Collective Classification in Network Data

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor
University of Maryland, College Park

Brian Gallagher, Tina Eliassi-Rad
Lawrence Livermore National Laboratory

(Technical Report CS-TR-4905 and UMIACS-TR-2008-04)

Abstract

Numerous real-world applications produce networked data such as web data (hypertext documents connected via hyperlinks) and communication networks (people connected via communication links). A recent focus in machine learning research has been to extend traditional machine learning classification techniques to classify nodes in such data. In this report, we attempt to provide a brief introduction to this area of research and how it has progressed during the past decade. We introduce four of the most widely used inference algorithms for classifying networked data and empirically compare them on both synthetic and real-world data.

1 Introduction

Networks have become ubiquitous. Communication networks, financial transaction networks, networks describing physical systems, and social networks are all becoming increasingly important in our day-to-day life. Often, we are interested in models of how objects in the network influence each other (e.g., who infects whom in an epidemiological network), or we might want to predict an attribute of interest based on observed attributes of objects in the network (e.g., predicting political affiliations based on online purchases and interactions), or we might be interested in identifying important links in the network (e.g., in communication networks). In most of these scenarios, an important step in achieving our final goal, that either solves the problem completely or in part, is to classify the objects in the network.

Given a network and an object o in the network, there are three distinct types of correlations that can be utilized to determine the classification or label of o:

1. The correlations between the label of o and the observed attributes of o.
2. The correlations between the label of o and the observed attributes (including observed labels) of objects in the neighborhood of o.

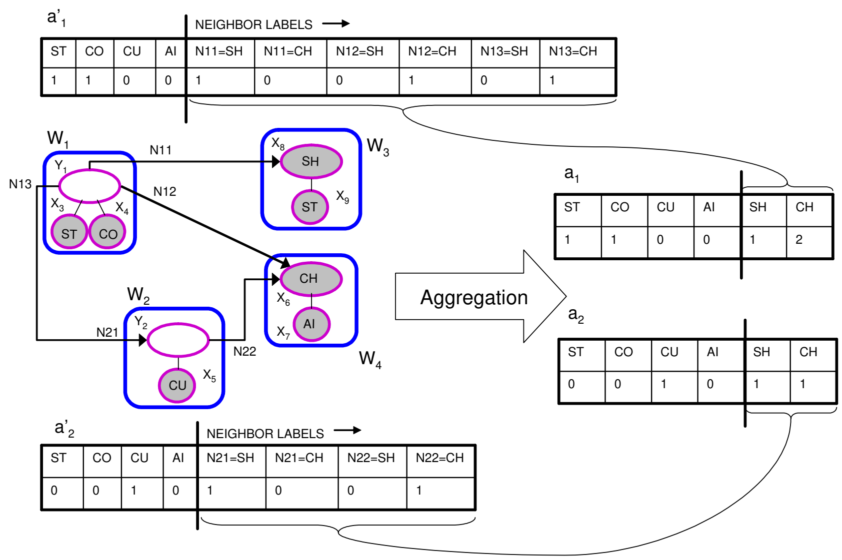


Figure 1: A small webpage classification problem. Each box denotes a webpage, each directed edge between a pair of boxes denotes a hyperlink, each oval node denotes a random variable, each shaded oval denotes an observed variable whereas an unshaded oval node denotes an unobserved variable whose value needs to be predicted. Assume that the set of label values is $\mathcal{L} = \{'SH', 'CH'\}$. The figure shows a snapshot during a run of ICA. Assume that during the last ICA labeling iteration we chose the following labels: $y_1 =' SH'$ and $y_2 =' CH'$. a'_1 and a'_2 show what may happen if we try to encode the respective values into vectors naively, i.e., we get variable-length vectors. The vectors a_1 and a_2 obtained after applying `count` aggregation shows one way of getting around this issue to obtain fixed-length vectors. See text for more explanation.

3. The correlations between the label of o and the unobserved labels of objects in the neighborhood of o.

Collective classification refers to the combined classification of a set of interlinked objects using all three types of information described above. Note that, sometimes the phrase *relational classification* is used to denote an approach that concentrates on classifying network data by using only the first two types of correlations listed above. However, in many applications that produce data with correlations between labels of interconnected objects (a phenomenon sometimes referred to as *relational autocorrelation* [41]) labels of the objects in the neighborhood are often unknown as well. In such cases, it becomes necessary to simultaneously infer the labels for all the objects in the network.

Within the machine learning community, classification is typically done on each object independently, without taking into account any underlying network that connects the objects. Collective classification does not fit well into this setting. For instance, in the webpage classification problem where webpages are interconnected with hyperlinks and the task is to assign each webpage with a label that best indicates its topic, it is common to assume that the labels on interconnected webpages are correlated. Such interconnections occur naturally in data from a variety of applications such as bibliographic data [10, 16], email networks [7] and social networks [41]. Traditional classification techniques would ignore the correlations represented by these interconnections and would be hard pressed to produce the classification accuracies possible using a collective classification approach.

Even though traditional exact inference algorithms such as *variable elimination* [68, 11] and the *junction tree algorithm* [22] harbor the potential to perform collective classification, they are practical only when the graph structure of the network satisfies certain conditions. In general, exact inference is known to be an NP-hard problem and there is no guarantee that real-world network data satisfy the conditions that make exact inference tractable for collective classification. As a consequence, most of the research in collective classification has been devoted to the development of *approximate inference algorithms*.

In this report we provide an introduction to four popular approximate inference algorithms used for collective classification, *iterative classification*, *Gibbs sampling*, *loopy belief propagation* and *mean-field relaxation labeling*. We provide an outline of the basic algorithms by providing pseudo-code, explain how one could apply them to real-world data, provide theoretical justifications (if there exist any), and discuss issues such as feature construction and various heuristics that may lead to improved classification accuracy. We provide case studies, on both real-world and synthetic data, to demonstrate the strengths and weaknesses of these approaches. All of these algorithms have a rich history of development and application to various problems relating to collective classification and we provide a brief discussion of this when we examine related work. Collective classification has been an active field of research for the past decade and as a result, there are numerous other approximate inference algorithms besides the four we describe here. We provide pointers to these works in the related work section. In the next section, we begin by introducing the required notation and define the collective classification problem formally.

2 Collective Classification: Notation and Problem Definition

Collective classification is a combinatorial optimization problem, in which we are given a set of nodes, $\mathcal{V} = \{V_1, \ldots, V_n\}$ and a neighborhood function $\mathcal{N}$, where $\mathcal{N}_i \subseteq \mathcal{V} \setminus \{V_i\}$, which describes the underlying network structure. Each node in $\mathcal{V}$ is a random variable that can take a value from an appropriate domain. $\mathcal{V}$ is further divided into two sets of nodes: $\mathcal{X}$, the nodes for which we know the correct values (observed variables) and, $\mathcal{Y}$, the nodes whose values need to be determined. Our task is to label the nodes $Y_i \in \mathcal{Y}$ with one of a small number of labels, $\mathcal{L} = \{L_1, \ldots, L_q\}$; we'll use the shorthand y_i to denote the label of node Y_i.

We explain the notation further using a webpage classification example that will serve as a running example throughout the report. Figure 1 shows a network of webpages with hyperlinks. In this example, we will use the words (and phrases) contained in the webpages as local attributes. For brevity, we abbreviate the local attributes, thus, 'ST' stands for "student", 'CO' stands for "course", 'CU' stands for "curriculum" and 'AI' stands for "Artificial Intelligence". Each webpage is indicated by a box, the corresponding topic of the webpage is indicated by an ellipse inside the box, and each word in the webpage is represented using a circle inside the box. The observed random variables $\mathcal{X}$ are shaded whereas the unobserved ones $\mathcal{Y}$ are not. We will assume that the domain of the unobserved label variables $\mathcal{L}$, in this case, is a set of two values: "student homepage" (abbreviated to 'SH') and "course homepage" (abbreviated to 'CH'). Figure 1 shows a network with two unobserved variables (Y_1 and Y_2), which require prediction, and seven observed variables (X_3, X_4, X_5, X_6, X_7, X_8 and X_9). Note that some of the observed variables happen to be labels of webpages (X_6 and X_8) for which we know the correct values. Thus, from the figure, it is easy to see that the webpage W_1, whose unobserved label variable is represented by Y_1, contains two words 'ST' and 'CO' and hyperlinks to webpages W_2, W_3 and W_4.

As mentioned in the introduction, due to the large body of work done in this area of research, we have a number of approaches for collective classification. At a broad level of abstraction, these approaches can be divided into two distinct types, one in which we use a collection of unnormalized local conditional classifiers and one where we define the collective classification problem as one global objective function to be optimized. We next describe these two approaches and, for each approach, we describe two approximate inference algorithms. For each topic of discussion, we will try to mention the relevant references so that the interested reader can follow up for a more in-depth view.

3 Approximate Inference Algorithms for Approaches based on Local Conditional Classifiers

Two of the most commonly used approximate inference algorithms following this approach are the *iterative classification algorithm* (ICA) and *gibbs sampling* (GS), and we next describe these in turn.

3.1 Iterative Classification

The basic premise behind ICA is extremely simple. Consider a node $Y_i \in \mathcal{Y}$ whose value we need to determine and suppose we know the values of all the other nodes in its neighborhood $\mathcal{N}_i$ (note that $\mathcal{N}_i$ can contain both observed and unobserved variables). Then, ICA assumes that we are given a local classifier f that takes the values of $\mathcal{N}_i$ as arguments and returns the best label value for Y_i from the class label set $\mathcal{L}$. For local classifiers f that do not return a class label but a goodness/likelihood value given a set of attribute values and a label, we simply choose the label that corresponds to the maximum goodness/likelihood value; in other words, we replace f with $\text{argmax}_{l \in \mathcal{L}} f$. This makes the local classifier f an extremely flexible function and we can use anything ranging from a decision tree to an SVM in its place. Unfortunately, it is rare in practice that we know all values in $\mathcal{N}_i$ which is why we need to repeat the process iteratively, in each iteration, labeling each Y_i using the current best estimates of $\mathcal{N}_i$ and the local classifier f, and continuing to do so until the assignments to the labels stabilize.

Algorithm 1 Iterative Classification Algorithm (ICA)

for each node $Y_i \in \mathcal{Y}$ **do** // bootstrapping
  // compute label using only observed nodes in $\mathcal{N}_i$
  compute $\vec{a}_i$ using only $\mathcal{X} \cap \mathcal{N}_i$
  $y_i \leftarrow f(\vec{a}_i)$
end for
repeat // iterative classification
  generate ordering $\mathcal{O}$ over nodes in $\mathcal{Y}$
  for each node $Y_i \in \mathcal{O}$ **do**
    // compute new estimate of y_i
    compute $\vec{a}_i$ using current assignments to $\mathcal{N}_i$
    $y_i \leftarrow f(\vec{a}_i)$
  end for
until all class labels have stabilized or a threshold number of iterations have elapsed

Most local classifiers are defined as functions whose argument consists of a fixed-length vector of attribute values. Going back to the example we introduced in the last section in Figure 1, assume that we are looking at a snapshot of the state of the labels after a few ICA iterations have elapsed and the label values assigned to Y_1 and Y_2 in the last iteration are ‘SH’ and ‘CH’, respectively. $\vec{a}'_1$ in Figure 1 denotes one attempt to pool all values of $\mathcal{N}_1$ into one vector. Here, the first entry in $\vec{a}'_1$ corresponds to the first neighbor of Y_1 is a ‘1’ denoting that Y_1 has a neighbor that is the word ‘ST’ (X_3), and so on. Unfortunately, this not only requires putting an ordering on the neighbors, but since Y_1 and Y_2 have a different number of neighbors, this type of encoding results in $\vec{a}'_1$ consisting of a different number of entries than $\vec{a}'_2$. Since the local classifier can take only vectors of a fixed length, this means we cannot use the same local classifier to classify both $\vec{a}'_1$ and $\vec{a}'_2$.

A common approach to circumvent such a situation is to use an aggregation operator such as `count` or `mode`. Figure 1 shows the two vectors $\vec{a}_1$ and $\vec{a}_2$ that are obtained by applying the `count` operator to $\vec{a}'_1$ and $\vec{a}'_2$, respectively. The `count` operator simply counts the number of neighbors assigned ‘SH’ and ‘CH’ and adds these entries to the vectors. Thus we get one new value for each entry in the set of label values. Assuming that the set of label values does not change from one unobserved node to another, this results in fixed-length vectors which can now be classified using the same local classifier. Thus, for instance, $\vec{a}_1$ contains two entries, besides the entries corresponding to the local word attributes, encoding that Y_1 has one neighbor labeled ‘SH’ (X_8) and two neighbors currently labeled ‘CH’ (Y_2 and X_6). Algorithm 1 depicts the ICA algorithm as pseudo-code where we use $\vec{a}_i$ to denote the vector encoding the values in $\mathcal{N}_i$ obtained after aggregation. Note that in the first ICA iteration, all labels y_i are undefined and to initialize them we simply apply the local classifier to the observed attributes in the neighborhood of Y_i, this is referred to as “bootstrapping” in

Algorithm 1.

3.2 Gibbs Sampling

Gibbs sampling (GS) [20] is widely regarded as one of the most accurate approximate inference procedures. It was originally proposed in Geman & Geman [15] in the context of image restoration. Unfortunately, it is also very slow and a common issue while implementing GS is to determine when the procedure has converged. Even though there are tests that can help one determine convergence, they are usually expensive or complicated to implement.

Due to the issues with traditional GS, researchers in collective classification [36, 39] use a simplified version where they assume, just like in the case of ICA, that we have access to a local classifier f that can be used to estimate the conditional probability distribution of the labels of Y_i given all the values for the nodes in $\mathcal{N}_i$. Note that, unlike traditional GS, there is no guarantee that this conditional probability distribution is the correct conditional distribution to be sampling from. At best, we can only assume that the conditional probability distribution given by the local classifier f is an approximation of the correct conditional probability distribution. Neville & Jensen [46] provide more discussion and justification for this line of thought in the context of relational dependency networks, where they use a similar form of GS for inference.

The pseudo-code for GS is shown in Algorithm 2. The basic idea is to sample for the best label estimate for Y_i given all the values for the nodes in $\mathcal{N}_i$ using local classifier f for a fixed number of iterations (a period known as "burn-in"). After that, not only do we sample for labels for each $Y_i \in \mathcal{Y}$ but we also maintain count statistics as to how many times we sampled label l for node Y_i. After collecting a predefined number of such samples, we output the best label assignment for node Y_i by choosing the label that was assigned the maximum number of times to Y_i while collecting samples. For all our experiments (that we report later) we set burn-in to 200 iterations and collected 1000 samples.

3.3 Feature Construction and Further Optimizations

One of the benefits of both ICA and GS is the fact that it is fairly simple to plug in any local classifier. Table 1 depicts the various local classifiers that have been used in the past. There is some evidence to indicate that some local classifiers tend to produce higher accuracies than others, at least in the application domains where such experiments have been conducted. For instance, Lu & Getoor [35] report that on bibliography datasets and webpage classification problems logistic regression outperforms naïve Bayes.

Recall that, to represent the values of $\mathcal{N}_i$, we described the use of an aggregation operator. In the example, we used `count` to aggregate values of the labels in the neighborhood but `count` is by no means the only aggregation operator available. Past research has used a variety of aggregation operators including `minimum`, `maximum`, `mode`, `exists` and `proportion`. Table 2 depicts various aggregation operators and the systems that used these operators. The choice of which aggregation operator to use depends on the application domain and relates to the larger question of *relational*

Algorithm 2 Gibbs sampling algorithm (GS)

for each node $Y_i \in \mathcal{Y}$ **do** // bootstrapping
  // compute label using only observed nodes in $\mathcal{N}_i$
  compute $\vec{a}_i$ using only $\mathcal{X} \cap \mathcal{N}_i$
  $y_i \leftarrow f(\vec{a}_i)$
end for
for n=1 to B **do** // burn-in
  generate ordering $\mathcal{O}$ over nodes in $\mathcal{Y}$
  for each node $Y_i \in \mathcal{O}$ **do**
    compute $\vec{a}_i$ using current assignments to $\mathcal{N}_i$
    $y_i \leftarrow f(\vec{a}_i)$
  end for
end for
for each node $Y_i \in \mathcal{Y}$ **do** // initialize sample counts
  for label $l \in \mathcal{L}$ **do**
    $c[i, l] = 0$
  end for
end for
for n=1 to S **do** // collect samples
  generate ordering $\mathcal{O}$ over nodes in $\mathcal{Y}$
  for each node $Y_i \in \mathcal{O}$ **do**
    compute $\vec{a}_i$ using current assignments to $\mathcal{N}_i$
    $y_i \leftarrow f(\vec{a}_i)$
    $c[i, y_i] \leftarrow c[i, y_i] + 1$
  end for
end for
for each node $Y_i \in \mathcal{Y}$ **do** // compute final labels
  $y_i \leftarrow \text{argmax}_{l \in \mathcal{L}} c[i, l]$
end for

feature construction where we are interested in determining which features to use so that classification accuracy is maximized. In particular, there is some evidence to indicate that new attribute values derived from the graph structure of the network in the data, such as the *betweenness centrality*, may be beneficial to the accuracy of the classification task [14]. Within the inductive logic programming community, aggregation has been studied as a means for propositionalizing a relational classification problem [28, 29, 30]. Within the statistical relational learning community, Perlich and Provost [48, 49] have studied aggregation extensively and Popescul and Ungar Popescul & Ungar [50] have worked on feature construction using techniques from inductive logic programming.

Other aspects of ICA that have been the subject of investigation include the ordering strategy to determine in which order to visit the nodes to relabel in each ICA iteration. There is some evidence to suggest that ICA is fairly robust to a number of simple ordering strategies such as random ordering, visiting nodes in ascending order of diversity of its neighborhood class labels and labeling nodes in descending order

Reference	local classifier used
Neville & Jensen [44]	naïve Bayes
Lu & Getoor [35]	logistic regression
Jensen, Neville, & Gallagher [25]	naïve Bayes, decision trees
Macskassy & Provost [36]	naïve Bayes, logistic regression, weighted-vote relational neighbor, class distribution relational neighbor
McDowell, Gupta, & Aha [39]	naïve Bayes, k-nearest neighbors

Table 1: Summary of local classifiers used by previous work in conjunction with ICA and GS.

of label confidences [18]. However, there is also some evidence that certain modifications to the basic ICA procedure tend to produce improved classification accuracies. For instance, both Neville & Jensen [44] and McDowell, Gupta, & Aha [39] propose a strategy where only a subset of the unobserved variables are utilized as inputs for feature construction. More specifically, in each iteration, they choose the top-k most confident predicted labels and use only those unobserved variables in the following iteration's predictions, thus ignoring the less confident predicted labels. In each subsequent iteration they increase the value of k so that in the last iteration all nodes are used for prediction. McDowell et al. report that such a "cautious" approach leads to improved accuracies.

4 Approximate Inference Algorithms for Approaches based on Global Formulations

An alternate approach to performing collective classification is to define a global objective function to optimize. In what follows, we will describe one common way of defining such an objective function and this will require some more notation.

We begin by defining a *pairwise Markov random field* (pairwise MRF) [55]. Let $G = (\mathcal{V}, E)$ denote a graph of random variables as before where $\mathcal{V}$ consists of two types of random variables, the unobserved variables, $\mathcal{Y}$, which need to be assigned values from label set $\mathcal{L}$, and observed variables $\mathcal{X}$ whose values we know. Let Ψ denote a set of *clique potentials*. Ψ contains three distinct types of functions:

- For each $Y_i \in \mathcal{Y}$, $\psi_i \in \Psi$ is a mapping $\psi_i : \mathcal{L} \rightarrow \Re_{\geq 0}$, where $\Re_{\geq 0}$ is the set of non-negative real numbers.

- For each $(Y_i, X_j) \in E$, $\psi_{ij} \in \Psi$ is a mapping $\psi_{ij} : \mathcal{L} \rightarrow \Re_{\geq 0}$.

Reference	aggr. operators
PRMs, Friedman *et al.* [12]	`mode`, `count`, SQL
RMNs, Taskar, Abbeel, & Koller [55]	`mode`, SQL
MLNs, Richardson & Domingos [51]	FOL
Lu & Getoor [35]	`mode`, `count`, `exists`
Macskassy & Provost [36]	`prop`
Gupta, Diwan, & Sarawagi [21]	`mode`, `count`
McDowell, Gupta, & Aha [39]	`prop`

Table 2: A list of systems and the aggregation operators they use to aggregate neighborhood class labels. The systems include probabilistic relational models (PRMs), relational Markov networks (RMNs) and Markov logic networks (MLNs). The aggregation operators include `Mode`, which is the most common class label, `prop` which is the proportion of each class in the neighborhood, `count`, which is the number of each class label, and `exists`, which is an indicator for each class label. SQL denotes the standard structured query language for databases and all aggregation operators it includes and, FOL stands for first order logic.

- For each $(Y_i, Y_j) \in E$, $\psi_{ij} \in \Psi$ is a mapping $\psi_{ij} : \mathcal{L} \times \mathcal{L} \to \Re_{\geq 0}$.

Let $\mathbf{x}$ denote the values assigned to all the observed variables in G and let x_i denote the value assigned to X_i. Similarly, let $\mathbf{y}$ denote any assignment to all the unobserved variables in G and let y_i denote a value assigned to Y_i. For brevity of notation we will denote by ϕ_i the clique potential obtained by computing $\phi_i(y_i) = \psi_i(y_i) \prod_{(Y_i, X_j) \in E} \psi_{ij}(y_i)$. We are now in a position to define a pairwise MRF.

Definition 1. *A pairwise Markov random field (MRF) is given by a pair $\langle G, \Psi \rangle$ where G is a graph and Ψ is a set of clique potentials with ϕ_i and ψ_{ij} as defined above. Given an assignment $\mathbf{y}$ to all the unobserved variables $\mathcal{Y}$, the pairwise MRF is associated with the probability distribution $P(\mathbf{y}|\mathbf{x}) = \frac{1}{\mathcal{Z}(\mathbf{x})} \prod_{Y_i \in \mathcal{Y}} \phi_i(y_i) \prod_{(Y_i, Y_j) \in E} \psi_{ij}(y_i, y_j)$ where $\mathbf{x}$ denotes the observed values of $\mathcal{X}$ and $\mathcal{Z}(\mathbf{x}) = \sum_{\mathbf{y}'} \prod_{Y_i \in \mathcal{Y}} \phi_i(y'_i) \prod_{(Y_i, Y_j) \in E} \psi_{ij}(y'_i, y'_j)$.*

The above notation is further explained in Figure 2 where we augment the running example introduced earlier by adding clique potentials. MRFs are defined on undirected graphs and thus we have dropped the directions on all the hyperlinks in the example. In Figure 2, ψ_1 and ψ_2 denote two clique potentials defined on the unobserved variables (Y_1 and Y_2) in the network. Similarly, we have one ψ defined for each edge that involves at least one unobserved variable as an end-point. For instance, ψ_{13} defines a mapping from $\mathcal{L}$ (which is set to $\{SH, CH\}$ in the example) to non-negative real numbers. There is only one edge between the two unobserved variables in the network and this edge is associated with the clique potential ψ_{12} that is a function over two arguments. Figure 2 also shows how to compute the ϕ clique potentials. Essentially, given an unobserved variable Y_i, one collects all the edges that connect it to observed

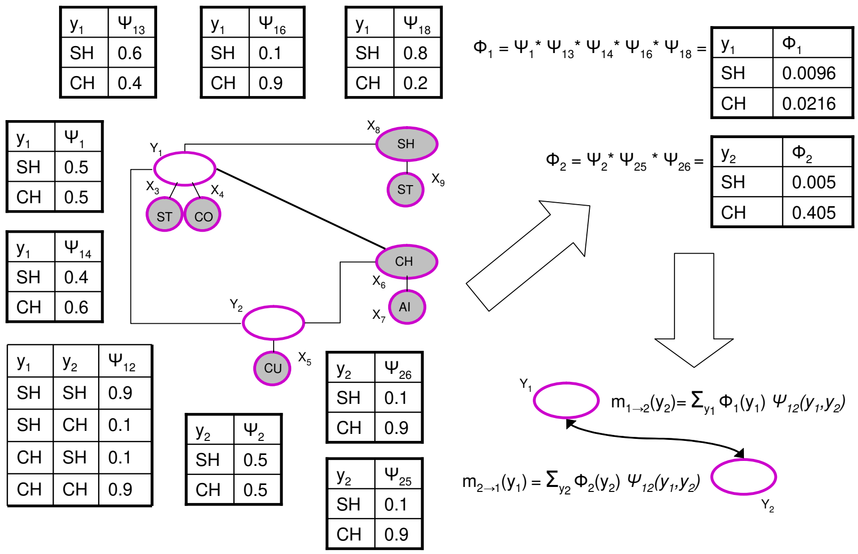


Figure 2: A small webpage classification problem expressed as a pairwise Markov random field with clique potentials. The figure also shows the message passing steps followed by LBP. See text for more explanation.

variables in the network and multiplies the corresponding clique potentials along with the clique potential defined on Y_i itself. Thus, as the figure shows, $\phi_2 = \psi_2 \times \psi_{25} \times \psi_{26}$.

Given a pairwise MRF, it is conceptually simple to extract the best assignments to each unobserved variable in the network. For instance, we may adopt the criterion that the best label value for Y_i is simply the one corresponding to the highest marginal probability obtained by summing over all other variables from the probability distribution associated with the pairwise MRF. Computationally however, this is difficult to achieve since computing one marginal probability requires summing over an exponentially large number of terms which is why we need approximate inference algorithms.

We describe two approximate inference algorithms in this report. Both of them adopt a similar approach to avoiding the computational complexity of computing marginal probability distributions. Instead of working with the probability distribution associated with the pairwise MRF directly (Definition 1) they both use a simpler "trial" distribution. The idea is to design the trial distribution so that once we fit it to the MRF distribution then it is easy to extract marginal probabilities from the trial distribution (as easy as reading off the trial distribution). This is a general principle which forms the basis of a class of approximate inference algorithms known as *variational methods* [26].

We are now in a position to discuss *loopy belief propagation* (LBP) and *mean-field relaxation labeling* (MF).

Algorithm 3 Loopy belief propagation (LBP)

for each $(Y_i, Y_j) \in E(G)$ s.t. $Y_i, Y_j \in \mathcal{Y}$ **do**
  for each $y_j \in \mathcal{L}$ **do**
    $m_{i \to j}(y_j) \leftarrow 1$
  end for
end for
repeat // perform message passing
  for each $(Y_i, Y_j) \in E(G)$ s.t. $Y_i, Y_j \in \mathcal{Y}$ **do**
    for each $y_j \in \mathcal{L}$ **do**
      $m_{i \to j}(y_j) \leftarrow \alpha \sum_{y_i} \psi_{ij}(y_i, y_j)\phi_i(y_i) \prod_{Y_k \in \mathcal{N}_i \cap \mathcal{Y} \setminus Y_j} m_{k \to i}(y_i)$
    end for
  end for
until all $m_{i \to j}(y_j)$ stop showing any change
for each $Y_i \in \mathcal{Y}$ **do** // compute beliefs
  for each $y_i \in \mathcal{L}$ **do**
    $b_i(y_i) \leftarrow \alpha\phi_i(y_i) \prod_{Y_j \in \mathcal{N}_i \cap \mathcal{Y}} m_{j \to i}(y_i)$
  end for
end for

4.1 Loopy Belief Propagation

Loopy belief propagation (LBP) applied to pairwise MRF $\langle G, \Psi \rangle$ is a message passing algorithm that can be concisely expressed as the following set of equations:

$$m_{i \to j}(y_j) = \alpha \sum_{y_i \in \mathcal{L}} \psi_{ij}(y_i, y_j)\phi_i(y_i) \prod_{Y_k \in \mathcal{N}_i \cap \mathcal{Y} \setminus Y_j} m_{k \to i}(y_i), \quad \forall y_j \in \mathcal{L} \tag{1}$$

$$b_i(y_i) = \alpha\phi_i(y_i) \prod_{Y_j \in \mathcal{N}_i \cap \mathcal{Y}} m_{j \to i}(y_i), \ \forall y_i \in \mathcal{L} \tag{2}$$

where $m_{i \to j}$ is a message sent by Y_i to Y_j and α denotes a normalization constant that ensures that each message and each set of marginal probabilities sum to 1, more precisely, $\sum_{y_j} m_{i \to j}(y_j) = 1$ and $\sum_{y_i} b_i(y_i) = 1$. The algorithm proceeds by making each $Y_i \in \mathcal{Y}$ communicate messages with its neighbors in $\mathcal{N}_i \cap \mathcal{Y}$ until the messages stabilize (Eq. (1)). After the messages stabilize, we can calculate the marginal probability of assigning Y_i with label y_i by computing $b_i(y_i)$ using Eq. (2). The algorithm is described more precisely in Algorithm 3. Figure 2 shows a sample round of message passing steps followed by LBP on the running example.

LBP has been shown to be an instance of a variational method. Let $b_i(y_i)$ denote the marginal probability associated with assigning unobserved variable Y_i the value y_i and let $b_{ij}(y_i, y_j)$ denote the marginal probability associated with labeling the edge (Y_i, Y_j) with values (y_i, y_j). Then Yedidia, Freeman, & Weiss [65] showed that the following choice of trial distribution,

$$b(\mathbf{y}) = \frac{\prod_{(Y_i,Y_j)\in E} b_{ij}(y_i, y_j)}{\prod_{Y_i\in\mathcal{Y}} b_i(y_i)^{|\mathcal{Y}\cap\mathcal{N}_i|-1}}$$

and subsequently minimizing the Kullback-Liebler divergence between the trial distribution from the distribution associated with a pairwise MRF gives us the LBP message passing algorithm with some qualifications. Note that the trial distribution explicitly contains marginal probabilities as variables. Thus, once we fit the distribution, extracting the marginal probabilities is as easy as reading them off.

4.2 Relaxation Labeling via Mean-Field Approach

Another approximate inference algorithm that can be applied to pairwise MRFs is *mean-field relaxation labeling* (MF). The basic algorithm can be described by the following fixed point equation:

$$b_j(y_j) = \alpha\phi_j(y_j) \prod_{Y_i\in\mathcal{N}_j\cap\mathcal{Y}} \prod_{y_i\in\mathcal{L}} \psi_{ij}^{b_i(y_i)}(y_i, y_j), \quad y_j \in \mathcal{L}$$

where $b_j(y_j)$ denotes the marginal probability of assigning $Y_j \in \mathcal{Y}$ with label y_j and α is a normalization constant that ensures $\sum_{y_j} b_j(y_j) = 1$. The algorithm simply computes the fixed point equation for every node Y_j and keeps doing so until the marginal probabilities $b_j(y_j)$ stabilize. When they do, we simply return $b_j(y_j)$ as the computed marginals. The pseudo-code for MF is shown in Algorithm 4.

Algorithm 4 Mean-field relaxation labeling (MF)

---

for each $Y_i \in \mathcal{Y}$ **do** // initialize messages
  for each $y_i \in \mathcal{L}$ **do**
    $b_i(y_i) \leftarrow 1$
  end for
end for
repeat // perform message passing
  for each $Y_j \in \mathcal{Y}$ **do**
    for each $y_j \in \mathcal{L}$ **do**
      $b_j(y_j) \leftarrow \alpha\phi_j(y_j) \prod_{Y_i\in\mathcal{N}_j\cap\mathcal{Y}, y_i\in\mathcal{L}} \psi_{ij}^{b_i(y_i)}(y_i, y_j)$
    end for
  end for
until all $b_j(y_j)$ stop changing

MF can also be justified as a variational method in almost exactly the same way as LBP. In this case, however, we choose a simpler trial distribution:

$$b(\mathbf{y}) = \prod_{Y_i\in\mathcal{Y}} b_i(y_i)$$

We refer the interested reader to Weiss [61], Yedidia, Freeman, & Weiss [65] for more details.

5 Experiments

In our experiments, we compared the four collective classification algorithms (CC) discussed in the previous sections and a content-only classifier (CO), which does not take the network into account, along with two choices of local classifiers on document classification tasks. The two local classifiers we tried were naïve Bayes (NB) and Logistic Regression (LR). This gave us 8 different classifiers: CO with NB, CO with LR, ICA with NB, ICA with LR, GS with NB, GS with LR, MF and LBP. The datasets we used for the experiments included both real-world and synthetic datasets.

5.1 Features used

For CO classifiers, we used the words in the documents for observed attributes. In particular, we used a binary value to indicate whether or not a word appears in the document. In ICA and GS, we used the same local attributes (i.e., words) followed by `count` aggregation to count the number of each label value in a node's neighborhood. Finally, for LBP and MF, we used pairwise MRF with clique potentials defined on the edges and unobserved nodes in the network.

5.2 Experimental Setup

Due to the fact that we are dealing with network data, traditional approaches to experimental data preparation such as creating splits for k-fold cross-validation may not be directly applicable. Splitting the dataset into k subsets randomly and using $k - 1$ of them for training leads to splits where we expect $\frac{k-1}{k}$ portion of the neighborhood of a test node are labeled. When k is reasonably high (say, 10) then this will result in almost the entire neighborhood of a test node being labeled. If we were to experiment with such a setup, we would not be able to compare how well the CC algorithms exploit the correlations between the unobserved labels of the connected nodes.

To construct splits whose neighbors are unlabeled as much as possible, we use a strategy that we refer to as snowball sampling evaluation strategy (SS). In this strategy, we construct splits for test data by randomly selecting the initial node and expanding around it. We do not expand randomly; instead, we select nodes based on the class distribution of the whole corpus; that is, the test data is stratified. The nodes selected by the SS are used as the test set while the rest are used for training. We repeat this process k times to obtain k test-train pairs of splits. Besides experimenting on test splits created using SS, we also experimented with splits created using the standard k-fold cross-validation methodology where we choose nodes randomly to create splits and refer to this as RS.

When using SS, some of the objects may appear in more than one test split. In that case, we need to adjust accuracy computation so that we do not over count those objects. A simple strategy is to average the accuracy for each instance first and then take the average of the averages. Further, to help the reader compare the SS results with the RS results, we also provide accuracies averaged per instance and across all instances that appear in at least one SS split. We denote these numbers using the term matched cross-validation (M).

5.3 Learning the classifiers

One aspect of the collective classification problem that we have not discussed so far is how to learn the various classifiers described in the previous sections. Learning refers to the problem of determining the parameter values for the local classifier, in the case of ICA and GS, and the entries in the clique potentials, in the case of LBP and MF, which can then be subsequently used to classify unseen test data. For all our experiments, we learned the parameter values from fully labeled datasets created through the splits generation methodology described above using gradient-based optimization approaches. For a more detailed discussion, see, for example,Taskar, Abbeel, & Koller [55], Sen & Getoor [52], and Sen & Getoor [53].

5.4 Real-world Datasets

We experimented with two real-world bibliographic datasets: Cora [38] and CiteSeer [19]. The *Cora* dataset contains a number of Machine Learning papers divided into one of 7 classes while the CiteSeer dataset has 6 class labels. For both datasets, we performed stemming and stop word removal besides removing the words with document frequency less than 10. The final corpus has 2708 documents, 1433 distinct words in the vocabulary and 5429 links, in the case of Cora, and 3312 documents, 3703 distinct words in the vocabulary and 4732 links in the case of CiteSeer. For each dataset, we performed both RS evaluation (with 10 splits) and SS evaluation (averaged over 10 runs).

5.4.1 Results

The accuracy results for the real world datasets are shown in Table 3. The accuracies are separated by sampling method and base classifier. The highest accuracy at each partition is in bold. We performed t-test (paired where applicable, and Welch t-test otherwise) to test statistical significance between results. Here are the main results:

1. Do CC algorithms improve over CO counterparts?

   In both datasets, CC algorithms outperformed their CO counterparts, in all evaluation strategies (SS, RS and M). The performance differences were significant for all comparisons except for the NB (M) results for Citeseer.

2. Does the choice of the base classifier affect the results of the CC algorithms?

   We observed a similar trend for the comparison between NB and LR. LR (and the CC algorithms that used LR as a base classifier) outperformed NB versions in all datasets, and the difference was statistically significant for Cora.

3. Is there any CC algorithm that dominates the other?

   The results for comparing CC algorithms are less clear. In the NB partition, ICA-NB outperformed GS-NB significantly for Cora using SS and M, and GS-NB outperformed ICA-NB for Citeseer SS. Thus, there was no clear winner between ICA-NB and GS-NB in terms of performance. In the LR portion, again

	Cora			Citeseer		
Algorithm	SS	RS	M	SS	RS	M
CO-NB	0.7285	0.7776	0.7476	0.7427	0.7487	0.7646
ICA-NB	**0.8054**	**0.8478**	**0.8271**	0.7540	**0.7683**	**0.7752**
GS-NB	0.7613	0.8404	0.8154	**0.7596**	0.7680	0.7737
CO-LR	0.7356	0.7695	0.7393	0.7334	0.7321	0.7532
ICA-LR	0.8457	0.8796	0.8589	**0.7629**	**0.7732**	0.7812
GS-LR	**0.8495**	**0.8810**	**0.8617**	0.7574	0.7699	**0.7843**
LBP	0.8554	0.8766	0.8575	**0.7663**	**0.7759**	0.7843
MF	**0.8555**	**0.8836**	**0.8631**	0.7657	0.7732	**0.7888**

Table 3: Accuracy results for the Cora and Citeseer datasets. For Cora, the CC algorithms outperformed their CO counterparts significantly. LR versions significantly outperformed NB versions. ICA-NB outperformed GS-NB for SS and M, the other differences between ICA and GS were not significant (both NB and LR versions). Even though MF outperformed ICA-LR, GS-LR, and LBP, the differences were not statistically significant. For Citeseer, the CC algorithms significantly outperformed their CO counterparts except for ICA-NB and GS-NB for matched cross-validation. CO and CC algorithms based on LR outperformed the NB versions, but the differences were not significant. ICA-NB outperformed GS-NB significantly for SS; but, the rest of the differences between LR versions of ICA and GS, LBP, and MF were not significant.

the differences between ICA-LR and GS-LR were not significant for all datasets. As for LBP and MF, they often slightly outperformed ICA-LR and GS-LR, but the differences were not significant.

4. How do SS results and RS results compare?

   Finally, we take a look at the numbers under the columns labeled M. First, we would like to remind the reader that even though we are comparing the results on the test set that is the intersection of the two evaluation strategies (SS and RS), different training data could have been potentially used for each test instance, thus the comparison can be questioned. Nonetheless, we expected the matched cross-validation results (M) to outperform SS results simply because each instance had more labeled data around it from RS splitting. The differences were not big (around 1% or 2%); however, they were significant. These results tell us that the evaluation strategies can have a big impact on the final results, and care must be taken while designing an experimental setup for evaluating CC algorithms on network data [13].

5.5 Synthetic Data

We implemented a synthetic data generator following Sen and Getoor [53]. High-level pseudo-code is given in Algorithm 5 for completeness, but more details can be found in Sen & Getoor.

Algorithm 5 Synthetic data generator

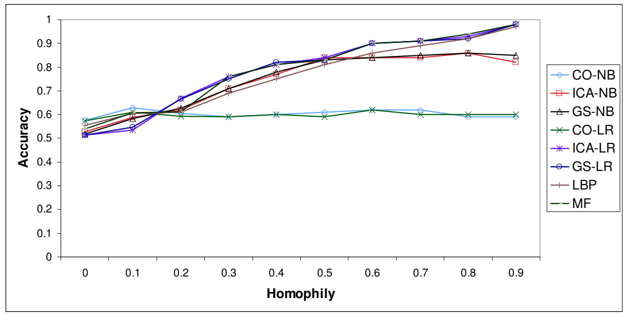
$i = 1$
while $i = 1 <$ numNodes **do**
  Sample r uniformly random from $[0, 1)$.
  if $r < ld$ **then**
    Pick a source node $node_s$
    Pick a destination node $node_d$ based on dh and degree
    Add a link between $node_s$ and $node_d$
  else
    Generate a node $node_i$
    Sample a class for $node_i$
    Sample attributes for $node_i$ using a binomial distribution. Introduce noise to the process based on the attribute noise parameter.
    Connect it to a destination node based on dh and degree
    $i \leftarrow i + 1$
  end if
end while

At each step, we either add a link between two existing nodes or create a node based on the ld parameter (such that higher ld value means higher link density, i.e., more links in the graph) and link this new node to an existing node. When we are adding a link, we choose the source node randomly but we choose the destination node using the dh parameter (which varies homophily [41] by specifying what percentage, on average, of a node's neighbor is of the same type) as well as the degree of the candidates (preferential attachment [3]). When we are generating a node, we sample a class for it and generate attributes based on this class using a binomial distribution. Then, we add a link between the new node and one of the existing nodes, again using the homophily parameter and the degree of the existing nodes. In all of these experiments, we generated 1000 nodes, where each node is labeled with one of 5 possible class labels and has 10 attributes. We experimented with varying degree of homophily and link density in the graphs generated and we used SS strategy to generate the train-test splits.

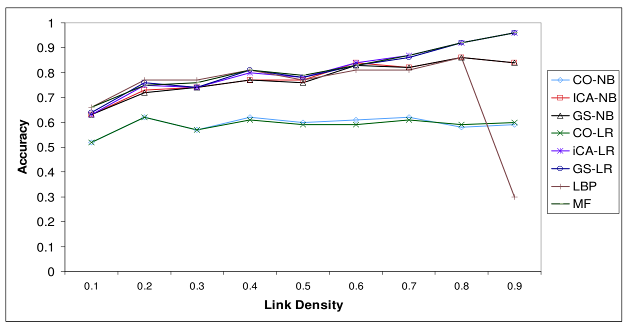
5.5.1 Results

The results for varying values of dh, homophily, are shown in Figure 3(a). When homophily in the graph was low, both CO and CC algorithms performed equally well, which was expected result based on similar work. As we increased the amount of homophily, all CC algorithms drastically improved their performance over CO classifiers. With homophily at dh=.9, for example, the difference between our best performing CO algorithm and our best performing CC algorithm is about 40%. Thus, for datasets which demonstrate some level of homophily, using CC can significantly improve performance.

We present the results for varying the ld, link density, parameter in Figure 3(b) . As we increased the link density of the graphs, we saw that accuracies for all algorithms



(a)



(b)

Figure 3: (a) Accuracy of algorithms through different values for dh varying the levels of homophily. When the homophily is very low, both CO and CC algorithms perform equally well but as we increase homophily, CC algorithms improve over CO classifier. (b) Accuracy of algorithms through different values for ld varying the levels of link density. As we increase link density, ICA and GS improve their performance the most. Next comes MF. However, LBP has convergence issues due to increased cycles and in fact performs worse than CO for high link density.

went up, possibly because the relational information became more significant and useful. However, the LBP accuracy had a sudden drop when the graph was immensely dense. The reason behind this result is the well known fact that LBP has convergence issues when there are many closed loops in the graph [45].

5.6 Practical Issues

In this section, we discuss some of the practical issues to consider when applying the various CC algorithms. First, although MF and LBP performance is in some cases a bit better than ICA and GS, they were also the most difficult to work with in both learning and inference. Choosing the initial weights so that the weights will converge during training is non-trivial. Most of the time, we had to initialize the weights with the weights we got from ICA in order to get the algorithms to converge. Thus, the MF and LBP had some extra advantage in the above experiments. We also note that of the two, we had the most trouble with MF being unable to converge, or when it did, not converging to the global optimum. Our difficulty with MF and LBP are consistent with previous work [61, 43, 64] and should be taken into consideration when choosing to apply these algorithms.

In contrast, ICA and GS parameter initializations worked for all datasets we used and we did not have to tune the initializations for these two algorithms. They were the easiest to train and test among all the collective classification algorithms evaluated.

Finally, while ICA and GS produced very similar results for almost all experiments, ICA is a much faster algorithm than GS. In our largest dataset, Citeseer, for example, ICA-NB took 14 minutes to run while GS-NB took over 3 hours. The large difference is due to the fact that ICA converges in just a few iterations, whereas GS has to go through significantly more iterations per run due to the initial burn-in stage (200 iterations), as well as the need to run a large number of iterations to get a sufficiently large sampling (800 iterations).

6 Related Work

Even though collective classification has gained attention only in the past five to seven years, initiated by the work of Jennifer Neville and David Jensen [44, 24, 25, 46] and the work of Ben Taskar et al. [59, 17, 55, 58], the general problem of inference for structured output spaces has received attention for a considerably longer period of time from various research communities including computer vision, spatial statistics and natural language processing. In this section, we attempt to describe some of the work that is most closely related to the work described in this report, however, due to the widespread interest in collective classification our list is sure to be incomplete.

One of the earliest principled approximate inference algorithms, *relaxation labeling* [23], was developed by researchers in computer vision in the context of object labeling in images. Due to its simplicity and appeal, relaxation labeling was a topic of active research for some time and many researchers developed different versions of the basic algorithm [34]. Mean-field relaxation labeling [61, 65], discussed in this report, is a simple instance of this general class of algorithms. Besag [5] also considered statistical analysis of images and proposed a particularly simple approximate inference algorithm called *iterated conditional modes* which is one of the earliest descriptions and a specific version of the *iterative classification algorithm* presented in this report. Besides computer vision, researchers working with an iterative decoding scheme known as "Turbo Codes" [4] came up with the idea of applying Pearl's belief propagation algorithm [47] on networks with loops. This led to the development of the approximate inference algorithm that we, in this report, refer to as *loopy belief propagation* (LBP) (also known as *sum product* algorithm) [31, 40, 32].

Another area that often uses collective classification techniques is document classification. Chakrabarti, Dom, & Indyk [8] was one of the first to apply collective classification to a corpora of patents linked via hyperlinks and reported that considering attributes of neighboring documents actually hurts classification performance. Slattery

& Craven [54] also considered the problem of document classification by constructing features from neighboring documents using an *inductive logic programming* rule learner. Yang, Slattery, & Ghani [63] conducted an in-depth investigation over multiple datasets commonly used for document classification experiments and identified different patterns. Since then, collective classification has also been applied to various other applications such as part-of-speech tagging [33], classification of hypertext documents using hyperlinks [55], link prediction in friend-of-a-friend networks [56], optical character recognition [58], entity resolution in sensor networks [9], predicting disulphide bonds in protein molecules [57], segmentation of 3D scan data [2] and classification of email "speech acts" [7].

Besides the four approximate inference algorithms discussed in this report, there are other algorithms that we did not discuss such as graph-cuts based formulations [6], formulations based on linear programming relaxations [27, 60] and expectation propagation [42]. Other examples of approximate inference algorithms include algorithms developed to extend and improve loopy belief propagation to remove some of its shortcomings such as alternatives with convergence guarantees [67] and alternatives that go beyond just using edge and node marginals to compute more accurate marginal probability estimates such as the cluster variational method [66], junction graph method [1] and region graph method [65].

More recently, there have been some attempts to extend collective classification techniques to the semi-supervised learning scenario [62, 37].

7 Conclusion

In this report, we gave a brief description of four popular collective classification algorithms. We explained the algorithms, showed how to apply them to various applications using examples and highlighted various issues that have been the subject of investigation in the past. Most of the inference algorithms available for practical tasks relating to collective classification are approximate. We believe that a better understanding of when these algorithms perform well will lead to more widespread application of these algorithms to more real-world tasks and that this should be a subject of future research. Most of the current applications of these algorithms have been on homogeneous networks with a single type of unobserved variable that share a common domain. Even though extending these ideas to heterogeneous networks is conceptually simple, we believe that a further investigation into techniques that do so will lead to novel approaches to feature construction and a deeper understanding of how to improve the classification accuracies of approximate inference algorithms. Collective classification has been a topic of active research for the past decade and we hope that more reports such as this one will help more researchers gain introduction to this area thus promoting further research into the understanding of existing approximate inference algorithms and perhaps help develop new, improved inference algorithms.

8 Acknowledgements

We would like to thank Luke McDowell for his useful and detailed comments. This material is based upon work supported in part by the National Science Foundation under Grant No.0308030. In addition, this work was partially performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

[1] Aji, S. M., and McEliece, R. J. 2001. The generalized distributive law and free energy minimization. In *Proceedings of the 39th Allerton Conference on Communication, Control and Computing*.

[2] Anguelov, D.; Taskar, B.; Chatalbashev, V.; Koller, D.; Gupta, D.; Heitz, G.; and Ng, A. 2005. Discriminative learning of markov random fields for segmentation of 3d scan data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[3] Barabasi, A.-L., and Albert, R. 2002. Statistical mechanics of complex networks. *Reviews of Modern Physics* 74:47–97.

[4] Berrou, C.; Glavieux, A.; and Thitimajshima, P. 1993. Near Shannon limit error-correcting coding and decoding: Turbo codes. In *Proceedings of IEEE International Communications Conference*.

[5] Besag, J. 1986. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*.

[6] Boykov, Y.; Veksler, O.; and Zabih, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[7] Carvalho, V., and Cohen, W. W. 2005. On the collective classification of email speech acts. In *Special Interest Group on Information Retrieval.*

[8] Chakrabarti, S.; Dom, B.; and Indyk, P. 1998. Enhanced hypertext categorization using hyperlinks. In *International Conference on Management of Data.*

[9] Chen, L.; Wainwright, M.; Cetin, M.; and Willsky, A. 2003. Multitarget-multisensor data association using the tree-reweighted max-product algorithm. In *SPIE Aerosense conference*.

[10] Cohn, D., and Hofmann, T. 2001. The missing link—a probabilistic model of document content and hypertext connectivity. In *Neural Information Processing Systems.*

[11] Dechter, R. 1996. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence.*

[12] Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence.*

[13] Gallagher, B., and Eliassi-Rad, T. 2007a. An evaluation of experimental methodology for classifiers of relational data. In *Workshop on Mining Graphs and Complex Structures, IEEE International Conference on Data Mining (ICDM).*

[14] Gallagher, B., and Eliassi-Rad, T. 2007b. Leveraging network structure to infer missing values in relational data. Technical Report UCRL-TR-231993, Lawrence Livermore National Laboratory.

[15] Geman, S., and Geman, D. 1984. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[16] Getoor, L.; Friedman, N.; Koller, D.; and Taskar, B. 2001a. Probabilistic models of relational structure. In *Proc. ICML01*.

[17] Getoor, L.; Segal, E.; Taskar, B.; and Koller, D. 2001b. Probabilistic models of text and link structure for hypertext classification. In *IJCAI Workshop on Text Learning: Beyond Supervision*.

[18] Getoor, L. 2005. *Advanced Methods for Knowledge Discovery from Complex Data*. Springer. chapter Link-based classification.

[19] Giles, C. L.; Bollacker, K.; and Lawrence, S. 1998. Citeseer: An automatic citation indexing system. In *ACM Digital Libraries*.

[20] Gilks, W. R.; Richardson, S.; and Spiegelhalter, D. J. 1996. *Markov Chain Monte Carlo in Practice*. Interdisciplinary Statistics. Chapman & Hall/CRC.

[21] Gupta, R.; Diwan, A. A.; and Sarawagi, S. 2007. Efficient inference with cardinality-based clique potentials. In *Proceedings of the International Conference on Machine Learning.*

[22] Huang, C., and Darwiche, A. 1994. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*.

[23] Hummel, R., and Zucker, S. 1983. On the foundations of relaxation labeling processes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[24] Jensen, D., and Neville, J. 2002. Linkage and autocorrelation cause feature selection bias in relational learning. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*.

[25] Jensen, D.; Neville, J.; and Gallagher, B. 2004. Why collective inference improves relational classification. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[26] Jordan, M. I.; Ghahramani, Z.; Jaakkola, T. S.; and Saul, L. K. 1999. An introduction to variational methods for graphical models. *Machine Learning*.

[27] Kleinberg, J., and Tardos, E. 1999. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *IEEE Symposium on Foundations of Computer Science.*

[28] Knobbe, A.; deHaas, M.; and Siebes, A. 2001. Propositionalisation and aggregates. In *Proceedings of the Fifth European Conference on Principles of Data Mining and Knowledge Discovery*.

[29] Kramer, S.; Lavrac, N.; and Flach, P. 2001. Propositionalization approaches to relational data mining. In Dzeroski, S., and Lavrac, N., eds., *Relational Data Mining*. New York: Springer-Verlag.

[30] Krogel, M.; Rawles, S.; Zeezny, F.; Flach, P.; Lavrac, N.; and Wrobel, S. 2003. Comparative evaluation of approaches to propositionalization. In *International Conference on Inductive Logic Programming.*

[31] Kschischang, F. R., and Frey, B. J. 1998. Iterative decoding of compound codes by probability progation in graphical models. *IEEE Journal on Selected Areas in Communication*.

[32] Kschischang, F. R.; Frey, B. J.; and Loeliger, H. A. 2001. Factor graphs and the sum-product algorithm. In *IEEE Transactions on Information Theory*.

[33] Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning.*

[34] Li, S.; Wang, H.; and Petrou, M. 1994. Relaxation labeling of markov random fields. In *In Proceedings of International Conference Pattern Recognition*, volume 94.

[35] Lu, Q., and Getoor, L. 2003. Link based classification. In *Proceedings of the International Conference on Machine Learning.*

[36] Macskassy, S., and Provost, F. 2007. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research.*

[37] Macskassy, S. A. 2007. Improving learning in networked data by combining explicit and mined links. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*.

[38] McCallum, A.; Nigam, K.; Rennie, J.; and Seymore, K. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*.

[39] McDowell, L. K.; Gupta, K. M.; and Aha, D. W. 2007. Cautious inference in collective classification. In *Proceedings of AAAI.*

[40] McEliece, R. J.; MacKay, D. J. C.; and Cheng, J. F. 1998. Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE Journal on Selected Areas in Communication*.

[41] McPherson, M.; Smith-Lovin, L.; and Cook, J. M. 2001. Birds of a feather: Homophily in social networks. *Annual Review of Sociology* 27. This article consists of 30 page(s).

[42] Minka, T. 2001. Expectation propagation for approximate bayesian inference. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence.*

[43] Mooij, J. M., and Kappen, H. J. 2004. Validity estimates for loopy belief propagation on binary real-world networks. In *NIPS*.

[44] Neville, J., and Jensen, D. 2000. Iterative classification in relational data. In *Workshop on Statistical Relational Learning, AAAI*.

[45] Neville, J., and Jensen, D. 2007a. Bias/variance analysis for relational domains. In *International Conference on Inductive Logic Programming*.

[46] Neville, J., and Jensen, D. 2007b. Relational dependency networks. *Journal of Machine Learning Research.*

[47] Pearl, J. 1988. Probabilistic reasoning in intelligent systems. In *Morgan Kaufmann, San Fansisco*.

[48] Perlich, C., and Provost, F. 2003. Aggregation-based feature invention and relational concept classes. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

[49] Perlich, C., and Provost, F. 2006. Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning Journal*.

[50] Popescul, A., and Ungar, L. 2003. Structural logistic regression for link analysis. In *KDD Workshop on Multi-Relational Data Mining*.

[51] Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning*.

[52] Sen, P., and Getoor, L. 2006. Empirical comparison of approximate inference algorithms for networked data. In *ICML workshop on Open Problems in Statistical Relational Learning (SRL2006).*

[53] Sen, P., and Getoor, L. 2007. Link-based classification. Technical Report CS-TR-4858, University of Maryland.

[54] Slattery, S., and Craven, M. 1998. Combining statistical and relational methods for learning in hypertext domains. In *International Conference on Inductive Logic Programming.*

[55] Taskar, B.; Abbeel, P.; and Koller, D. 2002. Discriminative probabilistic models for relational data. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence.*

[56] Taskar, B.; Wong, M. F.; Abbeel, P.; and Koller, D. 2003. Link prediction in relational data. In *Neural Information Processing Systems.*

[57] Taskar, B.; Chatalbashev, V.; Koller, D.; and Guestrin, C. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the International Conference on Machine Learning.*

[58] Taskar, B.; Guestrin, C.; and Koller, D. 2003. Max-margin markov networks. In *Neural Information Processing Systems.*

[59] Taskar, B.; Segal, E.; and Koller, D. 2001. Probabilistic classification and clustering in relational data. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

[60] Wainwright, M. J.; Jaakkola, T. S.; and Willsky, A. S. 2005. Map estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. In *IEEE Transactions on Information Theory*.

[61] Weiss, Y. 2001. *Advanced Mean Field Methods, Manfred Opper and David Saad (eds)*. MIT Press. chapter Comparing the mean field method and belief propagation for approximate inference in MRFs.

[62] Xu, L.; Wilkinson, D.; Southey, F.; and Schuurmans, D. 2006. Discriminative unsupervised learning of structured predictors. In *Proceedings of the International Conference on Machine Learning.*

[63] Yang, Y.; Slattery, S.; and Ghani, R. 2002. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*.

[64] Yanover, C., and Weiss, Y. 2002. Approximate inference and protein-folding. In *Neural Information Processing Systems.*

[65] Yedidia, J.; Freeman, W.; and Weiss, Y. 2005. Constructing free-energy approximations and generalized belief propagation algorithms. In *IEEE Transactions on Information Theory*.

[66] Yedidia, J.; W.T.Freeman; and Weiss, Y. 2000. Generalized belief propagation. In *Neural Information Processing Systems.*

[67] Yuille, A. L. 2002. CCCP algorithms to minimize the bethe and kikuchi free energies: Convergent alternatives to belief propagation. In *Neural Information Processing Systems.*

[68] Zhang, N. L., and Poole, D. 1994. A simple approach to bayesian network computations. In *Canadian Conference on Artificial Intelligence*.