar graphs, and compare graphs to each other.

1.2. Multiple Role Labels

Existing work in social networks on finding roles assumes that an entity plays only one role in the graph. However, many situation require multiple roles to adequately model link structure. For example, a dentist may also be a PTA member, with links generated by

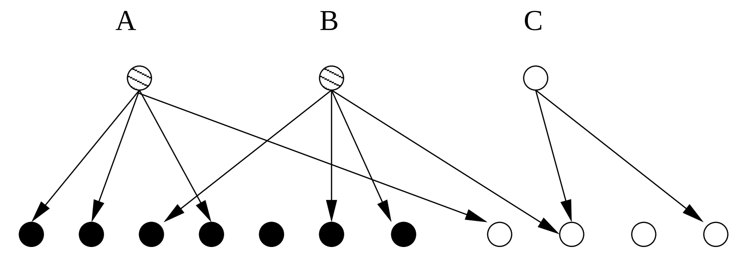


Figure 1. Nodes A and B have similar links to other nodes, and therefore the same role, whereas C shows a different link pattern. Only links connected to A, B and C are shown.

both roles. Methods which attempt to find only one set of role assignments in such a graph may be misled by the overlapping labels and end up finding neither correctly. If the number of labels used in the algorithm is appropriate for just one of the sets of role labels, the "noise" from the alternate role label can obscure both. For example, one might approximate the number of careers at 8 and attempt to find them, however, if there are 4 volunteer position roles which are also affecting the generation of links, the effective number of roles in the network is 32.

One adaptation of existing techniques to this problem would be to increase the number of role labels until each combination of roles has its own label, using the full 32 labels in our example. However, while this can find accurate results, it loses important information – we would really like to find a group containing all dentists, not all dentist-PTA members. This requires multiple labels on each node.

Using multiple labels provides a second benefit, as well, due to the overlapping labels on each node. The data provided by the graph is used more efficiently by this model, which means it can be learned from smaller graphs.

2. Modeling Multiple Roles

We assume that roles are separated into catagories, and that each entity has one label from each catagory. In our earlier example, one catagory would be "career" (dentist) another might be "volunteer position" (PTA member). It is always possible to include a "null" placeholder label in a catagory, if, for example, not all entities have volunteer roles.

We also assume that links are generated by each role catagory independently – there is a set of links generated by the fact that an entity is a dentist, another by the fact that they are a PTA member. The total set of links for an entity is the union of all such sets of generated links. The total probability of a link between two nodes is therefore the "or" of the probabilities that the role labels in each type generated a link.

It is therefore straightforward to calculate the total probability of a link between two nodes u and v with composite role labels I and J. If there are T role types, composite label $I = \{i_0, i_1, ...i_T\}$, and $J = \{j_0...j_T\}$. Label the probability of a link being generated between u and v by the t^{th} role label elements i_t and j_t, p_{i_t,j_t}. These p_{i_t,j_t} values will be the parameters of our model. The total probability of a link from node u to v is the "or" of the probabilities that each role type t generated a link:

$$p_{IJ} = 1 - \prod_{t \in T} 1 - p_{i_t,j_t}. \tag{1}$$

The parameters p_{IJ}, along with priors on the role labels, can be used to find the total probability of a particular labeling, parameters, and graph structure by combining the likelihood of each edge (present or absent) in the graph.

A graph G consists of V, the set of nodes, and E, the edge matrix with non-zero entries $e_{uv} = 1$ if and only if there is a link from node u to node v. If $\mathcal{R}$ is the set of role labels, where $R_u \in \mathcal{R}$ is the label for node u, then the joint probability of the entire graph $G = (V, E)$ with labeling $\mathcal{R}$ and parameters $\Theta = \{P(R_u)\} \cup \{p_{i_t,j_t}\}$ is:

$$p(G, \mathcal{R}, \Theta) = \prod_{u \in V} \left[P(R_u) \cdot \prod_{v \in V} pbinom(e_{uv}, p_{R_u,R_v}) \right]$$

where pbinom(sample, probability) is the probability of a particular sample of 0's and 1's being drawn from a binomial distribution with the specified probability.

This model can be used find the maximum likelihood role labelling and parameters given a graph, using Gibbs sampling.

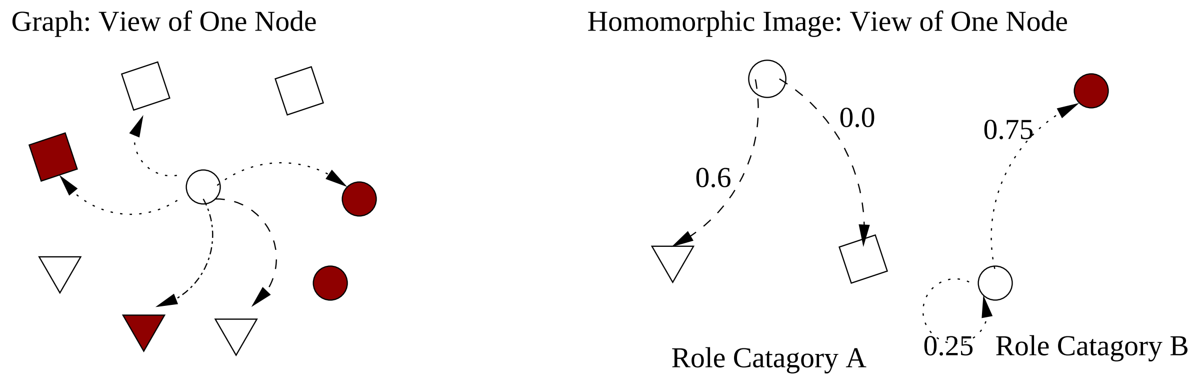


Figure 2. A multiple roles model of a graph. The image of the graph includes two types of role, one with 3 role labels, and one with two. Links can be generated by either role type.

3. Finding Multiple Roles

Snijders and Nowicki [16, 14] tackle the problem of finding both role labels and parameters for a graph with a single set of labels. They use EM, and, when that becomes intractible for larger graphs, Gibbs sampling. The unknown variables are the role labels of the nodes, and the parameters of the model are the link probabilities between role labels and priors for role labels. We extend their method by including multiple role labels using the model (and modeling assumptions) from the previous section.

Given an unlabeled graph G, the goal is to find the combination of role labels and parameters which is most likely. Gibbs sampling is one effective method of doing so, though as with many algorithms it may end up in a local extrema.

The entire algorithm is separated into two phases: burn in and sampling. Each phase consists of many incremental iterations of model estimation, each of which generates a sample labeling of the graph. The iteration steps in each of the two phases are identical, however, the samples generated by the burn in phase are assumed to be biased by the initial settings of the parameters, and are discarded. The sample labelings generated in the second phase are used to generate probabilities over labels for each node. The number of burn in and sampling steps necessary depends on the properties of the specific graph and model.

```
procedure $Gibbs - Sampling - Algorithm$

  Initialize parameters randomly
  Initialize labels randomly
  for numBurnInIters do
    RelabelGraph
  end for
  for numSampleIters do
    RelabelGraph
    for all $v \in V$ do
      $count_{v,R_v} = count_{v,R_v} + 1$
    end for
  end for
  for all $v \in V$ do
    for all $I \in \mathcal{R}$ do
      $P(R_v = I) = count_{v,I}/numSampleIters$
    end for
  end for

procedure $RelabelGraph$
  Part 1: Sample Labels
  for all $v \in V$ do
    $R_v =$ sample from $P(R_v = I|\{R_u, u \neq v\}, \Theta)$
  end for
  Part 2: Maximum Likelihood Parameters
  for all $I \in \mathcal{R}$ do
    $P(I) = argmax_{P(I)} P(G, V, \mathcal{R}, \Theta)$
    for all $J \in \mathcal{R}$ do
      for $t$ from 1 to $T$ do
        $p_{i_t,j_t} = argmax_{p_{i_t,j_t}} P(G, V, \mathcal{R}, \Theta)$
      end for
    end for
  end for
```

The Gibbs sampling algorithm is a general framework which can be applied with a variety of models. The algorithm can be adapted to a specific model (in our case the model from section 2) by inserting the appropriate formulae for the conditional distributions for the missing variables ($\mathcal{R}$), as well as the maximum likelihood values of the parameters (Θ).

3.1. Labeling Nodes

In the first part of each iteration, the algorithm generates a new value for each missing variable (role labels in this case) by sampling from the conditional distribution of the variable, given the rest of the graph, with its current labels. Since each node's role label is independent of the labels on nodes more than one link away, the conditional for the composite role label a single node v can be written as:

$$P(R_v = I | \{R_u, u \neq v\}, \Theta) \propto \\ P(I) \cdot \prod_{u \in V} pbinom(e_{uv}, p_{R_u,I}) \cdot pbinom(e_{vu}, p_{I,R_u})$$

The algorithm samples from this conditional distribution to generate a new label for each node in turn until the entire graph has been re-labeled.

3.2. Maximum Likelihood Parameters

Part 2 of each Gibbs sampling iteration calculates the maximum likelihood value of our model parameters, given a labeled graph. The parameters of our model consist of the link probabilities for each role pairing, and the priors for role labels. The priors for role labels can be estimated directly from counts over the data:

$$\begin{aligned} c_{R_k} &= \text{number of nodes labeled} R_k \\ n &= \text{number of nodes} \\ \hat{P}(R_k) &= c_{R_k}/n \end{aligned}$$

The remaining parameters to estimate are the link generation probabilities. Conveniently, these are factored according to the role elements. This will enable us to make more efficient use of the data in our social network, since each probability estimation for a role element includes information from a larger subset of the nodes in the network than if flat role labels were used. However, it also complicates our calculation of expected link probability values.

In the case where there is a single set of role labels, the new value of each link probability is easily calculated from simple counts. If c_{IJ} is the count of links between nodes of role I and nodes of role J, and n_{IJ} is the number of pairs of nodes labeled (I, J), the estimated link probability is simply $\hat{p}_{IJ} = c_{IJ}/n_{IJ}$.

However, finding the parameters when there are multiple role labels is slightly more complex. Consider two nodes, u and v. If there are T role labels on each node, each of the links between the nodes could have been generated by any one (or more) of the T pairs of role label elements on the nodes, as shown in figure 2. If a dentist/PTA member has a link to a hygenist/student, we must decide how much of the responsibility for the link to assign to the dentist-hygenist role pair, and how much to the PTA-student role pair.

This value is not immediately available, though it could be calculated from the actual parameters for the model. Instead, we create T "missing" variables on each link, $\{m_0, ...m_T\}$. Each m_t contains an estimate of the probability that a particular pair of role labels (in catagory t) generated the link.

Each m_t can be estimated from the parameter estimates from the last iteration. If u and v are the nodes at the ends of the link, and I and J are their role labels, then:

$$m_t = \hat{p}_{i_t,j_t}/\hat{p}_{IJ}$$

When the new value for $\hat{p}_{i_t,j_t}$ is calculated, all links between nodes that contain the labels i_t and j_t must be considered. If each link is weighted with the probability that it was generated by i_t and j_t, and the sum over all links taken, the result is a count of the number of links that should be attributed to the i_t and j_t pair. Dividing number of such pairs gives us the new link probability:

$$\begin{aligned} c_{i_t,j_t} &= \sum_{IJ | i_t \in I \wedge j_t \in J} m_t \cdot c_{IJ} \\ &= \sum_{IJ | i_t \in I \wedge j_t \in J} \hat{p}_{i_t,j_t}/\hat{p}_{IJ} \cdot c_{IJ} \\ \hat{p}'_{i_t,j_t} &= c_{i_t,j_t}/n_{i_t,j_t}. \end{aligned} \tag{2}$$

$\hat{p}_{IJ}$ is simply a function of the $\hat{p}_{i_t,j_t}$ values, as we saw in equation 1.

An alternative solution to this problem would be to create unknown labels on each link indicating the (hypothetical) set of role catagories which generated it. The sampling phase of the Gibbs sampler would then generate these labels, and counts could be taken over the set of links which have been labeled as being generated by a particular role catagory.

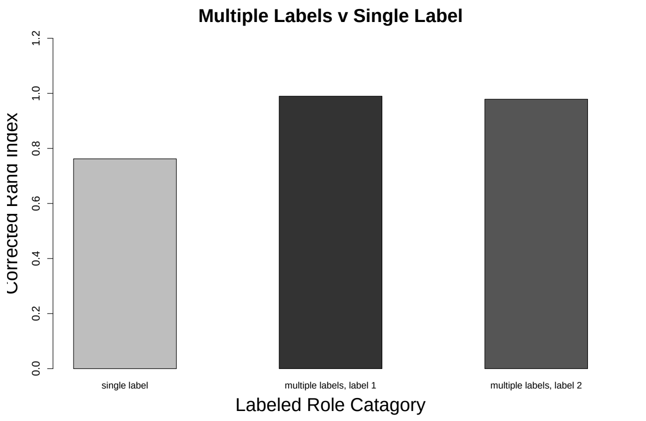
3.3. Reintroducing Lost Labels

If at the end of an iteration a label no longer appears anywhere in the graph, it is assigned to a random node before the begining of the next iteration.

4. Results on synthetic data

4.1. Synthetic Experiments

For our synthetic data experiments, graphs were constructed based on our modeling assumptions from sec-



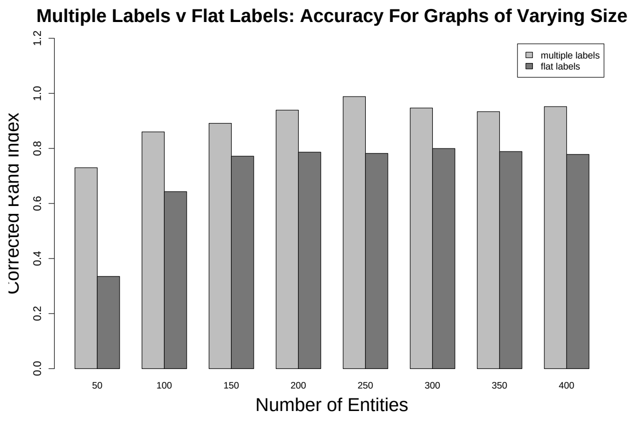


Figure 3. In the left chart, we can see the difference that ignoring one or more of the role catagories can make. For a graph with C role catagories, the multiple roles labeler was run with C catagories, and the results matched to the true labeling. It was then run with only 1, and again compared to the true labels. CRI values are the best match with any one of the C true role catagories, averaged over 20 runs. In the right chart, we see the advantage of using multiple roles over flattened roles when the data set is small. The multiple role labeler finds the true role structure, two roles with 3 values each, while the flat role labeler uses one role catagory with 9 values. CRI values are calculated with respect to the "flattened" correct role labelling, averaged over 20 runs.

tion 2. These graphs were used to evaluate the performance of the multiple roles Gibbs sampling algorithm and compare it to the same algorithm using a single role label. When the number of labels in each role catagory is the same, our experiments performed as expected, allowing the algorithm to find more accurate, expressive models and requiring less data to do so. However, there were also interesting problems when the number of values in different role catagories was not uniform, as we will see in section 4.5.

Each generated graph was constructed with a fixed number of role catagories, and a fixed number of role values in each catagory. Role priors and link generation probabilities were generated uniformly between 0.0 and 1.0. The graph labels and links were then assigned according to the model in section 2.

4.2. Evaluating Results on Labeled Data: the Corrected Rand Index

Since these experiments were run on generated data, the correct role labels, which generated the graph, were available. Therefore the roles found were evaluated with respect to this "correct" labeling using the Corrected Rand Index.

The Rand Index [15] evaluates the percentage of pairs of vertices which are correctly located either in the same set or different sets. [11] correct this measure for chance, with a resulting measure which has a mean of 0 for a randomly chosen partition, and a maximum value of 1.0 (complete agreement). If we define, for a correct labeling C and algorithmic labeling A:

$$\begin{aligned} n &= \text{number of nodes} \\ n_{ij} &= \text{count of nodes labeled i in C and j in A} \\ n_{i.} &= \text{count of nodes labeled i in C} \\ n_{.j} &= \text{count of nodes labeled j in A} \end{aligned}$$

The CRI measure is then:

$$\frac{\sum_{i,j}\binom{n_{ij}}{2} - \sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2} / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2} / \binom{n}{2}}$$

4.3. Insufficient Model Complexity: Single Role Label v Multiple Role Labels

If there are in fact C role catagories with 4 possible labels in each, the expectation would be that the effective number of roles in the graph would be 4^C, and therefore any model which uses only 4 labels to model the graph would perform poorly. Figure 3 demonstrates that this is in fact the case – a model with only 4 labels rarely matches any one of the C original role catagories.

The multiple roles labeling algorithm is also expected to generate two catagories of roles which match the two actual role catagories individually. Figure 3 is an also illustration that this property holds – each learned role catagory matches one of the true role catagories very closely (though it is not shown in the chart, they each match distinct role catagories from the actual graph).

4.4. Efficient Data Use: Multiple Roles vs Flat Labels

The underlying role structure can be better modeled with a single "flattened" label. In this case, if our graph has 2 role catagories with 3 values in each catagory, the single role labeler is allowed to use the full 9 role labels. When the graph is large this enables the model to compete with the multiple, however, when the graph is small the multiple roles model still has an advantage, as shown in figure 3. This stems from the fact that the multiple roles model has fewer parameters and reuses the data in the graph more efficiently.

4.5. Asymmetric Label Counts

In some cases, the fact that there is more than one label on each node can cause problems with the Gibbs sampling algorithm. Take a case in which there are 2 role catagories, with 3 elements in the first catagory, and 2 in the second. The algorithm simply assigns two labels to each node – it does not care whether a label is in the first catagory or the second. It may, therefore, model the 3 element catagory with 2 elements, and vice versa. In the experiments in figure 4 56 out of 100 trials had this type of problem. In most cases, the overall accuracy of the labeling is still moderately good, though the matches to individual labels are not as good.

However, one can do better. If the number of labels given to the algorithm for each role catagory is identical, the role catagories once again become truly interchangeable, and it does not matter which gets mapped to which true role catagory. In the experiments shown here, the Gibbs sampler was given two role catagories, each of which had the maximum number of labels across true role catagories (in the case of our graph with 3 and 2 labels in each catagory, the labeler would be asked to find two catagories with 3 labels in each). This model improved accuracy over all 100 trials, as we cans see in the first set of bars in figure 4. The surprising result here was that it does not cause a noticable drop in accuracy when the original results were correct. When the original algorithm would have mapped the 3 role catagory to 3 roles, and the 2 role catagory to 2 roles, resulting in very high accuracy, the model with 3 roles in each catagory barely changes the results (see the third set of bars in figure 4). The key here is that typically the algorithm almost completely eliminates the "extra" label in one catagory, which leads to a very close match in CRI values.

5. Discussion and Future Work

5.1. Finding Information in Real Data Sets

Role labels provide an interesting summary of the behavior of each node, which can then be treated as an attribute and used in other tasks such as classification. Using multiple roles provides an second level of complexity which gives us additional accuracy and information.

If part of the role label for a node is known (e.g., career but not volunteer postition), it becomes easier not just to infer the values of the other role, but to infer the existance of such a role. The model will have a more accurate labeling for careers when the role for volunteer position is added.

5.2. Alternate Inference Procedures

The model used here can be thought of as a bayes net consisting of many repeats of the structure in figure 5, in which two nodes and a single link variable are shown. "Link A" is true if a link is generated by role catagory A, "Link B" is true for links generated by role catagory B, and "link" is the or of these two values. Approximate inference procedures such as mean field, loopy belief propagation, generalized belief propagation should be compared to the sampling algorithm used here.

5.3. Interacting Roles

Independent link generation is convienient, since it leads to a simple model, however, in general, in may not be the case that roles generate links independently.

One interesting example of this type of role interaction is geographical location. Residents of the same town or neighborhood are likely to link to each other – they may attend the same schools, use the same dentist, or run into each other at the supermarket. A dentist in one town looks much like a dentist in another town, however, and still has links to the same other types – hygenists and patients. Geographical location does not so much generate additional links as determine which hygenists and patients the dentist will link to – those with similar locations.

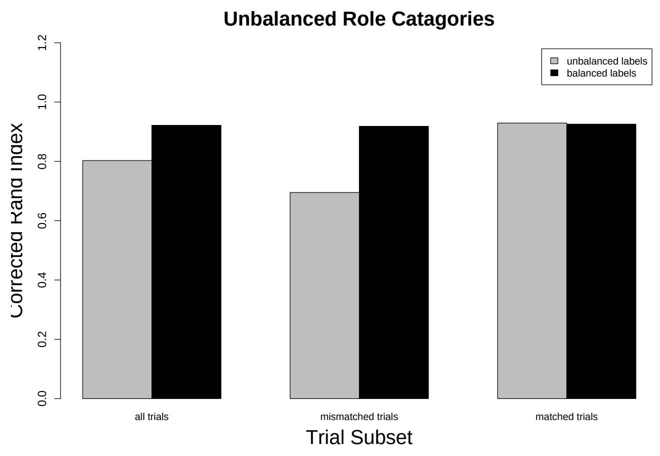


Figure 4. Trials on a graph with 3 roles in one catagory, 2 roles in the other. Two models were learned: an unbalanced model which started out with catagories with 2 and 3 roles, and a balanced model in which both catagories stared out with 3 roles. The leftmost graph plots CRI values for all 100 trials, the second only those trials where the unbalanced model found the wrong mapping of role catagories, and the third only those trials where the unbalanced model found the correct mapping of role catagories.

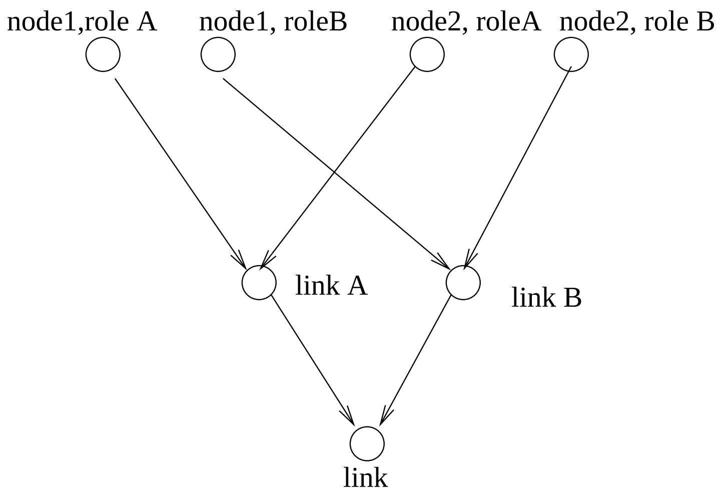


Figure 5. A partial bayes net showing link generation between two nodes, each with labels from two role catagories. The labels in each role catagory independently generate “Link A” and “Link B”. The presence of a link is determined by the “or” of these values.

5.4. Adding in Attribute Information on Nodes and Links

The model can easily be extended to include labeled links, by placing each link type in a separate graph and combining the likelihood of entity labels across these graphs.

Any attribute information which has only local effects (to the node or link on which it is located) can simply be added to the distribution over labels for the node. More complex interactions can introduced into the model, however, once the complexity reaches a certain point it may make more sense to treat the role labels as attribute inputs to another algorithm.

The role label on a node provides summary information about the link pattern for the node. It would be interesting to see if this summary information could at least partially replace or augment the "aggregation" functions used in some models. Aggregation functions compute a mathematical summary of a variable which exists on multiple neighbors of a node (using ave, mode, etc). Certainly the current role model at a first glance seems well suited to replacing aggregators which compute the count of related nodes with particular labels, and could perhaps be extended to compute summaries of other variables.

6. Acknowledgements

This research was facilitated in part by a National Physical Science Consortium Fellowship and by support from Sandia National Laboratories. This research was also supported by DARPA and AFRL under contract numbers F30602-00-2-0597. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA, AFRL, or the U.S. Government.

References

[1] C. Anderson, S. Wasserman, and K. Faust. Building stochastic blockmodels. *Social Networks*, 14:137–161, 1992.

[2] C. J. Anderson, S. Wasserman, and B. Crouch. A p* primer: logit models for social networks. *Social Networks*, 21:37–66, 1999.

[3] V. Batagelj. Notes on blockmodeling. *Social Networks*, 19:143–155, 1997.

[4] Vladimir Batagelj, Anuska Ferligoj, and Patrick Doreian. Generalized blockmodeling. *Informatica (Slovenia)*, 23(4), 1999.

[5] R.L. Breiger, S.A. Boorman, and P. Arabie. An algorithm for clustering relational data with applications to social network analysis and comparison to multidimensional scaling. *Journal of Mathematical Psychology*, 12:328–383, 1975.

[6] Martin G Everett. Role similarity and complexity in social graphs. *Social Networks*, 7:353–359, 1985.

[7] S.E. Fienberg and S. Wasserman. Categorical data analysis of single sociometric relations. *Sociological Methodology*, 12:156–192, 1981.

[8] O. Frank and D. Strauss. Markov random graphs. *Journal of the American Statistical Association*, 81:832–842, 1986.

[9] P. Holland, K.B. Laskey, and S. Leinhardt. Stochastic blockmodels: Some first steps. *Social Networks*, 5:109 – 137, 1983.

[10] P.W. Holland and S. Leinhardt. An exponential family of probability distributions for directed graphs. *JOurnal of the American Statistical Association*, 76(373):33–50, 1981.

[11] L. J. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[12] J. Kemeny and J. L. Snell. *Finite Markov Chains.* The University Series in Undergraduate Mathematics. Van Nostrand, 1960.

[13] F. Lorrain and H. C. White. The structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, 1:49–80, 1971.

[14] Krzysztof Nowicki and Tom A. B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–??, 2001.

[15] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

[16] T. Snijders and K. Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14:75–100, 1997.

[17] M.G. Everett S.P. Borgatti. The class of all regular equivalences: Algebraic structure and computation. *Social Networks*, 11:65–88, 1989.

[18] S. Wasserman and C. Anderson. Stochastic a posteriori blockmodels: Construction and assessment. *Social Networks*, 9:1–36, 1987.

[19] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications.* Cambridge University Press, 1994.

[20] S. Wasserman and P. Pattison. Logit models and logistic regression for social networks: I. an introduction to markov graphs and p*. *Psychometrika*, 61:401–425, 1996.

[21] D. R. White and K. P. Reitz. Graph and semigroup homomorphisms on networks of relations. *Social Networks*, 5:193–234, 1983.

Web Page Organization and Visualization Using Generative Topographic Mapping – A Pilot Study

Xiao-Feng Zhang, Chak-Man Lam, William K. Cheung

Department of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
{xfzhang, johanna, william}@comp.hkbu.edu.hk

Abstract

Automatic Web page organization and visualization is an effective way for foraging information in a Web structure. Web pages contain both text (content) and links (structure), implying that content and structure analysis techniques should be adopted and properly integrated. In this paper, we take the probabilistic model-based approach and extend a topography-preserving model known as Generative Topography Map (GTM). The extended GTM provides a principled way to integrate Web pages and hyperlinks and project them into a low-dimension latent space (2D in our case) for visualization. The proposed extension has been applied to an artificially created dataset and also the WebKB dataset for performance evaluation. Based on the preliminary results obtained, we propose several directions for future research.

Keywords: Web page organization and visualization, Web content and structure analysis, Generative Topography Map

1. Introduction

The need of foraging information in the Web has long been identified and the use of keyword-based search engines is by far the most effective way approach. However, the Web has been developed to such a stage that the information embedded in it can no longer be well managed using only the conventional keyword-based approach. In response to the need, many Web search service providers also provide some pre-categorized information. Web pages contain both text (content) and links (structure). Algorithms that can take into account both Web content and structures for supporting the self-organization and visualization of the Web-based information become increasingly important. While many related algorithms have been proposed for analyzing Web related information with applications to Web page classification [1], topic distillation [2], Web page ranking [3], Web communities identification [4], etc, not many of them are taking the statistical model-based approach which is known to be effective in discovering hidden knowledge in the observed data via the underlying model estimation process.

Generative Topographic Mapping is a model-based non-linear dimension reduction technique that tries to project the observed data space (normally at a high dimension) onto a latent variable space (at a lower dimension) via non-linear mappings, with the topographic relationship of the data preserved. It has known to be effective for data visualization. Among the related techniques, such as self-organizing map (SOM) [6] and linear local embedding (LLE) [7], GTM has the merit of possessing a rigorous probabilistic interpretation and is thus chosen in this paper to facilitate the integration of Web content and hyperlinks in a principled manner. Also, while this kind of techniques have long been applied to content-based automatic organization and visualization of digital archives like text documents [6] and music files [8], we, in this paper, extend the GTM to take into account also the hyperlink structure for the organization and visualization of Web pages. Our work is inspired from the use of the latent class model for related analysis done by Cohen *et al.* [9] which, however, does not support the visualization feature as provided by GTM.

1.1 Paper Organization

The remaining of the paper is as follow. Section 2 gives the background of GTM. Section 3 describes how GTM can be extended for integrating both content and links of Web pages. Section 4 provides the results obtained by applying the proposed GTM to the WebKB dataset. Limitations of this work as well as possible future research directions can be found in Section 5. Section 6 concludes the paper.

2. Generative Topographic Mapping (GTM)

GTM was first introduced in [5] as a probabilistic non-linear latent variable model. It was primarily designed for exploring the intrinsic dimension of a set of high-dimensional data by assuming that the data are generated due to a set of latent variables in a low-dimensional (usually 2D or 3D) latent variable space. Visualizing the latent variable space with the original data projected back to it can result in a *map* (for 2D case) that provides very intuitive understanding of the structure and organization of the high-dimensional data.

Let $\boldsymbol{t}_n = (t_{n1}, \dots, t_{nD})$ denote an instance of the set of observed high-dimensional data $\boldsymbol{T}$, $\boldsymbol{z} = (z_i, \dots, z_k)$ denote a finite set of sample points defined in the L-dimensional latent space, $y(z;\boldsymbol{W}) := \boldsymbol{W}\Phi(z)$ maps in an non-linear fashion a point in the latent space onto a corresponding point in the data space, with the mapping governed by a generalized linear regression model Φ weighted by $\boldsymbol{W}$. It is generative as it tries to find a parametric representation for explaining the data distribution in the data space, i.e., $p(\boldsymbol{T})$, based on $y(z;\boldsymbol{W})$. Thus, each sample point in the latent space would be mapped onto an L-dimensional non-Euclidean manifold in the data space. A Gaussian distribution in the data space is assumed for $\boldsymbol{t}$ given z_k in data space, given as

$$p(\boldsymbol{t} \mid z_k, \boldsymbol{W}, \beta) = (\frac{\beta}{2\pi})^{D/2} \exp\{-\frac{\beta}{2} \| y(z_k;\boldsymbol{W}) - \boldsymbol{t} \|^2\} \tag{1}$$

The overall log likelihood function for GTM is given as

$$L(W,\beta) = \sum_i \ln\{\frac{1}{K}\sum_k p(t_i \mid z_k, W, \beta)\} \tag{2}$$

The EM algorithm is typically used for the GTM parameter estimation, where the E-step is given as

$$\begin{aligned} P(z_k \mid \boldsymbol{t}_i, W_{old}, \beta_{old}) &= R_{ki}(\boldsymbol{W}_{old}, \beta_{old}) \\ &= \frac{p(\boldsymbol{t}_i \mid z_k, \boldsymbol{W}_{old}, \beta_{old})}{\sum_{k'} p(\boldsymbol{t}_i \mid z_{k'}, \boldsymbol{W}_{old}, \beta_{old})} \end{aligned} \tag{3}$$

And the M-step is

$$\sum_i \sum_k R_{ki}(\boldsymbol{W}_{old}, \beta_{old})\{\boldsymbol{W}_{new}\phi(z_k) - t_i\}\phi^T(z_k) = 0 \tag{4}$$

$$\frac{1}{\beta_{new}} = \frac{1}{ND}\sum_i \sum_k R_{ki}(\boldsymbol{W}_{old}, \beta_{old}) \| \boldsymbol{W}_{new}\phi(z_k) - \boldsymbol{t}_i \|^2 \tag{5}$$

GTM has been applied to visualizing high-dimensional data like images and documents. In this paper, we would like to extend it for applying to Web pages, where the modeling of hyperlink structure has to be incorporated.

3. Extending GTM for Modeling Web Content and Links

It is believed that the hyperlinks in a Web page provide further (non-linear) cues about how it should be related to other pages, instead of only relying on the Web page content. The question is how the link information should properly be taken into account in order to manifest its effect. Figure 1 shows one possible way for the integration which is inspired from [9].

Let d_j denotes the jth document, c_l denotes the lth cited document, and z_k denote the kth sample point in the latent space, N_{ij} denote the number of occurrences of the feature vector $\boldsymbol{t}_i$ in the jth document and A_{lj} denote the times that the jth document links to the lth document. Note that it is different from the one used in [9], where $\boldsymbol{t}_i$ stands for a word in [9] but a document feature vector in our case.

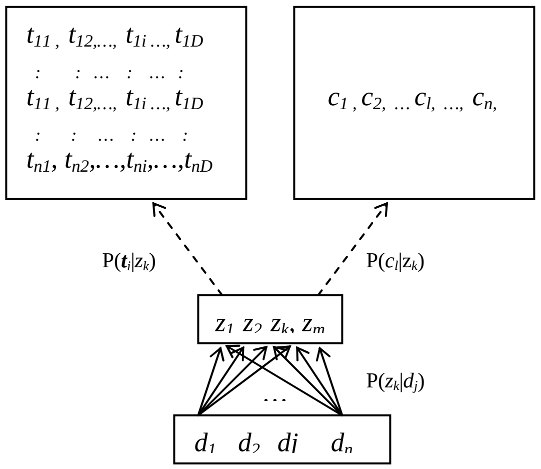


Figure 1 The extended Generative Topographic Mapping.

The whole likelihood function of this model can be written as:

$$
\begin{aligned}
L &= \sum_i \sum_l \sum_j \log p(t_i, c_l \mid d_j) \\
&= \sum_i \sum_l \sum_j \log[p(t_i \mid z_k, d_j) p(c_l \mid z_k, d_j) p(z_k \mid d_j)] \\
&= \alpha \sum_i \sum_j \log[p(t_i \mid z_k) p(z_k \mid d_j)] + (1-\alpha) \sum_l \sum_j \log[p(c_l \mid z_k) p(z_k \mid d_j)]
\end{aligned}
$$

By assuming that $\boldsymbol{t}_i$ and c_l are independent given d_j,

$$
\begin{aligned}
& p(\boldsymbol{t}_i, c_l \mid d_j) = p(\boldsymbol{t}_i \mid d_j) P(c_l \mid d_j) \\
& p(\boldsymbol{t}_i \mid d_j) = \sum_k p(\boldsymbol{t}_i \mid z_k) P(z_k \mid d_j) \\
& P(c_l \mid d_j) = \sum_k P(c_l \mid z_k) P(z_k \mid d_j)
\end{aligned}
\tag{6}
$$

Consider first $p(\boldsymbol{t}_i \mid z_k)$. The probability that a document feature vector $\boldsymbol{t}_i$ occurs in d_j can be computed, given as

$$
\begin{aligned}
p(\boldsymbol{t}_i \mid z_k) &= \int_\beta \int_{\boldsymbol{W}} p(\boldsymbol{t}_i, \boldsymbol{W}, \beta \mid z_k) d\boldsymbol{W} d\beta \\
&= \int_\beta \int_{\boldsymbol{W}} p(\boldsymbol{t}_i \mid \boldsymbol{W}, \beta, z_k) p(\boldsymbol{W}, \beta \mid z_k) d\boldsymbol{W} d\beta \\
&\approx p(\boldsymbol{t}_i \mid \boldsymbol{W}^*, \beta^*, z_k) p(\boldsymbol{W}^*, \beta^* \mid z_k)
\end{aligned}
\tag{7}
$$

By assuming that $p(\boldsymbol{W}, \beta \mid z_k)$ is uniformly distributed, we get

$$
p(t \mid z_k) \propto p(t_i \mid z_k, w, \beta) \tag{8}
$$

The log likelihood function becomes

$$
\begin{aligned}
L(\boldsymbol{W}, \beta) = \sum_j \Big[& \alpha \sum_i \ln\{\frac{1}{K} \sum_k N_{ij} p(\boldsymbol{t}_i \mid z_k, \boldsymbol{W}, \beta) P(z_k \mid d_j)\} \\
& + \sum_l (1-\alpha) \ln\{\frac{1}{K} \sum_k A_{lj} P(c_l \mid z_k) P(z_k \mid d_j)\} \Big]
\end{aligned}
\tag{9}
$$

where α here acts as a trade-off value to balance the weight of the content information and link information for the parameter estimation. Also, N_{ij} is diagonal in our case. In general, we can in fact use some kind of similarity measure for defining the value of N_{ij} (see Section 5 for more discussion).

Following the EM algorithm, one can then easily get corresponding E-step concerning contents as

$$
\begin{aligned}
P(z_k \mid t_i, d_j) = R_{ijk}(W_{old}, \beta_{old}) &= \frac{p(\boldsymbol{t}_i \mid z_k) P(z_k \mid d_j)}{\sum_{k'} p(\boldsymbol{t}_i \mid z_{k'}) P(z_{k'} \mid d_j)} \\
&\approx \frac{p(\boldsymbol{t}_i \mid z_k, w^*, \beta^*) P(z_k \mid d_j)}{\sum_{k'} p(\boldsymbol{t}_i \mid z_{k'}, w^*, \beta^*) P(z_{k'} \mid d_j)}
\end{aligned}
\tag{10}
$$

and the M-step related to the contents as

$$
\sum_i \sum_k \sum_l R_{ijk}(W_{old}, \beta_{old}) \{W_{new} \phi(z_k) - t_i\} \phi^T(z_k) = 0 \tag{11}
$$

$$
\frac{1}{\beta_{new}} = \frac{1}{ND} \sum_l \sum_j \sum_k R_{ijk}(W_{old}, \beta_{old}) \| W_{new} \phi(z_k) - t_l \|^2 \tag{12}
$$

Similar calculation can be applied to links related parameters for deriving the corresponding E-step and M-step. Finally, the E-step and M-step for the extended GTM can be summarized as Figure 2. Note that the effect due to content and links are aggregated in the M-step for estimating $p(z_k \mid d_j)$, which will then affect the posterior probability estimation for z_k in the E-step.

E-step:

$$
R_{ijk} = \frac{p(\boldsymbol{t}_i \mid z_k, \boldsymbol{W}, \beta) P(z_k \mid d_j)}{\sum_{k'} p(\boldsymbol{t}_i \mid z_{k'}, \boldsymbol{W}, \beta) P(z_{k'} \mid d_j)}
$$

$$
R_{ljk} = \frac{p(c_l \mid z_k) P(z_k \mid d_j)}{\sum_{k'} p(c_l \mid z_{k'}) P(z_{k'} \mid d_j)}
$$

M-step:

For content:

$$
\sum_i \sum_k \sum_l R_{ijk}(W_{old}, \beta_{old}) \{W_{new} \phi(z_k) - t_i\} \phi^T(z_k) = 0
$$

$$
\frac{1}{\beta_{new}} = \frac{1}{ND} \sum_l \sum_j \sum_k R_{ijk}(W_{old}, \beta_{old}) \| W_{new} \phi(z_k) - t_l \|^2
$$

For links:

$$
p(c_l \mid z_k) = \frac{\sum_j A_{lj} R_{ljk}}{\sum_{l'} \sum_j A_{l'j} R_{cdz}}
$$

$$
p(z_k \mid d_j) \propto \alpha \frac{\sum_i N_{ij} R_{ijk}}{\sum_k \sum_{i'} N_{i'j} R_{ijk}} + (1-\alpha) \frac{\sum_l A_{lj} R_{ljk}}{\sum_k \sum_{l'} A_{l'j} R_{ljk}}
$$

Figure 2 The EM algorithm for the extended GTM.

4. Experiments

To evaluate the proposed extension, we have first applied the extended GTM model to a small dataset based on around 100 artificially created Web pages of two different categories. As the Web pages are not linking to each other, we randomly added hyperlinks to the pages of the same category. In addition, we have also evaluated the proposed model using three data subsets extracted from the WebKB dataset. In particular, we have prepared three different subsets which consist of 10, 30 and 182 examples from each of the 3 categories of WebKB: course, department and faulty. Thus, the three subsets contain 30, 90 and 546 examples and are labeled as D30, D90, and D546 respectively.

4.1 Preprocessing

Given the data subsets, each Web page has to be pre-processed and represented as a document feature vector to be used for the subsequent model-learning step. A number of pre-processing steps have to be performed. First, all the unnecessary HTML tags and scripts should be removed. Also, the typical stop words removal and stemming steps should be followed. Lastly, only terms with their document frequencies higher than a threshold are retained. The threshold for the artificial dataset is 6 and those for the three data subsets D30, D90 and D546 are 2, 5 and 20 respectively. The distinct terms (and thus the dimension of the document feature vector) obtained for them range from 109 to 551. Then, the content related data ***T*** can readily be used for the model learning. As it is well known that the document feature vector dataset can be quite sparse, an interpolation-like process is also performed in such a way that each feature vector is replaced by an averaged version of its neighbors in the feature space.

The link related data ***A*** can easily be prepared by following the anchor tags <A> of the Web pages. The hyperlinks existed in the datasets are quite sparse. Preliminary experiments show that the help due to the incorporation of the link information is not significant. In order to amplify their effect on organizing Web pages, we, for each category, have also added hyperlinks, while leaving the inter-class links (provided by the dataset) untouched.

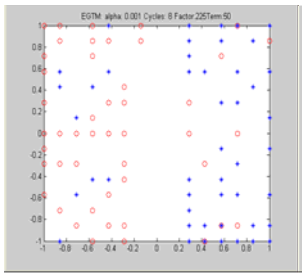
4.2 Experiment 1: Artificial Dataset

To illustrate how the proposed model can make use of within-category links to help differentiate Web pages with overlapping concepts, two FAQ Web pages on *Natural Language Processing* and *Neural Networks* are first chosen.[1] They are sub-fields under Artificial Intelligence. For each of the two pages, we randomly removed different parts to create a set of related documents forming two classes of data --- *NLP* and *NN*, 50 for each class. As expected, terms like "neural, networks, machine, artificial, computer, software" appear frequently in the class *NN*, and terms like "natural, language, linguistics, processing, speech" appears frequently in the class *NLP*. Those differences help the differentiation of the two classes. However, there are terms which are common to the two classes, such as "computer, software, science, project, university" which will sometimes confuse the classification.

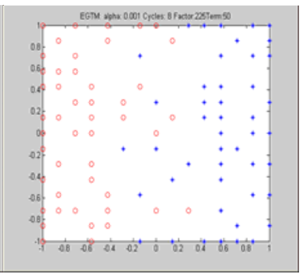
In this experiment, altogether 10 latent sample points z_i are evenly selected along the horizontal and vertical directions of a 2D latent space. Given a trained GTM, the posterior probability $p(z_k \mid \boldsymbol{t}_n)$ given a Web page, and thus the mean value of z, can be computed and visualized as a point on the latent space. The visualization of the dataset with 50 terms used for each document feature vector is shown in Figure 3. It is supposed to be manifesting the intrinsic dimension of the high-dimensional data. By varying the value of α of the extended GTM, we would like to see how the incorporation of the link information could help exploring a better mapping, and thus a better data organization in the latent space.

In Figure 3a-f, the red circles and the blue asterisks correspond to the projections of the two categories of data in the latent space with the help of the extended GTM. For α=0.001 (so data organization solely based on content), Figure 3a shows the data organization results without the use of the feature vector averaging step. We noted that the red circles and blue asterisks overlap with each other. This is more or less the original GTM setting. For the same α value, Figure 3b shows that the averaging is an effective way to improve the organization performance.

Besides properly choosing the α value, we also noted significant improvement in data organization. From Figure 3c to Figure 3f, where α is changing from 0.3 to 0.75, it can be seen that the gap between the two class boundaries is getting wider and reaches its best performance at α=0.5~0.75. When α=0.99 (Figure 3f), which means hyperlink information is dominating, the organization quality dropped. In general, this reveals that proper integration of both content and structure information can achieve better data organization quality.



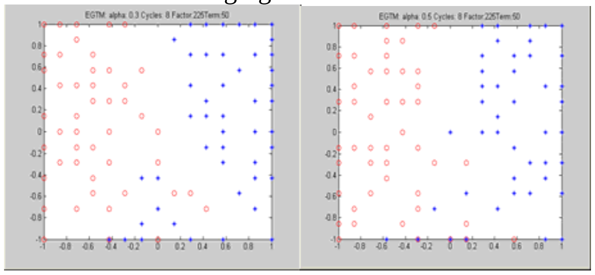
(a) α=0.001, no



(b) α=0.001

[1] The URLs of the two pages are:
- http://www-2.cs.cmu.edu/Groups/AI/html/faqs/ai/nlp/ nlp_faq/faq.html
- ftp://ftp.sas.com/pub/neural/FAQ.html

feature averaging


(c) α=0.3 (d) α=0.5

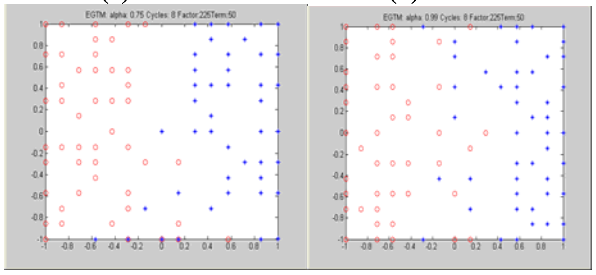

(e) α=0.75 (f) α=0.99

Figure 3 Performance comparison using the Artificial Dataset for the extended GTM with different values of α. The red circles and the blue asterisks correspond to labels for the classes NN and NLP, respectively. The dimension of the feature vector is 50 and the feature vectors are locally smoothed versions as mentioned in Section 4.1.

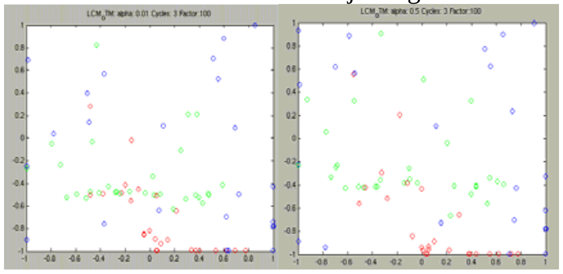
4.3 Experiment 2: WebKB Dataset

In this experiment, altogether 100 latent sample points z_i are evenly selected along the horizontal and vertical directions of the latent space. The visualization of the dataset D90 with 20 terms used for each document feature vector is shown in Figure 4.

In Figure 4a, it is noted that when α=0.01, which means the data organization is solely based on the content information, more than half of the red circles are in the bottom half of the map. Some red circles (corresponding to the student class) are mixed up with some green ones (corresponding to the faculty class), forming virtually a horizontal linear structure. So, this extended GTM fails to identify a latent space that can make the mixed-up part to be more uniformly distributed and separated. When the value of α is increased to 0.5 (Figure 4b), more emphasis is put on the link information. We can see that the linear structure spreads out and the blue circles (corresponding to the department class) move towards the edges of the map, when compared with Figure 4a.

When the value of α is further raised to 0.99, the projections of the Web pages are separated further apart but with the region of the red circles (student) and that of the green circles (faculty) overlapping with each other to a bit great extent.

We have repeated the experiments for different data subsets with different numbers of terms (10, 30, 50, 80) used for each document feature vector. Similar phenomena are observed when adjusting the value of α.


(a) α=0.01 (b) α=0.5

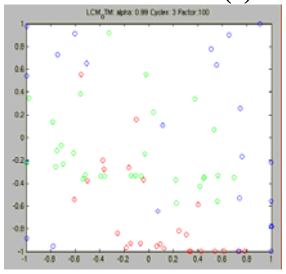

(c) α=0.99

Figure 4 Performance comparison for the extended GTM with different values of α. The red, blue and green circles correspond to labels for the classes - student, department and faculty. The dimension of the feature vector is 20.

Figure 5 shows the result of the same dataset D90 but with 50 terms used for each document feature vector instead. With the introduction of the link information, it is noted that the Web pages feature vectors are further apart from each other in the latent space with α=0.9.

All the results obtained based on the WebKB assumes that artificial links are created to make Web pages of each class fully connected. We have also tested the cases with different densities of the within-class linkage. Preliminary experimental results show that our proposed model is not very sensitive to this type of variations.

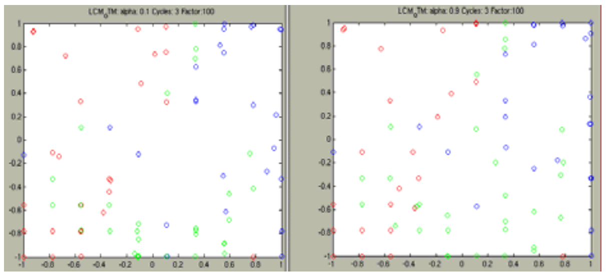

(a) α=0.1 (b) α=0.9

Figure 5 Performance comparison for the extended GTM with different values of α. The dimension of the feature vector is 50.

5. Discussion and Future Work

To summarize, it is noted that by properly choosing the value of the trade-off parameter α (between 0 to 1), the extended GTM shows some (though not very significant) improvement in identifying a better latent space for spreading out the Web page projections, when compared with the pure GTM case (i.e., α=0.01 or 0.001). We argue the limited improvement by the fact that the proposed way of incorporating hyperlinks could only help the cases with marginally similar within-class documents to be assigned to the same sample point in the latent space, but failed in grouping together, via hyperlinks, within-class documents with significantly different use of terms.

Based on our preliminary experimental results, we believe that the extended model can further be enhanced in at least the following two ways:

1. A more accurate model for representing text and links in some intrinsic latent space is needed. One possibility is to introduce an additional latent variable corresponding directly to the possible classes of the data, with the hope that the link information can help not only one particular sample point in the latent space but the group of samples points which are supposed to be within the same class.

2. An orthogonal direction for further enhancement is to represent each document as an averaged version of the feature vectors of its own as well as those linked (or cited) by it. This approach is analogous to interpolating the missing values of the term frequencies of a document using the linked documents and to some extend similar to the feature-averaging trick used in the paper.

6. Conclusion

In this paper, we have extended the GTM for automatic organization and visualization of Web pages using both the content-based and link-based information. The preliminary results show marginal improvement based on the proposed extension when compared with pure GTM. A number of important intrinsic problems of the extension as well as possible enhancement have been discussed for future research.

Acknowledgment

This research is jointly supported by Hong Kong Baptist University via Faculty Research Grant FRG/03-04/I-27 and FRG/03-04/II-20.

References

1. Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor, “Composite kernels for hypertext categorization,” *Proceedings of the 18th International Conference on Machine Learning*, pages 250--257, Williams College, US, 2001.
2. Jon M. Kleinberg, “Authoritative sources in a hyperlinked environment,” *Journal of the ACM*, 46(5):604--632, 1999.
3. Sergey Brin and Lawrence Page, “The anatomy of a large-scale hypertextual {Web} search engine,” *Computer Networks and ISDN Systems*, 30(1--7):107--117, 1998.
4. Gary William Flake, Steve Lawrence, C. Lee Giles, and Frans Coetzee, “Self-organization of the web and identification of communities,” *IEEE Computer*, 35(3):66--71, 2002.
5. Christopher M. Bishop, Markus Svensen, and Christopher K. I. Williams, “GTM: The generative topographic mapping,” *Neural Computation*, 10(1):215--234, 1998.
6. T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela, “Self Organization of a Massive Document Collection,” *IEEE Transactions on Neural Networks, Special Issue on Neural Networks for Data Mining and Knowledge Discovery*, volume 11, number 3, pages 574-585. May 2000
7. Sam Roweis & Lawrence Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, v.290, no.5500, Dec.22, 2000. pp. 2323--2326.
8. E. Pampalk and A. Rauber and D. Merkl, “Content-based Organization and Visualization of Music Archives,” *Proceedings of the ACM Multimedia*, pp.570-579, Juan les Pins, France, December, 2002
9. David Cohn and Thomas Hofmann, “The missing link - a probabilistic model of document content and hypertext connectivity,” *Neural Information Processing Systems*, 13, 2001.

A Regularization Framework for Learning from Graph Data

Dengyong Zhou DENGYONG.ZHOU@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tuebingen, Germany

Bernhard Schölkopf BERNHARD.SCHOELKOPF@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tuebingen, Germany

Abstract

The data in many real-world problems can be thought of as a graph, such as the web, co-author networks, and biological networks. We propose a general regularization framework on graphs, which is applicable to the classification, ranking, and link prediction problems. We also show that the method can be explained as lazy random walks. We evaluate the method on a number of experiments.

1. Introduction

In many real-world problems, the data can be represented as a graph. Each vertex of the graph corresponds to a datum, and the edges encode the pairwise relationships or similarities among the data. A typical example of graph data is the web. The vertices are just the web pages, and the edges denote the hyperlinks. In market basket analysis, the items also form a graph by connecting any two items which have appeared in the same shopping basket. More examples include co-author relationships, terrorists networks, biological networks and so on.

One problem addressed in this paper is classification. Specifically speaking, some of the vertices are labeled, and the task is to classify the remaining unlabeled vertices. A toy classification problem is shown in Figure 1. Two vertices of the graph are labeled as positive and negative respectively. A real-world example is to classify the web pages into different categories given some manually classified instances. Obviously, a good classifier should effectively exploit the relations among the data.

Appearing in *Workshop on Statistical Relational Learning at International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the first author.

The other problem investigated here is ranking. This problem generally can be understood as finding the vertices of most interest. For instance, given a terrorist network, detect the people who are the core of the criminal community. Another ranking problem is to find the vertices most relevant to some given vertices (Figure 3), which are often called quires. We call this problem relative ranking as distinct from the absolute ranking without quires. The example of relative ranking in terrorist networks is to discover the criminals who have strong connections to some given criminals.

A problem closely related to ranking is link prediction (Figure 3). For example, given a co-author network, predict which two scientists are most likely to collaborate in the future. Link prediction is essentially the relative ranking problem. In fact, we can compute the pairwise relevance between any two unconnected vertices, and then pick up the pair of vertices which are most relevant.

Recently there has been considerable research on these problems. Here we try to introduce a general regularization framework on graphs as a new approach to these problems. Our idea is very simple. First develop discrete calculus on graphs, and then naturally shift classical regularization from the continuous case to graph data.

2. Regularization Framework

A graph $\Gamma = (V, E)$ consists of a set V of vertices and a set of pairs of vertices $E \subseteq V \times V$ called edges. A graph is undirected if for each edge $(u, v) \in E$ we also have $(v, u) \in E$. Edge e is incident on vertex v if e contains v. Suppose that Γ is connected, i.e., there is a path from every vertex to every other vertex. Suppose further that Γ has no self-loops or multiple edges. A graph is weighted if it is associated with a function

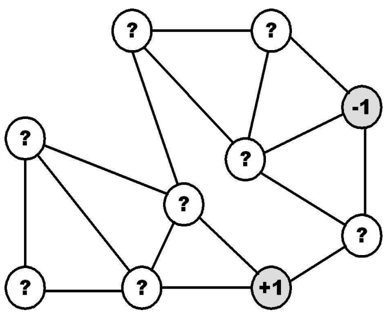


Figure 1. Classification problem on a graph. Two vertices are labeled as $+1$ and -1 respectively. The goal is to classify the remaining unlabeled vertices.

$w : V \times V \to \mathbb{R}$ satisfying

$$w(u,v) > 0, \text{ if } (u,v) \in E, \tag{2.1}$$

and

$$w(u,v) = w(v,u). \tag{2.2}$$

The degree function $d : V \to \mathbb{R}$ is defined to be

$$d(v) = \sum_{u \sim v} w(u,v), \tag{2.3}$$

where $u \sim v$ denotes all vertices u connected to v by the edges (u,v).

Let $L^2(V)$ denote the Hilbert space of real-valued function $f : V \to \mathbb{R}$ endowed with the usual inner product

$$\langle f, g\rangle = \sum_v f(v)g(v), \tag{2.4}$$

i.e., the functions are thought of as column vectors.

Let e denote the edge between vertices u and v. The *edge derivative* of function f along e at the vertex u is defined to be

$$\left.\frac{\partial f}{\partial e}\right|_u = \sqrt{\frac{w(u,v)}{d(u)}} f(u) - \sqrt{\frac{w(u,v)}{d(v)}} f(v). \tag{2.5}$$

See Appendix A for the reason of adopting such a definition. Clearly

$$\left.\frac{\partial f}{\partial e}\right|_u = -\left.\frac{\partial f}{\partial e}\right|_v. \tag{2.6}$$

The *local variation* of f at each vertex v is then defined to be

$$\|\nabla_v f\| = \sqrt{\sum_{e \vdash v} \left(\left.\frac{\partial f}{\partial e}\right|_v\right)^2}, \tag{2.7}$$

where $e \vdash v$ denotes the set of the edges incident with vertex v. The *smoothness* of f is then naturally measured by the sum of the local variations at each vertex:

$$\mathcal{S}(f) = \frac{1}{2}\sum_v \|\nabla_v f\|^2. \tag{2.8}$$

The learning problems on graphs generally can be thought of seeking for a function f, which is smooth and simultaneously close to another given function y. This view can be formalized as the following optimization problem:

$$\arg\min_{f \in L^2(V)} \left\{ \mathcal{S}(f) + \frac{\mu}{2}\|f - y\|^2 \right\}. \tag{2.9}$$

The first term in the bracket measures the smoothness of the function f, and the second term measures its *closeness* to the given function y. The trade-off between these two terms is captured by a nonnegative parameter μ. Later we will show how this general framework fits into the learning problems introduced in Section 1.

Before solving this optimization problem, we introduce the Laplace operator $\Delta : L^2(V) \to L^2(V)$ defined by

$$(\Delta f)(v) = \frac{1}{2}\sum_{e \vdash v} \frac{1}{\sqrt{d}} \left.\left(\frac{\partial}{\partial e}\sqrt{d}\frac{\partial f}{\partial e}\right)\right|_v. \tag{2.10}$$

We can show that Δ is linear and symmetrical. In fact,

$$\begin{aligned}
(\Delta f)(v) &= \frac{1}{2\sqrt{d(v)}} \sum_{e \vdash v} \left.\frac{\partial}{\partial e}\left(\sqrt{d}\frac{\partial f}{\partial e}\right)\right|_v \\
&= \frac{1}{2\sqrt{d(v)}} \sum_{e \vdash v} \left[\sqrt{\frac{w(u,v)}{d(u)}} \left.\left(\sqrt{d}\frac{\partial f}{\partial e}\right)\right|_u \right. \\
&\quad \left. - \sqrt{\frac{w(u,v)}{d(v)}} \left.\left(\sqrt{d}\frac{\partial f}{\partial e}\right)\right|_v \right] \\
&= \frac{1}{2\sqrt{d(v)}} \sum_{e \vdash v} \sqrt{w(u,v)} \left(\left.\frac{\partial f}{\partial e}\right|_u - \left.\frac{\partial f}{\partial e}\right|_v \right) \\
&= \frac{1}{\sqrt{d(v)}} \sum_{e \vdash v} \sqrt{w(u,v)} \left.\frac{\partial f}{\partial e}\right|_v \\
&= \frac{1}{\sqrt{d(v)}} \sum_{u \sim v} \left(\frac{w(u,v)}{\sqrt{d(v)}} f(v) - \frac{w(u,v)}{\sqrt{d(u)}} f(u) \right) \\
&= f(v) - \sum_{u \sim v} \frac{w(u,v)}{\sqrt{d(u)d(v)}} f(u).
\end{aligned}$$

The last equality is usually used as the definition of the graph Laplacian in many literatures, such as [2].

It is not hard to show

$$f^T \Delta f = \mathcal{S}(f). \tag{2.11}$$

Note that this also shows that Δ is positive semi-definite.

Theorem 1. *The solution of the optimization problem (2.9) satisfies* $\Delta f + \mu(f - y) = 0$.

Proof. By Equality (2.11), we have

$$(\Delta f)(v) = \left.\frac{\partial \mathcal{S}(f)}{\partial f}\right|_v .$$

Differentiating the cost function in the bracket of (2.9) with respect to f completes the proof. $\square$

Let us introduce another operator $S : L^2(V) \to L^2(V)$ by

$$(Sf)(v) = \sum_{u\sim v} \frac{w(u,v)}{\sqrt{d(u)d(v)}} f(u). \tag{2.12}$$

Then Δ can be rewritten into

$$\Delta = I - S, \tag{2.13}$$

where I is the identity operator.

Corollary 2. *If* $y \neq 0$ *and* $\mu > 0$, *the solution of the optimization problem (2.9) is*

$$f = (1-\alpha)(I - \alpha S)^{-1} y, \tag{2.14}$$

where $\alpha = 1/(1+\mu)$.

Proof. By Theorem 1 and Equality (2.13), we have

$$(I - S)f + \mu(f - y) = 0,$$

which can be transformed into

$$f - \frac{1}{1+\mu} Sf - \frac{\mu}{1+\mu} f = 0.$$

Note the definition of α. Then

$$(I - \alpha S)f = (1-\alpha)y,$$

It is not hard to see that the eigenvalues of S are at most equal to 1. Hence $I - \alpha S$ is invertible, and we have

$$f = (1-\alpha)(I - \alpha S)^{-1} y.$$

$\square$

(2.14) is in fact the algorithm proposed by [7], where it was applied to semi-supervised learning problems with vectorial data.

For large-scale datasets, we can consider using the following iteration to compute f:

$$f(v) \leftarrow \alpha (Sf)(v) + (1-\alpha) y, \ \forall v \tag{2.15}$$

with the initial value $f(v) = y(v)$. We can intuitively understand the iteration as the process of information diffusion on graphs. In each round, every vertex updates its value by linearly combining its neighbor's current values with the initial value of itself. The positive parameter $\alpha \in (0,1)$ specifies the relative amount. It is not hard to show that the iteration converges to (2.14)[7]. The iteration can be expected to converge quickly when the graph is sparse (for instance, the web graph).

Corollary 3. *If* $y = 0$ *or* $\mu = 0$, *the non-zero solution of the optimization problem (2.9) is the principle eigenfunction of* S.

Proof. In the case of $y = 0$, by Theorem 1, we have $\Delta f + \mu f = 0$. Substituting (2.13) into the equality, then we obtain $Sf = (1+\mu)f$, which means that f is one of the eigenfunctions of S. Note that the eigenvalues of S are at most equal to 1. This requires that $1 + \mu \leq 1$. Since $\mu \geq 0$, we have $\mu = 0$. Hence f is the eigenfunction corresponding to the largest eigenvalue 1, i.e. the principle eigenfunction. $\square$

3. Lazy Random Walks

The regularization framework can be interpreted as *lazy random walks*. Let D denote the diagonal matrix with the (u,u)-entry equal to $d(u)$, and let W denote the matrix with the (u,v)-entry equal to $w(u,v)$ if $(u,v) \in E$ and 0 otherwise. A lazy random walk on graph Γ is decided by the transition probability matrix $P = (1-\alpha)I + \alpha D^{-1}W$, where α is a parameter in $(0,1)$. This means, with the probability α, following one link incident with the vertex of the current position and is chosen with the probability proportional to the weight of the link, and with the probability $1-\alpha$, just staying at the current position.

Assume there are n vertices. There exists a unique stationary distribution $\pi = [\pi_1, \ldots, \pi_n]$ for the lazy random walk, i.e. a unique probability distribution satisfying the balance equation

$$\pi = \pi P. \tag{3.1}$$

Let $\mathbf{1}$ denote the $1 \times n$ vector with all entries equal to 1.

$$\begin{aligned} \mathbf{1}DP &= \mathbf{1}D[(1-\alpha)I + \alpha D^{-1}W] \\ &= (1-\alpha)\mathbf{1}D + \alpha \mathbf{1}DD^{-1}W \\ &= (1-\alpha)\mathbf{1}D + \alpha \mathbf{1}W \\ &= (1-\alpha)\mathbf{1}D + \alpha \mathbf{1}D \\ &= \mathbf{1}D. \end{aligned}$$

Let vol Γ denote the volume of the graph, which is defined to be the sum of vertex degrees. Then the stationary distribution can be written as

$$\pi = \mathbf{1}D/\text{vol}\,\Gamma. \tag{3.2}$$

Note that π does not depend on α. Hence π is also the stationary distribution of the random walk with the transition probability matrix $M = D^{-1}W$. In addition, the matrix $S = D^{-1/2}WD^{-1/2}$ can be rewritten in terms of stationary distribution:

$$S(u,v) = \pi_u^{1/2} M(u,v) \pi_v^{-1/2}, \tag{3.3}$$

which also shows that S is similar to M.

Let X_t denote the position of the random walk at time t. Write $T(u,v) = \min\{t \geq 0 | X_t = v, X_0 = u, u \neq v\}$ for the *first hitting time* to v with the initial position u, and write $T(v,v) = \min\{t > 0 | X_t = v, X_0 = v\}$, which is called the *first return time* to v [1]. Let $H(u,v)$ denote the expected number of steps required for a random walk to reach v with an initial position u, i.e. $H(u,v)$ is the expectation of $T(u,v)$. $H(u,v)$ is called the *hitting time* [1]. Let $C(u,v)$ denote the expected number of steps for a random walk starting at u to reach v and then return, i.e. $C(u,v) = H(u,v) + H(v,u)$. $C(u,v)$ is called the *commute time* between u and v. Clearly, $C(u,v)$ is symmetrical, but $H(u,v)$ may be not. Let G denote the inverse of the matrix $D - \alpha W$. For distinct vertices u and v, the commute time satisfies [3]:

$$C(u,v) \propto G(u,u) + G(v,v) - G(u,v) - G(v,u), \tag{3.4}$$

and [1]

$$C(u,u) = 1/\pi(u). \tag{3.5}$$

The relation between G and C is similar to the inner product and the norm in Euclidean space. In other words, we can think of G as a Gram matrix which specifies a kind of inner product on the dataset. The commute time is the corresponding metric derived from this inner product [3].

Note that $H(u,v)$ is quite small whenever v is a node with a large stationary probability $\pi(v)$. Thus we can consider normalizing $H(u,v)$ by

$$\bar{H}(u,v) = \sqrt{\pi(u)\pi(v)}H(u,v). \tag{3.6}$$

Accordingly, the normalized commute time is

$$\bar{C}(u,v) = \bar{H}(u,v) + \bar{H}(v,u). \tag{3.7}$$

Let $\bar{G}$ denote the inverse of the matrix $I - \alpha S$. Then the normalized commute time satisfies

$$\bar{C}(u,v) \propto \bar{G}(u,u) + \bar{G}(v,v) - \bar{G}(u,v) - \bar{G}(v,u). \tag{3.8}$$

Noting Equality (3.5), we have

$$\bar{G}(u,v) = \frac{G(u,v)}{\sqrt{C(u,u)C(v,v)}}, \tag{3.9}$$

which is similar to the normalized Euclidean product or cosine.

4. Bayesian Interpretation

There is a simple Bayesian interpretation for the regularization framework inspired by [6]. Let $p(f)$ denote the prior probability of f, and let $p(y|f)$ denote the conditional probability of y given f. Then the MAP estimation is given by

$$\arg\max_{f\in L^2(V)} \left\{ \log p(y|f) + \log p(f) \right\}. \tag{4.1}$$

A general model for the prior distribution $p(f)$ is given by

$$p(f) = \frac{1}{Z_r} \exp\left[-\frac{\mathcal{S}(f)}{\mu} \right], \tag{4.2}$$

where Z_r is a normalization constant. Further, the conditional probability is given by

$$p(y|f) = \frac{1}{Z_c} \exp\left(-\frac{\|f - y\|}{2\sigma^2} \right), \tag{4.3}$$

where Z_c is another normalizing constant. Thus the MAP estimator (4.1) yields

$$\arg\min_{f\in L^2(V)} \left\{ \mathcal{S}(f) + \frac{\mu}{2}\|f - y\|^2 \right\}. \tag{4.4}$$

5. Applications

In this section, we apply the general framework to the learning problems introduced in Section 1, including classification, relative ranking, and link prediction. In all experiments, the regularization parameter α is simply fixed at 0.90. In our future research, we will address how to choose a suitable α.

5.1. Classification

Assume partial vertices are labeled as positive or negative. We want to predict the labels of the other vertices. Define $y(v) = 1$ or -1 if u is labeled as positive or negative and 0 otherwise. Then compute $f = (I - \alpha S)^{-1}y$, and classify the unlabeled vertices v as $y(v) = \text{sgn}(f(v))$.

We applied this classification method to the toy problem shown in Figure 1 and obtained the result shown

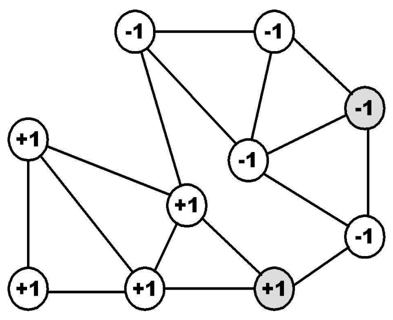


Figure 2. Classification on a toy graph. The two shaded circles are the initially labeled vertices. Note that the nodes can not be classified correctly if the classification only naively depends on the shortest paths to the labeled nodes.

in Figure 2. Apparently, the classification is consistent with our intuition. If we simply classify the unlabeled vertices by comparing the minimal distances from them to the labeled vertices, i.e., the length of the shortest paths, then some vertices can not be correctly classified.

5.2. Relative Ranking

Given a vertex (query) of interest in a graph, rank the remaining vertices with respect to their relevances to the given vertex. This can be viewed as an extreme case of classification problem, in which only positive examples are available. In other words, in some sense, relative ranking can be regarded as one-class classification problem [5]. Hence, similarly, define y with $y(v) = 1$ if v is the query and 0 otherwise and compute $f = (I - \alpha S)^{-1}y$. Then rank the vertices v according with $f(v)$ (largest ranked first) [8].

We address a toy ranking problem shown in the top panel of Figure 3. This toy network was first suggested by [4] for the problem measuring betweenness centrality in social networks. Assume that A is the query. The task is to rank the remaining vertices with respect to A. According with our intuition, D should be most similar to A because they are in the same *group*. Next should be B. Let us imagine that the vertices respectively in the left and right groups communicate with each other by passing their messages through the links. Then both A and B will be quite busy exchanging messages from the two opposite groups. Next C, which can be regarded as a redundant vertex in the context of communication. Hence the idea ranking list should be $A \to D \to B \to C \to E$. If we simply rank the vertices according with shortest paths, then the vertices D, B and C are similar to A at the same level. The ranking cores given by our method is as follows:

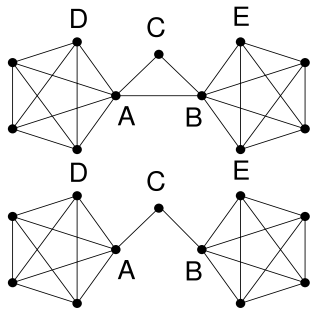


Figure 3. Ranking on a toy network. Top panel: ranking the vertices according to their relevances to the vertex A. Bottom panel: link prediction problem, in which the edge between the vertices A and B is removed.

	B	C	D	E
A	0.99	0.87	1.33	0.56

This ranking list is consistent with our intuitive analysis.

5.3. Link Prediction

This problem is essentially as same as ranking: choosing each vertex as the query and ranking the other vertices with respect to it, then adding a link between two most relevant vertices. This is equivalent to compute the matrix $(I - \alpha S)^{-1}$ and choose the entry which corresponds to the maximal value.

We investigate the toy network shown in the bottom panel of Figure 3, in which the link between A and B is removed with respect to the network shown in the top panel. The goal is to predict this removed link.

The relevance scores among unconnected vertices are the following:

(A, B)	(A, E)	(D, B)	(D, E)	(C, E)
0.48	0.29	0.29	0.18	0.52

This means the most possible link in the future is between C and E (and symmetrically the link between C and D as well). Unfortunately, this is not consistent with our setting. However, if we choose $\alpha = 0.95$, then

(A, B)	(A, E)	(D, B)	(D, E)	(C, E)
1.27	0.93	0.93	0.69	1.16

Now the next link predicted by the scores is between A and B. Different from the previous experiments, the result is sensitive to the choice of α. This shows how to choose a suitable α is a very important problem.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments.

A. Laplacian in Euclidean Space

Let f denote a differentiable function defined on $\mathbb{R}^m$. Then the gradient of function f at point $x = [x_1, \dots x_m]^T$ is the vector

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_m}\right]^T,$$

and

$$\|\nabla f(x)\| = \sqrt{\sum_{i=1}^{m}\left(\frac{\partial f}{\partial x_i}\right)^2}.$$

The smoothness of f is generally measured by

$$\mathcal{S}(f) = \frac{1}{2}\int \|\nabla f\|^2 dx,$$

which is usually called the *Dirichlet form* or *energy function.*

The Laplacian in the continuous case is a second-order differential operator defined by

$$\Delta f = \sum_{i=1}^{m} \frac{\partial^2 f}{\partial^2 x_i}.$$

The connection between the Laplacian and the gradient is expressed by

$$\int f(\Delta f) dx = \int \|\nabla f\|^2 dx,$$

which is exactly parallel to (2.11). In this sense, it is reasonable to think of (2.5) as the derivative on graphs and further use (2.8) to measure the smoothness of functions.

References

[1] D. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs.* In Preparation, http://stat-www.berkeley.edu/users/aldous/RWG/book.html.

[2] F. Chung. *Spectral Graph Theory.* Number 92 in Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[3] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[4] M. Newman. A measure of betweenness centrality based on random walks. Preprint cond-mat/0309045, 2003.

[5] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[6] G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* SIAM, 1990.

[7] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT press.

[8] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT press.