View Learning Extended: Inventing New Tables for Statistical Relational Learning

Jesse Davis JDAVIS@CS.WISC.EDU
Department of Computer Science, University of Wisconsin, Madison, WI 53705 USA

Elizabeth Burnside ES.BURNSIDE@HOSP.WISC.EDU
Department of Radiology, University of Wisconsin, Madison, WI 53705 USA

David Page PAGE@BIOSTAT.WISC.EDU
Department of Biostatistics and Medical Informatics, University of Wisconsin, Madison, WI 53705 USA

Vítor Santos Costa VITOR@COS.UFRJ.BR
COPPE/Sistemas, UFRJ Centro de Tecnologia, Bloco H-319, Cx. Postal 68511 Rio de Janeiro, Brasil

Abstract

Statistical relational learning (SRL) algorithms learn statistical models from relational data, such as that stored in a relational database. Last year saw the definition of *view learning* for SRL, in which the view of a relational database can be automatically modified, yielding more accurate statistical models. The present paper advances beyond the initial view learning approach in two ways. First, it learns views that introduce new relational *tables*, rather than merely new fields for an existing table of the database. Second, new tables or new fields are not limited to being approximations to some target concept; instead, the new approach performs a type of predicate invention. The new approach avoids the classical problem with predicate invention, of learning many useless predicates, by keeping only new fields or tables (i.e., new predicates) that immediately improve the performance of the statistical model. Retained fields or tables can then be used in the definitions of further new fields or tables.

Presented at the ICML Workshop on Open Problems in Statistical Relational Learning, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

1. Introduction

Statistical relational learning (SRL) algorithms learn statistical models from relational data, such as that stored in a relational database. SRL can be seen as a way to permit statistical models to reason about sets of related objects. Conversely, SRL can be seen as upgrading logic to handle the inherent uncertainty present in the world. Despite these advances over ordinary statistical and ordinary relational learning, SRL techniques are still constrained to use only the tables and fields already in the database, without modification. In contrast, many human users of relational databases find it beneficial to define alternative *views* of a database—further fields or tables that can be computed from existing ones. Last year saw the introduction of *view learning* for SRL, in which the *view* of a relational database can be automatically modified, yielding more accurate statistical models (Davis et al., 2005b). Despite its benefits, the original formulation of *view learning* falls short of its full potential.

The present paper advances beyond the initial view learning approach in two major ways. First, it learns new views that involve new *relational tables*, rather than merely new fields for an existing table of the database. Second, new tables or new fields are not limited to being approximations to some target concept; instead, the new approach is performing a type of *predicate invention* or *constructive induction.* Our approach learns new predicates by performing a search over the bodies of definite clauses. Within each clause body, we pick a set of "distinguished" variables, which will appear in the head of the clause. We evaluate each potential predicate by aggregating away a subset of the distinguished variables and introducing the predicate as a feature into the SRL model. We call this new approach VISTA (View Invention by Scoring Tables through Aggregation). VISTA avoids the classical problem with constructive induction or predicate invention—learning too many useless predicates— by keeping only new fields or tables (i.e., new predicates) that immediately improve the performance of the statistical model. Retained fields or tables can then, in turn, be used in the definitions of further new fields or tables. We present results on three real world applications: social networks (UW-CSE), citation matching (Cora) and cancer prediction (Mammography).

2. Motivating Example

The original motivation for view learning centered on learning a statistical expert system to provide decision support to radiologists (Davis et al., 2005b). Following the authors of that work, we motivate the extended version of view learning using the mammography task. The authors of that work used SRL because the learned statistical model sits on top of the National Mammography Database (NMD) schema, a standard established by the American College of Radiology (ACR, 2004). View learning automatically augmented the NMD schema by adding fields learned using inductive logic programming (ILP). The new intensionally-defined fields improved the prediction, by the SRL model, of which mammogram abnormalities were malignant. The following is an example of an ILP-induced rule defining a new field.

```
malignant(A) if:
    density(A,D1),
    prior-abnormality-same-location(A,B),
    density(B,D2),
    D1 > D2.
```

Here, `prior-abnormality-same-location` is a predicate that is true of a pair of abnormalities `A` and `B` in the same location on the same patient, where `B` occurred on an earlier mammogram than `A`. Therefore, the rule says that an abnormality is malignant if there was an earlier abnormality at the same location, such that the current abnormality has a greater density than the previous abnormality. Of course, the rule may not always be true; this change might occur and yet the abnormality may not be malignant. But the rule results in a field being added to every abnormality record, where this field has value 1 for an abnormality `A` if the body (condition) of the rule is true of `A`, and value 0 otherwise. Even if the rule is not universally true, it may provide predictive value for the final SRL model. It is worth noting that, in this use of a rule to add an additional field, the only purpose really being served by the head (consequent) of the rule is to distinguish `A` as the variable that links us to any particular abnormality. Variable `A` holds the abnormality key; to compute the value of new field for any particular abnormality record, one substitutes that abnormality's key for variable `A` and checks whether the body (condition) of the rule holds.

From this discussion, we see that we can change the head of a rule in any way we like without damaging our ability to use the rule. All we really need in order to use the rule to augment the database are: (1) the body of the rule, and (2) a "distinguished" variable or variables in the rule body that will hold the keys to records in the database. Thus for example, instead of the earlier rule, we would get an identical result with the following rule, where `A` is selected as the distinguished variable denoting the key.

```
density-increase(A,B) if:
     density(A,D1),
     prior-abnormality-same-location(A,B),
     density(B,D2),
     D1 > D2.
```

This second rule actually defines a new relational table relating pairs of abnormalities for the same patient. Note that a new binary predicate, such as density-increase, is an intensionally defined relational table in database terminology. Most tables learned by VISTA, such as density-increase, represent many-to-many relationships between their arguments and so cannot be captured by merely adding a new field to an existing table.

An advantage of the VISTA approach is that `density-increase`, once it is learned, can appear in the bodies of other rules. For example, this predicate could be used in another rule that flags abnormalities that have increased in size and changed shape. This latter rule could be learned directly, without the new density-increase predicate, but its length without using the new `density-increase` predicate makes it unlikely to be found by typical ILP algorithms, since these algorithms examine shorter clauses first.

Learning new predicates, such as `density-increase`, is known as *predicate invention* (Muggleton & Raedt, 1994). A particularly difficult aspect of predicate invention is that many possible predicates can be invented. Consequently, the new predicates cause the search space for clauses to grow radically and can cause overfitting. As a result, predicate invention actually can hurt performance. A unique aspect of predicate invention as proposed within VISTA is the constraint to prevent invention of arbitrarily many new predicates. The constraint already noted is that the invented predicates must themselves be immediately useful; for one of the variables—e.g., the distinguished variable A in the clause above—the resulting new feature must improve the precision-recall area (or some other score metric) of the SRL model.

To further constrain the invented predicates, we keep a bound of two or three on the number of arguments in the head of any clause, and hence on the arity of invented predicates. We also require that the arguments in the head correspond to database keys, i.e., IDs of abnormalities (as with `A` and `B` above), patients, mammograms or physicians.

3. Learning New Predicates

VISTA learns new predicates by performing a search over the bodies of definite clauses and selecting those bodies that improve the performance of the statistical model on a classification task. We use Tree Augmented Naive Bayes (TAN) (Friedman et al., 1997) as our statistical model. This section motivates and defines the VISTA algorithm; we will use the Mammography domain to help illustrate key points about our algorithm.

3.1. Algorithm Overview

The VISTA algorithm implements greedy search.

- *Initialize*: in our experiments this is performed by initializing a Bayesian network to empty
- While there is time:
    1. Randomly select an arity of predicate to invent, and randomly select types for that arity
    2. *Generate* and *evaluate* clauses until *termination-condition*
    3. If a good clause was found (2% improvement in precision-recall area in our experiments), add it to the network

Next we discuss these steps in more detail. The *Generate* clause algorithm implements a top-down, breadth-first refinement search. The space of candidate literals to add during refinement is defined using modes, as in Aleph (Srinivasan, 2001). This clause search *terminates* in four cases: **(i)** a good clause is found; **(ii)** the search space was fully explored; **(iii)** the clause limit was exceeded; **(iv)** the global time limit was exceeded.

We evaluate clauses by how much they improve the statistical classifier, more specifically the Bayesian network. Each clause is introduced as a binary variable into the network, just as in the recent algorithms nFOIL (Landwehr et al., 2005) and SAYU (Davis et al., 2005a). In nFOIL and SAYU, the head of a clause has the same predicate and arity as the example, allowing us to precisely define whether a clause succeeds for a given example and hence whether the corresponding variable is true. For an illustration from Mammography, a positive example has the form `malignant(ab1)`, where `ab1` is a primary key for some abnormality. Every learned rule has the head `malignant(A)` as in the rule we saw earlier:

```
malignant(A) if:
    density(A,D1),
    prior-abnormality-same-location(A,B),
    density(B,D2),
    D1 > D2.
```

The Bayesian network variable corresponding to this rule will take value *true* for the example `malignant(ab1)` just if the clause body succeeds when the logical variable `A` is bound to `ab1`. Recall that in VISTA we may have a clause head such as `density-increase(A,B)` instead of `malignant(A)`. In this case, it is less clear how to match an example, such as `malignant(ab1)`, to the head. VISTA maps, or links, one argument to the example key and aggregates away any remaining arguments. While many natural options exist for performing aggregation and linkage, we choose simple approaches which we describe in the following two sections.

3.2. Aggregation

In order to transform an invented predicate into a feature in our statistical model, we must perform aggregation. VISTA currently implements two aggregation operators: `exists` and `count`. The `exists` operator can be implemented by simply testing whether the clause is satisfiable. The `count` operator counts the number of bindings for free variables with which the clause succeeds. For simplicity, we focus on aggregating over a single attribute: if the key corresponds to the first $N-1$ arguments, we aggregate over the last, Nth argument.

To illustrate the `exists` operator, consider again the following clause:

```
density-increase(A,B) if:
     density(A,D1),
     prior-abnormality-same-location(A,B),
     density(B,D2),
    D1 > D2.
```

In this clause variable `A` represents the more recent abnormality. Suppose we wish to create a feature for this clause, using existence aggregation. The feature is true for a given binding of `A` if there exists a binding for `B` that satisfies the body of the clause. Specifically, for an example `malignant(ab1)`, this `density-increase` feature is true if there exists another abnormality `ab2` such that `density-increase` is true of the tuple ⟨`ab1,ab2`⟩.

Using the same clause and same example abnormality `ab1`, we now turn to the `count` operator. In this case, we are interested in the number of solutions for `B` given that `A` is set to `ab1`. This means that the new feature we will propose is not binary. Currently, VISTA discretizes aggregated features using a binning strategy that creates three equal-cardinality bins, where three was chosen arbitrarily before the running of any experiments.

Aggregation queries are, in general, more expensive to compute than standard queries, as we may need to compute all solutions, instead of simply proving satisfiability. Thus, using aggregated views when inventing new views can be very computationally expensive (in fact, we can aggregate over aggregates). To address this problem, whenever VISTA learns an aggregated view, VISTA does not store the learned intensional definition of the view. Instead, VISTA materializes the view, that is, computes the model and stores the logical model as a set of facts. This solution consumes more storage, but it makes using aggregated views as efficient as using any other views.

3.3. Linkage

So far we have simplified matters by assuming that the first argument to the learned predicate has the same type as the example key. In our examples so far, this type has been *abnormality id.* There is no need to enforce this limitation. For example, in predicting whether an abnormality is malignant, it might be useful to use the following clause, where `Visit` is a key that refers to all abnormalities found on a given mammogram:

```
p(Visit) :-
   visit(Visit,Ab),
   MassesShape(Ab,oval).
```

Predicate `p` is true of a visit, or mammogram, that contains at least one abnormality with an oval shape.

Linkage declarations are background knowledge that establish the connection between objects in the examples and objects in the newly invented predicates. When these objects are of the same type, the linkage is trivial; otherwise, it must be defined. For mammography, we use linkage definitions that link an *abnormality* to its *patient* or to its *visit* (mammogram). The linkages for Cora and UW-CSE are equally straightforward and are defined when we describe the datasets.

4. Data and Methodology

UW-CSE. This common SRL dataset was constructed by Richardson and Domingos (2006) and is publicly available. The goal is to predict the advisor of a graduate student. The information comes from the University of Washington CS Department and contains 113 positive examples versus 2,711 negative examples. We defined *students*, *professors*, *courses* and *publications* as keys that could appear in the head of a clause. We link a courses to a graduate student by the TA relationship, and we link papers to a graduate student by the author relationship. We link a course to a professor by the teaches relationship and we link papers to a professor by the author relationship. We aggregate over students, professors, papers and courses.

Cora. The objective of this dataset is to predict whether two citations refer to the same paper. The dataset was originally constructed by McCallum et al. (2000). We used the same version of the data as Kok and Domingos (2005). Cora includes 1295 citations to 112 Computer Science papers, resulting in 25072 positive examples and 597310 negative examples. The background knowledge includes data on title, venue, author(s), and year for each citation. We defined *paper*, *title*, *venue*, *author* and *year* as keys that can appear in heads of clauses. We link a paper to its title, venue, author(s) and year fields. We aggregate over papers and authors.

Mammography. The objective of this dataset is to predict whether an abnormality on a mammogram is benign or malignant (Davis et al., 2005b). This dataset consists of a radiologist's interpretation of a mammogram and not the raw image data. The dataset contains 435 positive examples and 65365 negative examples. We used the same version of the data as Davis et al. (2005b). We define *abnormality*, *visit* and *patient* as keys that can appear in the head of the clause. We aggregate over abnormalities.

	SAYU	VISTA	p-value
Mammography	0.10337	0.1038	0.9693
Cora	.36581	.44556	$3.55{*}10^{-6}$
UW-CSE	0.097466	0.16721	0.05807

Table 1. Average AUCPR for recall $\geq$ 0.5 for each task using TAN as the statistical model.

5. Experiments and Results

The state-of-the-art view learning implementation is the SAYU algorithm (Davis et al., 2005a), a follow-up to the original view learning paper (Davis et al., 2005b). Therefore, we compare VISTA to SAYU in our experiments. SAYU only learns additional fields for existing tables; these fields are defined by learned rules that are approximations to the target concept.

We used the same methodology for learning with SAYU and VISTA. We use a training set to learn the network parameters, while we use a tuning set to score potential clauses. For all datasets we use area under the precision-recall curve (AUCPR) as our score metric. However, we only look at AUCPR for recalls $\geq$ 0.5. We do this for two reasons. First, precision can have high variance at low levels of recall. Second, in domains such as mammography, we are only interested in high levels of recall. A practicing radiologist would need to achieve at least this level of recall. A clause must improve the AUCPR by at least 2% in order to be retained in the network. This is an arbitrary parameter setting; in fact we did not try any other thresholds. We had a time-based stop criteria for all our experiments. To offset potential differences in computer speeds, all experiments were run on identically configured machines.

UW-CSE. Following Richardson and Domingos (2006), we perform five-fold cross validation on the UW-CSE dataset. We used two folds for the training set and two folds for a tuning set. We gave each fold two hours of run time. Each approach could evaluate up to 10000 clauses, before either selecting a new seed (SAYU) or a new predicate head. Let us examine two predicates learned by VISTA:

```
p16(Student,Professor):-
   ta(Course,Student,Date),
   taughtby(Course,Professor,Date).

p22(Student,Professor,Paper):-
   publication(Paper,Student),
   publication(Paper,Professor).
```

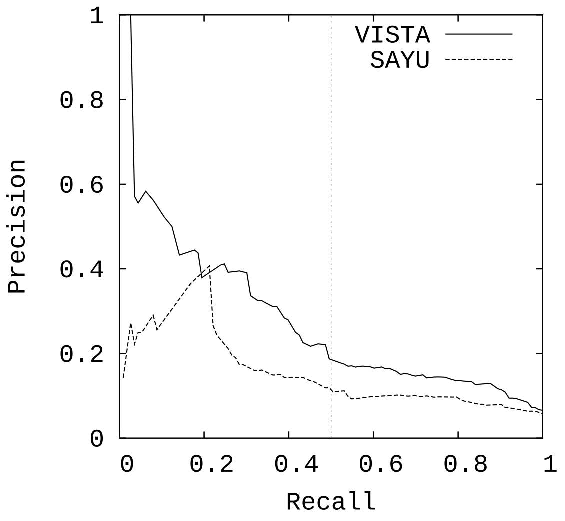

Figure 1. Precision-Recall Curves comparing VISTA and SAYU on UW-CSE

Both `p16` and `p22` capture intuitive knowledge about the advisee-advisor relationship. First, a graduate students often TA's a class taught by their advisor. Second, a graduate student will usually co-author a paper with their advisor. Predicate `p22` makes use of VISTA's ability to learn a predicate with a higher arity than the target predicate.

Table 1 reports the average AUCPR for UW-CSE and the p-value for a two-tailed paired t-test between the two algorithms. VISTA comes close to performing significantly ($0.05 < p < 0.06$) better than SAYU on this domain. Although performance varies widely between the 5 folds, VISTA had a higher AUCPR than SAYU on each fold. Figure 1 shows precision-recall curves for both algorithms on this dataset. We pooled results across all five folds to generate the curves. Even though we measured AUCPR for recall $\geq$ 0.5, VISTA dominates SAYU for most levels of recall.

Cora. Following Kok and Domingos (2005) we perform two-fold cross validation on this dataset for five different random train-test splits. We divide the training set in half, to form a new train set and a tuning set. Each fold received three hours of CPU time to run. SAYU and VISTA can evaluate up to 300 clauses, before a selecting a new seed or clause head. Let us discuss two predicates found by VISTA on the Cora dataset:

```
p1(Venue1,Venue2):-
   commonWordsInVenue80(Venue1,Venue2).

p4(Paper1,Paper2,Paper3):-
```

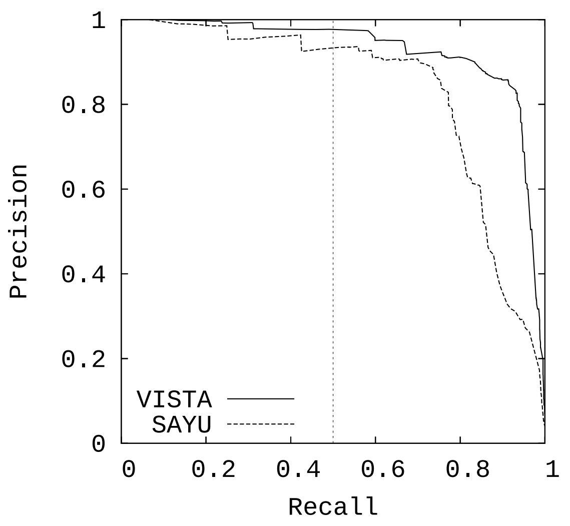

Figure 2. Precision-Recall Curves comparing VISTA and SAYU on Cora

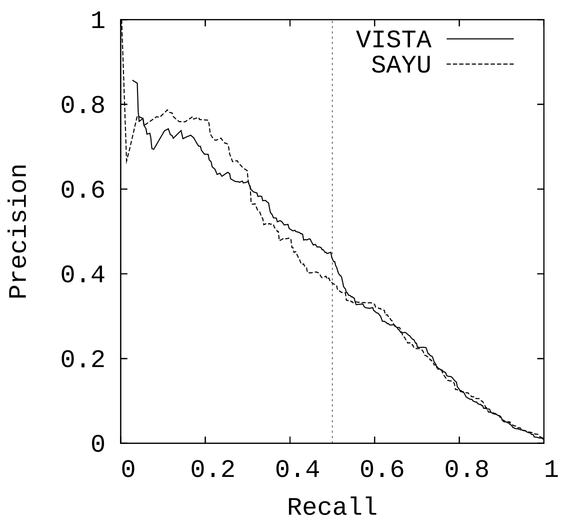

Figure 3. Precision-Recall Curves comparing VISTA and SAYU on Mammography

```
   paperTitle(Paper2,Title),
   paperTitle(Paper1,Title),
   paperTitle(Paper3,Title).
```

Predicate p1 captures a crucial piece of partial knowledge in the citation matching domain by checking how similar the venue field is between two citations. This rule demonstrates how VISTA can learn predicates whose "distinguished variables" have a different type than "distinguished variables" in the target relation. Predicate p4 establishes a transitive relationship between papers with exactly the same title.

Table 1 reports the average AUCPR for Cora and the p-value for for a two-tailed paired t-test between the two algorithms. VISTA performs significantly better than SAYU on this domain. Figure 2 shows precision-recall curves for both algorithms on this dataset. We pooled results across all folds to generate the curves. Again, VISTA dominates SAYU throughout precision-recall space.

Mammography. Following Davis et al. (2005b) we perform ten-fold cross validation on this dataset. We used four folds for a training set and five folds as a tuning set. We allow each fold three hours of CPU time to run. Each algorithm can evaluate at most 300 clause for a given seed (SAYU) or clause head (VISTA). Here is a sample predicate found by VISTA:

```
p23(Ab1,Ab2):-
   aggregate(Ab2,count,same_loc(Ab1,Ab2))
```

This predicate counts the number of prior abnormalities that were in the same location as the current abnormality. This predicate displays VISTA's ability to learn clauses that compute aggregate information about prior abnormalities.

For the previous two datasets, we initially started with a Bayesian network that only contained a feature for the target predicate. However, in the mammography domain we have access to a set of expert defined features (from the NMD). Furthermore, we could define a set of aggregate features as Davis et al. (2005b) did. Opposed to starting with an empty network structure, we begin with a network that contained both NMD features and the aggregate features.

Table 1 reports the average AUCPR over all folds. We use a two-tailed paired t-test to compute significant results, and the p-value for the test can also be found in the Table 1. We find no significant difference between VISTA and SAYU on this task. However VISTA does not perform any worse than SAYU. In this case, SAYU receives a large number of features—the precomputed aggregates—that VISTA could potentially learn, but SAYU cannot capture. Furthermore, by leveraging Aleph, SAYU has more directed search. Figure 3 shows precision-recall curves for both algorithms on this dataset. We pooled results across all folds to generate the curves. On this dataset, VISTA and SAYU have comparable performance for all levels of recalls.

Reuse of invented predicates. Over the three tasks, VISTA invented a total of 513 predicates. Of those 513 predicate definitions, 89 of them (17%) reused at least one of the other invented predicates. In total, we saw 118 reuses of previously invented predicates, meaning that 29 predicate definitions contained multiple previously-invented predicates.

6. Relationship to Other Work

We already have discussed how the present paper moves beyond the previous work on view learning (Davis et al., 2005b). The most-closely related work we know about is that of Popescul and Ungar on Structural Logistic Regression (Popescul et al., 2003). Their work—like the present paper—constructs new predicates that can be used by a statistical learning algorithm. They too “aggregate away” some of the variables within predicate definitions. The significant difference is that their learned predicates are not available for use within the definitions of further learned predicates; rather, their learned predicates serve solely as additional features to logistic regression. An extension to their approach—Cluster-based Concept Invention for SRL (Popescul & Ungar, 2004)—does in fact construct new predicates that can be used within the definitions of learned predicates. But these new predicates are based on clustering and are constructed in an initial pre-processing step, before the learning of predicates to define features for logistic regression. Nevertheless, it remains the case that the predicates learned in order to provide features to logistic regression still cannot be used in the definitions of further such learned predicates.

Other major areas of related work are of course constructive induction (Rendell, 1985), predicate invention (Muggleton & Buntine, 1988; Zelle et al., 1994), and propositionalization within ILP (Lavrac et al., 1991). Predicate invention is a specific type of constructive induction, where a new predicate is defined not based directly on examples of that predicate, but on the ability of that predicate to help in learning the definitions of other predicates for which examples are available. The classic difficulties with predicate invention are that, unless predicate invention is strongly constrained: (1) the search space of possible predicates is too large, (2) too many new predicates are retained, thus reducing efficiency of learning, and (3) the ability to invent arbitrary new predicates leads to overfitting of training data. The approach in the present paper is to constrain predicate invention by requiring invented predicates to be of immediate value to the statistical learner in order to be retained for further use.

Propositionalization is related to the present paper in that many propositionalization approaches use learned rules to define new propositional features. The present work moves beyond propositionalization in two major ways. First, the present work learns rules that define not merely new fields of the database—which can be thought of as propositions—but new tables, or new predicates of arity greater than one. Second, once learned the new predicates are available for use in the definitions of additional learned predicates, as noted earlier in this section.

Finally, we draw on the idea of aggregation in SRL. Perlich and Provost discuss several approaches for attribute construction using aggregates over multi-relational features (Perlich & Provost, 2003). Vens et al. incorporate complex, aggregate queries into the relational decision tree learner TIDLE (2004). However, the aggregate functions we consider during learning are not as complex as those learned by either Perlich and Provost or Vens et al.

7. Conclusion

SRL algorithms can construct probabilistic models from relational databases. A key capability of SRL is the learning of arcs (in the Bayes net sense) connecting entries in different rows of a relational table, or in different tables. Nevertheless, most SRL approaches currently are constrained to use only the existing database schema. View learning (Davis et al., 2005b) provides the capability to automatically change the schema, or more specifically, define a new view of the database. Nevertheless, the previous approach to view learning could not define new tables for the database schema, but only new fields for existing tables in the schema. This paper has presented an SRL algorithm, VISTA, that can learn new views that include new relational tables. View learning in this general case is isomorphic to predicate invention, a type of constructive induction investigated within ILP.

As with predicate invention, the space of new views one can define for a given relational database is vast, leading to problems of overfitting and search complexity. VISTA constrains this space by

- learning definitions of new relations (tables or fields) one at a time
- considering only new relations that can be defined by short clauses expressed in terms of the present view of the database (including background knowledge relations provided as intensional definitions)
- re-constructing the SRL model when testing each potential new relation, and keeping a new relation only if the resulting SRL model significantly outperforms the previous one

The last step requires matching a subset of the arguments in the relation with the arguments in the data points, or examples, and aggregating away the remaining arguments in the relation. VISTA moves beyond prior work on view learning by (1) inventing new relational tables rather than only new fields for existing tables, and (2) treating these new tables or fields as "first-class citizens of the database," that is, as relations that can be used in the definitions of further new tables or fields. Our experimental results indicate that VISTA invents interesting relations and performs as well as, or significantly better than, the prior view learning approach on challenging SRL tasks.

A number of parameter settings and design decisions within VISTA were selected arbitrarily, to avoid biasing experimental results, because time did not permit tuning or trying different designs on subsets of the data. VISTA's performance probably can be improved, therefore, by testing variations in the design and by tuning parameters. Plans for such future work include using best-first search rather than breadth-first search over intensional definitions of relations, testing other aggregation operators and other types of linkages between key types, testing other types of statistical models, and varying the amount by which a new relation must improve model performance in order to be retained. Another direction for further work is to permit the interleaving of steps that refine the definition of a learned relation and steps that vary the statistical model, or Bayesian network (in VISTA's case). A particular challenge within this direction is deciding what to do when varying the definition of a relation that itself appears (is used) in the definition of another relation.

Acknowledgments

Support for this research was partially provided by U.S. Air Force grant F30602-01-2-0571.

References

ACR (2004). Breast imaging reporting and data system (bi-rads).

Davis, J., Burnside, E., Dutra, I. C., Page, D., & Costa, V. S. (2005a). An integrated approach to learning bayesian networks of rules. *16th European Conference on Machine Learning* (pp. 84–95). Springer.

Davis, J., Burnside, E., Dutra, I. C., Page, D., Ramakrishnan, R., Costa, V. S., & Shavlik, J. (2005b). View learning for statistical relational learning: With an application to mammography. *Proceedings of the 19th International Joint Conference on Artificial Intelligence* (pp. 677–683). Edinburgh, Scotland.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian networks classifiers. *Machine Learning*, *29*, 131–163.

Kok, S., & Domingos, P. (2005). Learning the structure of markov logic networks. *Proceedings of the Twenty-Second International Conference on Machine Learning* (pp. 441–448). Bonn, Germany: ACM Press.

Landwehr, N., Kersting, K., & Raedt, L. D. (2005). nFOIL: Integrating Naive Bayes and FOIL. *National Conference on Artificial Intelligene (AAAI)*.

Lavrac, N., Dzeroski, S., & Grobelnik, M. (1991). Learning non-recursive definitions of relations with LINUS. In Y. Kodratoff (Ed.), *Proceedings of the fifth European working session on learning, lnai 482*, 265–281. Springer-Verlag.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, *3*, 127–163. www.research.whizbang.com/data.

Muggleton, S., & Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 339–352). Kaufmann.

Muggleton, S., & Raedt, L. D. (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 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 167–176). Washington, D.C.

Popescul, A., & Ungar, L. H. (2004). Cluster-based concept invention for statistical relational learning. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 665–670).

Popescul, A., Ungar, L. H., Lawrence, S., & Pennock, D. M. (2003). Statistical relational learning for document mining. *ICDM03* (pp. 275–282).

Rendell, L. (1985). Substantial constructive induction using layered information compression: tractable feature formation in search. *IJCAI-85* (pp. 650–658). Kaufmann.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *To appear in Machine Learning*.

Srinivasan, A. (2001). *The aleph manual*.

Vens, C., Assche, A. V., Blockeel, H., & Džeroski, S. (2004). First order random forests with complex aggregates. *Proceedings of the 14th International Conference on Inductive Logic Programming* (pp. 323–340).

Zelle, J., Mooney, R., & Konvisser, J. (1994). Combining top-down and bottom-up techniques in inductive logic programming. *Proceedings of the Eleventh International Workshop on Machine Learning* (pp. 343–351).