Link-based Classification

Prithviraj Sen *
sen@cs.umd.edu

Lise Getoor *
getoor@cs.umd.edu

Abstract

Over the past few years, a number of approximate inference algorithms for networked data have been put forth. We empirically compare the performance of three of the popular algorithms: loopy belief propagation, mean field relaxation labeling and iterative classification. We rate each algorithm in terms of its robustness to noise, both in attribute values and correlations across links. We also compare them across varying types of correlations across links.

1 Introduction

Traditional machine learning classification algorithms attempt to classify data organized as a collection of independent and identically distributed samples. Most real world data, on the other hand, is *relational* where different samples are *related* to each other. Classification in the presence of such relationships requires that we exploit correlations present in them. *Link-based classification* is the task of classifying samples using the *relations* or *links* present amongst them.

In *link-based classification*, more often that not, the hypothesis is that the *classifications of linked entities are correlated*. This creates a problem since when classifying entity 1 we need the classification of entity 2 and when classifying entity 2 we need the classification of entity 1, where entities 1 and 2 are two linked entities. Such interactions render most traditional classification algorithms ineffective. A prime objective of *link-based classification* has been the development of novel inference methods or *collective classification algorithms* that allow us to classify multiple entities in a joint fashion.

In many cases, link-based classification needs to be applied to data where the link structure harbours cycles. For example, it is commonly believed that webpages hyperlink to other webpages with similiar topics. Quite often webpage A hyperlinks to webpage B which hyperlinks to webpage C and C hyperlinks back to webpage A producing a cycle in the link structure. If we are to use the hyperlink structure to classify webpages with their topic labels we need to deal with cycles of this kind. Moreover, such link-based classification problems usually needs to be applied to data which consists of tens of thousands of interconnected samples forming an irregular graph. With graphs of such size, consisting of cycles and no apparent structure, it becomes infeasible to apply exact inference and one usually resorts to *approximate inference*. Even though

*Department of Computer Science, University of Maryland, College Park, MD 20742.

approximate inference does not promise exactness, it helps keep the computation and time requirements tractable. A number of *approximate collective classification algorithms* (ACCA) have been proposed. In this report, we aim to study and compare these inference algorithms on the task of classification in irregular graphs.

Studies comparing CCAs have been performed before but in different contexts. Murphy et al. [15] studies the performance of *loopy belief propagation* on various networks with loops. Their aim was to find out if *loopy belief propagation* can indeed approximate the posterior marginal probabilities in graphs with loops. Murphy et al. does not compare *loopy belief propagation* to other approximate CCA. Similarly, Jensen et al. [11] attempts to compare *various classifiers* for relational data where each classifier has a different set of features. Our aim is to compare the CCAs in a fair setting where each CCA is used to learn a classifier with the same number of parameters and features.

In this report, we aim to empirically study three of the most popular ACCAs available, viz. *loopy belief propagation* (LBP), *mean field relaxation labeling* (MF) and *iterative classification algorithm* (ICA). All three ACCAs can be viewed as message passing algorithms that proceed in rounds where each round consists of a set of messages being passed around. In ICA, each message is a class label and in each round of message passing each node asks for its neighbours current best guess of classification. After attaining the class labels in its neighbourhood the node refines its guess for its own class label. Mean field also proceeds in rounds of message passing where each node asks its neighbours for their *probability distribution* over class labels instead of just a class label. Having obtained the probability distribution of each and every neighbour the node recomputes the probability distribution over its own set of labels. LBP also proceeds in rounds of message passing like the previous two but is more sophisticated compared to ICA and MF on two counts. First, in each round of message passing each node asks its neighbour for a probability distribution over the set of class labels but this distribution consists of *the beliefs for the node itself*. More specifically, in a particular round of messages, node i would ask its neighbour node j for the set of probabilities of node i being labeled with label c for each c in the set of labels. Further, the neighbour would compute the probability distribution without taking into account the message it received from the node its going to send its messages to. Neither ICA nor MF makes this distinction and all computations are done based on *all* messages received till that point. It remains to be seen whether such sophistications in the case of LBP (that add considerable complexity both in time and space) allow it to perform better than its competitors.

We make minimal assumptions about the background of the reader and describe each ACCA before describing our experiments. Note that even though the developers of these algorithms suggest using them for inference in irregular graphs, usually the experiments they report are on regular graphs like 2-D grids ([29]). Our aim is to compare the performance of the ACCAs on the task of link-based classification in irregular graphs possibly consisting of cycles.

All three algorithms we compare here began as heuristics. Two of these have subsequently been justified by theory (MF and LBP) and there is some understanding as to what they are trying to do. We would like to investigate if this justification in theory translates to better performance in practice. Moreover, none of the algorithms we discuss in this report have a proof of convergence, in

reality they do not converge on many occassions. One of the goals of this report is to identify the accuracy and convergence properties of the various algorithms with changing graph characteristics.
Contributions of this report:

- A description of the *link-based classification* problem with various examples explaining its different applications.
- A description of three of the most popular and simple CCAs available.
- Experiments on synthethic datasets consisting of irregular graphs comparing the various ACCAs' accuracies on the task of classification and their convergence properties. We compare the effects of changing various graph characteristics (like link density) on the different ACCAs.
- Experiments comparing the performance of various ACCAs on synthetic datasets with different link patterns.
- Experiments on classification with real-world datasets composed of irregular graphs.
- Identification of the strengths and weaknesses of the various ACCAs compared and directions for future work.

In the next section we begin by introducing some notation and discussing the basic link-based classification problem. In Section 3 we introduce two of the common forms of input used to invoke various CCA. In Section 4, we discuss, in depth, four of the commonly used ACCAs. In Section 5, we discuss the classifiers and learning algorithms we used to test the ACCAs we discussed. In Section 6, we provide our experimental results. Section 7 provides a brief summary of the related work in link-based classification.

2 Problem Formulation

In the link-based classification problem, we have a data graph $G^d = (\mathbf{O}^d, \mathbf{R}^d)$ composed of a set objects $\mathbf{O}^d$ connected to each other via a set of relationships $\mathbf{R}^d$. The task is to label the members of $\mathbf{O}^d$ from a finite set of categorical values.

To define a probabilistic model for classifying such *relational* data G^d we will define a graph $G = (\mathbf{V}, \mathbf{E})$ composed of a set of random variables $\mathbf{V}$ and a set of links $\mathbf{E}$. Each random variable $v \in \mathbf{V}$ corresponds to an object $o^d \in \mathbf{O}^d$ and v can be assigned values from the set of labels that o^d can be labeled with. Let $\mathbf{v}$ denote an assignment to $\mathbf{V}$. The set of links $\mathbf{E}$ which interconnect $\mathbf{V}$ is derived from the set of relationships $\mathbf{R}^d$ amongst $\mathbf{O}^d$ and we will assume that we know the mapping to derive the set of links $\mathbf{E}$ from $\mathbf{R}^d$, $\mathbf{O}^d$ and $\mathbf{V}$. One common approach is to take the set of relationships $\mathbf{R}^d$ to be the set of links $\mathbf{E}$, in other words, we define a link $e \in \mathbf{E}$ between random variables $v_i, v_j \in \mathbf{V}$ for every relationship $r^d \in \mathbf{R}^d$ which exists between the corresponding objects $o_i^d, o_j^d \in \mathbf{O}^d$. For each edge $e \in \mathbf{E}$ we denote its head by $e.head$ and its tail by $e.tail$.

For the purposes of classification we are usually interested in conditional probability models. Let $\mathbf{X}$ be the set of observed random variables or "evidence"

random variables whose assignments can be obtained by simply observing the data G^d. Let $\mathbf{x}$ denote an assignment to $\mathbf{X}$. Let $\mathbf{Y}$ be the set of target random variables whose assignments are not present in the data G^d and need to be determined. Let $\mathbf{y}$ denote an assignment to $\mathbf{Y}$. Note that $\mathbf{X}$ and $\mathbf{Y}$ are such that $\mathbf{X} \cup \mathbf{Y} = \mathbf{V}$ and $\mathbf{X} \cap \mathbf{Y} = \emptyset$.

Let $\mathbf{L}$ denote the set of categorical values we can label $\mathbf{V}$ with. Each $v \in \mathbf{V}$ is typed and $v.T$ denotes the type of v. We are given the possible types $\mathcal{T}_v$. Each type $v.T$ is observed and is derived from the corresponding object in $\mathbf{O}^d$. Each $l \in \mathbf{L}$ is also typed with $l.T \in \mathcal{T}_v$. Any $y \in \mathbf{Y}$ can be labeled with $l \in \mathbf{L}$ if and only if $l.T = y.T$. Further, each $e \in \mathbf{E}$ is typed and $e.T$ denotes the type of e. We are given the possible types $\mathcal{T}_e$ and each type $e.T$ is observed. Note that $\mathcal{T}_e$ can be different from $\mathcal{T}_v$.

In this report, we consider the problem of supervised classification which can be formulated in the following two distinct ways:

- *Training with fully labeled dataset*: We are given two disjoint instances of relational data from which we derive two graphs composed of random variables interconnected by links $G_{\text{train}} = (\mathbf{V}_{\text{train}}, \mathbf{E}_{\text{train}})$ and $G_{\text{test}} = (\mathbf{V}_{\text{test}}, \mathbf{E}_{\text{test}})$. G_{train} is a fully labeled dataset where we are given the correct labeling assignment $\mathbf{y}_{\text{train}}$ whereas, the correct labels for G_{test} are unknown and need to be determined. Our task is to learn a probability distribution which assigns the maximum probability to the correct labeling assignment from the fully labeled training data G_{train} and then apply this learned probability distribution to determine the most likely labeling assignment for G_{test}. In this case there are two separate steps, one where we learn the distribution and two, where we apply it. Also note that G_{train} and G_{test} are distinct graphs which don't share anything in common, in particular they have distinct link structures which slightly complicates the issue since we now need to decide what we need to learn from $(G_{\text{train}}, \mathbf{y}_{\text{train}})$ so that the learned distribution can be applied to the completely different graph G_{test}.

- *Using a partially labeled dataset*: We are given an instance of relational data from which we derive a graph $G = (\mathbf{V}, \mathbf{E})$ composed of random variables interconnected by links. We are also given the correct labels $\mathbf{y}_{\text{train}}$ for a subset of the target random variables $\mathbf{Y}_{\text{train}} \subseteq \mathbf{Y} \subseteq \mathbf{V}$. We need to determine the correct labels for the remaining target random variables $\mathbf{Y}_{\text{test}} = \mathbf{Y} - \mathbf{Y}_{\text{train}}$. Note that in this formulation, unlike the former, we do not have two distinct steps, instead we have only one fully integrated step where we determine the most probable labeling assignment to $\mathbf{Y}_{\text{test}}$. Also note that we can, if we want, convert this problem into an instance of the former formulation. One could extract the part of the graph composed entirely of $\mathbf{Y}_{\text{train}}$ and the edges $\mathbf{E}_{\text{train}} = \{e | ((e.head \in \mathbf{Y}_{\text{train}}) \wedge (e.tail \in \mathbf{X})) \vee ((e.tail \in \mathbf{Y}_{\text{train}}) \wedge (e.head \in \mathbf{X})) \vee ((e.tail \in \mathbf{Y}_{\text{train}}) \wedge (e.head \in \mathbf{Y}_{\text{train}}))\}$ to form G_{train} which is now fully labeled. Similarly we can form G_{test} which contains all the target random variables that require labeling. But, as should be apparent, this results in the loss of certain edges, viz. those which connect members of $\mathbf{Y}_{\text{train}}$ to members of $\mathbf{Y}_{\text{test}}$, which might be invaluable for the task of classification.

To ground the above discussion we now present a few examples.

Example [Hypertext Classification]. Consider the webpage classification problem where G^d is composed of webpages, the words in the webpages and the hyperlinks amongst webpages. If we use the *bag of words* model, every webpage would have a set of boolean valued *word* attributes where the i^{th} attribute is an indicator variable indicating the presence or absence of the i^{th} word in the dictionary. To define G, we begin by defining $\mathbf{V}$. We introduce a random variable for every webpage which can take values from the set of class labels defined by the user. Each word attribute is represented by a binary valued random variable in G. The set of all random variables corresponding to the webpages and their word attributes constitute $\mathbf{V}$. It is common practice to introduce an edge in G between two document random variables if there exists a hyperlink between the corresponding webpages. $\mathbf{E}$ also contains edges connecting document random variables to its binary valued word random variables. This forms the edge set $\mathbf{E}$. Thus G, in this case, is made of two types of random variables and $\mathcal{T}_v$ = {'word', 'document'}. Given a collection of interlinked webpages, our aim is to classify them with their correct class labels. Suppose we want to classify the webpages either as a student homepage or a faculty homepage, then, in this case $\mathbf{L}$ = {'student', 'faculty', 'present', 'not present'} where 'student'.T = 'faculty'.T = 'document' and 'present'.T = 'not present'.T = 'word'. We assume that the content of each webpage is observed and so, all $v \in \mathbf{V}$ such that v.T = 'word' are included in the set of observed random variables $\mathbf{X}$. The unobserved random variables which we want to label are the 'document' random variables with unobserved topic labels and these form the set $\mathbf{Y}$. In this case, we have two types of edges with $\mathcal{T}_e$ = {'content', 'hyperlink'}. Every $e \in \mathbf{E}$ which connects a 'document' node to a 'word' node is of type $e.T$ = 'content' and every $e \in \mathbf{E}$ which is a consequence of a hyperlink is of type $e.T$ = 'hyperlink'.

Example [Hypertext Classification contd.]. Besides the inclusion of hyperlinks one might also want to add 'co citation' links to the above example. Here we need to augment $\mathcal{T}_e$ with the type 'co citation'. A 'co-citation' link connects two random variables $v_i, v_j \in \mathbf{V}$ if $v_i.T = v_j.T$ = 'document' and $\exists v_k \in \mathbf{V}$ along with $e_r, e_s \in \mathbf{E}$ such that $v_k.T$ = 'document', $e_r.head = v_i, e_r.tail = v_k$ and $e_s.head = v_j, e_s.tail = v_k$. Note that two nodes of type 'document' can have both an 'hyperlink' and a 'co-citation' between them.

Example [Bibliographic Classification]. The previous examples dealt with "homogeneous" target random variables, in other words, we needed to deal with only one type of target random variable. Consider the problem of classifying scientific publications or bibliographic datasets. In this domain, G^d consists of a set of scientific publications, the words in the publications and the authors who wrote the publications. Each publication is related to other publications via citations or references. A publication is also related to the authors who wrote the document. The set $\mathbf{V}$ consists of the random variables corresponding to each publication and each author in the corpus. As before, we also have in $\mathbf{V}$ observed random variables denoting the set of words in the corpus. Thus $\mathcal{T}_v$ = {'word', 'publication', 'author'}. In this problem not only do we want to classify each document by assigning it a topic label but we also want to label each author with the topic s/he publishes most often about. So $\mathbf{Y}$ consists of the set of random variables corresponding to each author and each publication

in the corpus. Suppose we consider classifying authors with one of two topic labels 'rule-based AI author' and 'probabilistic author' besides having two topic labels to label the publications with. Thus the label set is $\mathbf{L}$ = {'rule-based AI author', 'probabilistic author', 'rule-based AI publication', 'probabilistic publication' ,'present', 'not present'} with 'rule-based AI author'.T = 'probabilistic author.T' = 'author', 'rule-based AI publication'.T = 'probabilistic publication.T' = 'publication' and 'present'.T = 'not present'.T = 'word'. $\mathbf{E}$ is also considerably richer with atleast three different types of edges $\mathcal{T}_e$ = {'content', 'citation', 'authorship'} where the 'authorship' links connect the authors to their publications, 'citation' links connect publications which cite one another and the 'content' links connect publications to their word attributes. As before we can make $\mathbf{E}$ even richer by adding more types of links to it.

Example [Social Networks]. All the above examples considered relations which do not have features. Here we describe an example where the relations have descriptive attributes. This particular example (Taskar et al. [24]) also shows how a link-prediction problem can be posed as link-based classification problem. Consider the problem of predicting friendship-bonds. In this problem we introduce a random variable for every potential friendship-bond in the data graph G^d. Thus if there are N friends then we have N^2 random variables. One could imagine using a host of descriptive attributes to predict the existence of friendship-bonds. Just as we used binary valued random variables to represent words in documents in the above examples, one could use binary valued random variables which denote the mis/match of hobbies for each friendship-bond random variable. For example, suppose we want to consider using soccer as a hobby. So if both the people at either ends of the friendship-bond variable in question play soccer then we will denote this by using an observed random variable which stands for soccer and assign it the value of 'present'. We could use many such hobbies and use them as descriptive attributes for the friendship-bond random variables. For two friendship-bond random variables which emanate from the same person we place a link in G. As stated earlier, the links in this example have descriptive attributes. Any useful attribute of the person which might help in predicting her/his friendship-bonds might be used as descriptive attributes for the links in this example. For example, suppose the people on whom this data is based on are mainly composed of students then the student's major might provide important evidence. It is conceivable that students majoring in a certain subject (say Humanities) might have more friends on an average than students majoring in a different subject (say Computer Science). Such attributes are, in fact, attributes of the links.

Thus, if we have N students in G^d then we have N^2 binary valued random variables for the potential friendship-bonds which form $\mathbf{Y}$. $\mathbf{X}$ is composed of each of the random variables which record hobby-matches between two people and the random variables which denote the subjects the students major in. Thus $\mathcal{T}_v$ = {'hobby', 'major', 'friendship-bond'}. The label set is $\mathbf{L}$ = {'major-present', 'major-absent', 'hobby-match', 'hobby-mismatch', 'bond-present', 'bond-absent'} with 'major-present'.T = 'major-absent'.T = 'major', 'hobby-match'.T = 'hobby-mismatch'.T = 'hobby' and 'bond-present'.T = 'bond-absent'.T = 'friendship-bond'. We have a link between every pair of friendship-bond random variables which emanate from the same person, thus we have $N(N-1)(N-2)/2$ edges in $\mathbf{E}$ besides the edges connecting the random vari-

ables in $\mathbf{Y}$ to random variables in $\mathbf{X}$. Thus, in this case, $\mathcal{T}_e$ = {'person', 'hobby', 'major'} where 'person' links connect 'friendship-bond' random variables to other 'friendship-bond' random variables, 'hobby' links connect 'hobby' random variables to 'friendship-bond' random variables and 'major' links connect 'major' random variables to the 'friendship-bond' random variables at the ends of the 'person' links which they describe.

For the purposes of comparing various approximate collective classification algorithms (ACCA) we test each algorithm on a simplified version of the link-based classification problem:

- We will address the problem for homogeneous data where we have only one type of target random variables. In other words, $|\mathcal{T}_v| = 2$, containing a type for the target random variables and another type for the observed random variables.
- We will restrict ourselves to two types of links, or $|\mathcal{T}_e| = 2$, containing a type for the edges which connect the target random variables to observed random variables and another which connects the target random variables to other target random variables.
- In quite a few cases, due to various reasons, it becomes necessary to consider interactions amongst three or more random variables. For brevity, we will not consider correlations beyond labels of two random variables and will restrict ourselves to pairwise interactions.

Moreover, we will only consider the supervised classification scenario where we treat training and testing as two separate steps (*training with fully labeled dataset*). Using a partially labeled dataset is an interesting and, computationally, harder problem but we do not present experiments for it in this report.

Throughout the report, we will use the notation introduced in this section. Table 1 lists the important symbols for quick reference.

3 Input to Collective Classification Algorithms

Every *collective classification algorithm* (CCA) assumes some organization of the input. The input to a CCA can roughly be divided into a qualitative part and a quantitative part. The qualitative part consists of the graph of random variables $G = (\mathbf{V}, \mathbf{E})$. The quantitative part consists of a set of *parameters*. All three *approximate collective classification algorithms* (ACCAs) we discuss share the property of being *iterative*. In each iteration, the ACCA goes through G performing computations involving the G and the set of *parameters*. Each ACCA assumes a different set of *parameters* with various assumptions and semantics. Here we discuss the forms of input assumed by the ACCAs discussed in Section 4. Note that there are many other forms of input specific to CCAs not discussed here. We do not discuss those forms in this report.

3.1 Conditional Markov Networks

Markov Networks are undirected graphical models and have been used as input to CCAs. We review the basic definitions from Taskar et al. [19]. For a graph

Symbol used	Definition
G	the graph formed by random variables
$\mathbf{V}$	set of random variables in G
$\mathbf{E}$	set of edges in G
$\mathbf{Y}$	the set of target random variables which require labeling
$\mathbf{X}$	the set of observed random variables whose values are known
$\mathbf{v}$	a complete assignment to $\mathbf{V}$
$\mathbf{y}$	a complete assignment to $\mathbf{Y}$
$\mathbf{x}$	a complete assignment to $\mathbf{X}$
V_i	the i^{th} member of $\mathbf{V}$
Y_i	the i^{th} member of $\mathbf{Y}$
X_i	the i^{th} member of $\mathbf{X}$
y_i	the label assigned to Y_i
C	the set of labels we can label each $Y_i \in \mathbf{Y}$ with

Table 1: Notation used throughout this report

$G = (\mathbf{V}, \mathbf{E})$, a Markov Network defines a joint distribution over $\mathbf{V}$. It consists of the qualitative component G, which is assumed to be an undirected dependency graph, and a quantitative component, a set of parameters associated with the graph. Central to a Markov Network is the concept of a *clique*. A *clique* is a set of nodes denoted by $\mathbf{V}_c \subseteq \mathbf{V}$, not necessarily maximal, such that each $V_i, V_j \in \mathbf{V}_c$ are connected by an edge in $\mathbf{E}$. Note that a single node can also be considered a clique.

Definition [Markov Networks]. Let $G = (\mathbf{V}, \mathbf{E})$ be an undirected graph with a set of cliques $C(G)$. Each $c \in C(G)$ is associated with a set of nodes $\mathbf{V}_c$ and a *clique potential* $\psi_c(\mathbf{V}_c)$, which is a non-negative function defined on the joint domain of $\mathbf{V}_c$. Let $\Psi = \{\psi_c(\mathbf{V}_c)\}_{c \in C(G)}$. The Markov network (G, Ψ) defines the distribution $P(\mathbf{v}) = \frac{1}{Z} \prod_{c \in C(G)} \psi_c(\mathbf{v}_c)$, where $\mathbf{v}$ denotes an assignment to $\mathbf{V}$, $\mathbf{v}_c$ denotes an assignment to $\mathbf{V}_c$ and Z is the partition function $Z = \sum_{\mathbf{v}'} \prod_{c \in C(G)} \psi_c(\mathbf{v}'_c)$.

For link-based classification, we will primarily be interested in *conditional markov networks* which are simply markov networks renormalized given $\mathbf{X}$, the set of observed random variables, to produce a conditional distribution.

Definition [Conditional Markov Networks]. A *conditional markov network* is a markov network (G, Ψ) which defines the distribution $P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C(G)} \psi_c(\mathbf{x}, \mathbf{y}_c)$ where $Z(\mathbf{x})$ is the partition function now dependent on $\mathbf{x}$: $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \prod_{c \in C(G)} \psi_c(\mathbf{x}, \mathbf{y}'_c)$.

For simplicity we will only use pairwise interactions which can be represented by *Pairwise Markov Networks*. *Pairwise Markov Networks* are markov networks whose set of cliques is composed of the target random variables and the set of edges in the graph, $C(G) = \mathbf{Y} \cup \mathbf{E}$. We use $(Y_i, Y_j) \in \mathbf{E}$ to denote a clique formed by $Y_i, Y_j \in \mathbf{Y}$ and $\psi_{i,j}(y_i, y_j)$ to denote the corresponding clique potential.

In a *Pairwise Markov Network*, for each node clique potential, it is possible to absorb the evidence for Y_i to form a new clique potential denoted by $\phi_i(Y_i = y_i)$:

$$\phi(Y_i = y_i) = \psi_i(y_i) \prod_{e=(Y_i,X_j) \wedge e \in C(G)} \psi_e(y_i, x_j)$$

This will help simplify notation when we discuss the various ACCAs.

3.2 Neighbourhood based formulations

Another common way to present input to a CCA is to describe each random variable in terms of its neighbours in G. Let $\mathcal{N}(Y_i) = \{V | V \in \mathbf{V}, (Y_i, V) \in \mathbf{E}\}$ denote the set of neighbours of Y_i in G. The qualitative part of the input is represented by the collection of target random variables which need to be classified $\mathbf{Y}$ and for each $Y_i \in \mathbf{Y}$, $\mathcal{N}(Y_i)$ its set of neighbours in G.

The quantitative part of the input consists of a function $g(y|\mathbf{v}_{\mathcal{N}(Y_i)}; \mathbf{w})$ which can compute the probability of an assignment to Y_i given the assignments to its neighbourhood $pr(Y_i = y|\mathbf{v}_{\mathcal{N}(Y_i)})$. $\mathbf{w}$ is the set of parameters to the function g which we refer to as the "local classifier".

A major difference between the previous input formulation in terms of markov networks and this one is that, a neighbourhood based formulation does not assume a form of the distribution for the local classifier whereas a markov network imposes an exponential distribution. Thus, if in some domain there is reason to believe that a particular classifier might perform well, a CCA which assumes input organized according to the neighbourhood based formulation has a better chance to incorporate the classifier than a CCA which assumes a markov network.

4 Collective Classification Methods

The most important step in link-based classification is to determine the set of labels $\mathbf{y}$ to the target random variables $\mathbf{Y}$ which minimizes the *zero-one loss* or *error-rate*, in other words, the number of incorrect classifications. Any algorithm which does this using the links $\mathbf{E}$ in G is called a *collective classification algorithm* (CCA). Link-based classification usually involves data with large graphs and exact inference is seldom tractable. Any algorithm which finds the optimal set of labels *approximately* is called an *approximate collective classification algorithm* (ACCA).

In this section we discuss three of the most popular ACCAs used to determine the most probable assignment to each target random variable in G. Each of these algorithms assume a slightly different input. For each algorithm we will first describe the input, then the algorithm and finally, the algorithm's justification if there exist any.

4.1 Loopy Belief Propagation (LBP)

LBP assumes that (G, Ψ) is a *pairwise Markov Network* [19] with $C(G)$ denoting its set of cliques where $\Psi = \{\psi_c(v_c)\}_{c \in C(G)}$ is the set of non-negative *clique potentials*. We use $\phi_i(y_i)$ to denote the modified node clique potentials.

4.1.1 The algorithm (Yedidia et al. [30])

LBP is a message passing algorithm and can be expressed as follows:

$$m_{i\to j}(y_j) = \alpha \sum_{y_i \in C} \psi_{i,j}(y_i, y_j)\phi_i(y_i) \prod_{Y_k \in \mathcal{N}(Y_i)\setminus Y_j} m_{k\to i}(y_i), \quad \forall y_j \in C$$

$$b_i(y_i) = \alpha \phi_i(y_i) \prod_{Y_j \in \mathcal{N}(Y_i)} m_{j\to i}(y_i), \quad \forall y_i \in C$$

where $m_{i\to j}$ is a message sent by Y_i to Y_j, α is a generic normalization constant and $\mathcal{N}(Y)$ denotes the set of target random variables in the neighborhood of Y. The algorithm proceeds by making all the target random variables communicate messages until the messages stabilize. When the messages stabilize we compute $b_i(y_i)$ (approximate marginal probability) for every target random variable Y_i for every label y_i and assign Y_i the label y_i with highest marginal probability. Algorithm 1 describes the pseudocode.

Algorithm 1 Loopy Belief Propagation

```
for each Y_i ∈ Y do {initialize all messages}
  for each Y_j ∈ Y s.t. (Y_i, Y_j) ∈ C(G) do
    for each y_j ∈ C do
      m_{i→j}(y_j) ← 1
    end for
  end for
end for
repeat {perform message passing}
  for each Y_i ∈ Y do
    for each Y_j ∈ Y s.t. (Y_i, Y_j) ∈ C(G) do
      for each y_j ∈ C do
        m_{i→j}(y_j) ← α Σ_{y_i} ψ_{i,j}(y_i, y_j)φ_i(y_i) Π_{Y_k ∈ N(Y_i)\Y_j} m_{k→i}(y_i)
      end for
    end for
  end for
until all m_{i→j}(y_j) stop showing any change
for each Y_i ∈ Y do {compute beliefs}
  for each y_i ∈ C do
    b_i(y_i) = α φ_i(y_i) Π_{Y_j ∈ N(Y_i)} m_{j→i}(y_i)
  end for
end for
```

4.1.2 Justification for LBP (Yedidia et al. [29])

Yedidia et al. justifies LBP as a variational method and here we review their justification.

The joint probability distribution defined by a pairwise markov network ([19]) is:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{(Y_i,Y_j)\in C(G)} \psi_{i,j}(y_i, y_j) \prod_{Y_i \in C(G)} \phi_i(y_i)$$

Energy of a configuration:	$E(\mathbf{y})$	$= -\sum_{(Y_i,Y_j)\in C(G)} \log \psi_{i,j}(y_i, y_j)$ $-\sum_{Y_i \in C(G)} \log \phi_i(y_i)$
Helmholtz free energy:	F_H	$= -\log Z$
variational average energy:	$U(b)$	$= \sum_{\mathbf{y}} b(\mathbf{y}) E(\mathbf{y})$
variational entropy:	$H(b)$	$= -\sum_{\mathbf{y}} b(\mathbf{y}) \ln b(\mathbf{y})$
variational free energy:	$F(b)$	$= U(b) - H(b)$

Table 2: Definitions from Yedidia et al. [29]

where Z is a normalization constant known as the *partition function*:

$$Z = \sum_{\mathbf{y}} \prod_{(Y_i,Y_j)\in C(G)} \psi_{i,j}(y_i, y_j) \prod_{Y_i \in C(G)} \phi_i(y_i)$$

To compute the optimal labels $\forall Y \in \mathbf{Y}$ we need to compute $p(\mathbf{y}|\mathbf{x})$ (the probability of a complete assignment $\mathbf{y}$) and choose the $\mathbf{y}$ with the maximum $p(\mathbf{y}|\mathbf{x})$. One way to compute $p(\mathbf{y}|\mathbf{x})$ is to use a *variational* approach by introducing a trial probability distribution $b(\mathbf{y})$.

In Table 2, we reproduce some of the definitions introduced in Yedidia et al. [29]. From these definitions it follows that:

$$F(b) = F_H + \mathrm{KL}(b||p)$$

where $KL(b||p)$ is the Kullback-Leibler divergence between $b(\mathbf{y})$ and $p(\mathbf{y}|\mathbf{x})$. Since KL divergence is non-negative, minimizing $F(b)$ with respect to $b(\mathbf{y})$ is an exact procedure to compute F_H and recover $p(\mathbf{y}|\mathbf{x})$. But this is too difficult. A more practical approach is to minimize $F(b)$ over a restricted class of probability distributions $b(\mathbf{y})$.

Yedidia et al. [30] justifies LBP as a variational method by choosing the following form for $b(\mathbf{Y})$:

$$b(\mathbf{Y}) = \frac{\prod_{(Y_i,Y_j)\in C(G)} b_{i,j}(y_i, y_j)}{\prod_{Y_i \in \mathbf{Y}} b_i(y_i)^{|\mathcal{N}(Y_i)|-1}}$$

where $b_c(y_c)$ denotes the marginal probability of assigning clique $c \in C(G)$ labels y_c.

The above distribution is equal to the probability distribution defined by a pairwise markov network only when G is singly connected or tree structured ([29]). In all other cases it is, at best, an approximation. With this form in

mind, we can define the following two quantities:

$$\begin{aligned} U_{\text{Bethe}} &= + \sum_{(Y_i,Y_j)\in C(G)} \sum_{y_i,y_j \in C} b_{i,j}(y_i,y_j)[\log \psi_{(i,j)}(y_i,y_j) \\ &\quad + \log \phi_i(y_i) + \log \phi_j(y_j)] \\ &\quad - \sum_{Y_i \in \mathbf{Y}} (|\mathcal{N}(Y_i)| - 1) \sum_{y_i \in C} b_i(y_i) \log \phi_i(y_i) \\ H_{\text{Bethe}} &= + \sum_{(Y_i,Y_j)\in C(G)} \sum_{y_i,y_j \in C} b_{i,j}(y_i,y_j) \log b_{i,j}(y_i,y_j) \\ &\quad - \sum_{Y_i \in \mathbf{Y}} (|\mathcal{N}(Y_i)| - 1) \sum_{y_i \in C} b_i(y_i) \log b_i(y_i) \end{aligned}$$

Note that H_{Bethe} is quite different from $H(b)$ we defined earlier (or as Yedidia et al. puts it, "...the entropy term is incorrect..."). It is another approximation introduced to provide computational feasibility. Thus the *Bethe Free Energy* (F_{Bethe}) is defined to be:

$$F_{\text{Bethe}} = U_{\text{Bethe}} - H_{\text{Bethe}}$$

The *constrained* Bethe free energy is defined by enforcing that the trial distribution obeys the normalization constraints:

$$\sum_{y_i \in C} b_i(y_i) = 1, \quad \forall Y_i \in \mathbf{Y}$$

the marginalization constraints:

$$\sum_{y_i \in C} b_{i,j}(y_i,y_j) = b_j(y_j), \quad \forall y_j \in C, \forall (Y_i,Y_j) \in C(G)$$

and the inequality constraints:

$$0 \leq b_i(y_i) \leq 1, \quad \forall y_i \in C, \forall Y_i \in \mathbf{Y}$$

Yedidia et al. prove that the stationary points of the constrained Bethe free energy correspond to the fixed points of the LBP algorithm described in Section 4.1.1.

4.2 Relaxation Labeling

Here we describe two relaxation labeling algorithms. The first algorithm Rosenfeld et al. [17] was proposed in the vision community and is considered to be one of the first relaxation labeling algorithms. The second algorithm is also referred to as Mean Field and has close connections to LBP.

4.2.1 The algorithm (Rosenfeld et al. [17])

This version of relaxation labeling assumes a pairwise markov network as input. Let $\mathcal{N}(Y)$ denote set of target random variables in the neighbourhood of Y, in other words, $\mathcal{N}(Y) = \{Y_j | (Y,Y_j) \in C(G)\}$. Instead of assuming clique

potentials, this version of relaxation labeling assumes a slightly different set of parameters. We assume that $d_{i,j}$ denotes the "importance" of target random variable $Y_j \in \mathcal{N}(Y_i)$ to $Y_i \in \mathbf{Y}$ and that $\sum_{Y_j \in \mathcal{N}(Y_i)} d_{i,j} = 1$. $d_{i,j}$ may be such that $d_{i,j} = d_{i,k} \quad \forall Y_j, Y_k \in \mathcal{N}(Y_i)$. We also assume that we are given $r_{i,j}(y_i, y_j)$ which denotes the "compatibility" of labels y_i and y_j $\forall Y_j \in \mathcal{N}(Y_i), \quad \forall Y_i \in \mathbf{Y}$. Further, we assume that $\forall Y_i, Y_j \in \mathbf{Y}$ $|r_{i,j}(y_i, y_j)| \leq 1$ $\forall y_i, y_j \in C$. For the time being we will ignore $\mathbf{X}$. We will have something to say about it later.

Let $b_i(y_i)$ denote the marginal probability of label y_i being assigned to target random variable Y_i. $b_i(y_i)$ is of course a normalized probability distribution obeying $\sum_{y_i} b_i(y_i) = 1, \quad \forall Y_i \in \mathbf{Y}$. Rosenfeld et al. suggest the following relaxation labeling iteration:

$$\begin{aligned} q_j(y_j) &= \sum_{Y_i \in \mathcal{N}(Y_j)} d_{j,i} \left[\sum_{y_i \in C} r_{j,i}(y_j, y_i) b_i(y_i) \right] \\ b_j(y_j) &= \alpha b_j(y_j) \left[1 + q_j(y_j) \right] \end{aligned}$$

where α is a normalization constant. Rosenfeld et al.'s relaxation labeling iteration does not take the "local" evidence into consideration. It is conceivable that we can initialize $b_i(y_i)$ to probabilities depending on the "local" evidence and then continue with the above relaxation labeling iterations. Algorithm 2 describes the pseudocode where we initialize $b_i(y_i)$ with the local evidence. Rosenfeld et al. suggest setting $r_{i,j}(y_i, y_j)$ to the correlation of the labels y_i and y_j which would satisfy $|r_{i,j}(y_i, y_j)| \leq 1$.

Algorithm 2 Relaxation Labeling

```
1:  for each Y_i ∈ Y do {initialize messages with local evidence}
2:    for each y_i ∈ C do
3:      b_i(y_i) ← α φ_i(y_i)
4:    end for
5:  end for
6:  repeat {perform message passing}
7:    for each Y_j ∈ Y do
8:      for each y_j ∈ C do
9:        q_j(y_j) = Σ_{Y_i ∈ N(Y_j)} d_{j,i} [ Σ_{y_i} r_{j,i}(y_j, y_i) b_i(y_i) ]
10:       b_j(y_j) = α b_j(y_j) [1 + q_j(y_j)]
11:     end for
12:   end for
13: until all b_j(y_j) stop changing
```

This form of relaxation labeling was proposed as a heuristic. There have been many attempts to justify and modify Rosenfeld et al.'s relaxation labeling iteration. We refer the reader to Li et al. [13] for a comparison of various relaxation labeling iterators. One particular iteration has very close connections to LBP since it can be justified in almost the same way and we derive it next.

4.2.2 Relaxation Labeling via Mean-Field Approach (MF)(Yedidia et al. [29], Li et al. [13])

Mean-Field Relaxation Labeling assumes that (G, Ψ) is a *pairwise Markov Network* [19] with $C(G)$ denoting its set of cliques where $\Psi = \{\psi_c(v_c)\}_{c \in C(G)}$ is the set of non-negative *clique potentials*. We use $\phi_i(y_i)$ to denote the modified node clique potentials.

Recall that if we define the variational free energy to be:

$$F(b) = U(b) - H(b)$$

then it can be shown using the definitions in Table 2 that:

$$F(b) = F_H + KL(b||p)$$

A very simple trial distribution is the factorized form:

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

We also define:

$$\begin{aligned} U_{\text{MF}} &= + \sum_{(Y_i,Y_j) \in C(G)} \sum_{y_i, y_j \in C} b_i(y_i) b_j(y_j) \log \psi_{i,j}(y_i, y_j) \\ &\quad + \sum_{Y_i \in \mathbf{Y}} \sum_{y_i \in C} b_i(y_i) \log \phi_i(y_i) \\ H_{\text{MF}} &= \sum_{Y_i \in \mathbf{Y}} \sum_{y_i \in C} b_i(y_i) \log b_i(y_i) \end{aligned}$$

Note the two approximations made, first in choosing the trial distribution and second, while defining H_{MF}. We define the MF variational free energy to be:

$$F_{\text{MF}} = U_{\text{MF}} - H_{\text{MF}}$$

Now we define the constrained variational free energy optimization to be:

$$\begin{array}{rc} \text{minimize} & F_{MF} \\ \text{with respect to} & \sum_{y_i \in C} b_i(y_i) = 1, \forall Y_i \in \mathbf{Y} \end{array}$$

The lagrangian formulation is thus:

$$\begin{aligned} L &= + \sum_{(Y_i,Y_j) \in C(G)} \sum_{y_i, y_j \in C} b_i(y_i) b_j(y_j) \log \psi_{i,j}(y_i, y_j) \\ &\quad + \sum_{Y_i \in \mathbf{Y}} \sum_{y_i \in C} b_i(y_i) \log \phi_i(y_i) - \sum_{Y_i \in \mathbf{Y}} \sum_{y_i \in C} b_i(y_i) \log b_i(y_i) \\ &\quad - \sum_{Y_i} \lambda_i \Big(\sum_{y_i \in C} b_i(y_i) - 1 \Big) \end{aligned}$$

$$\frac{\partial L}{\partial b_j(y_j)} = + \sum_{Y_i \in \mathcal{N}(Y_j)} \sum_{y_i \in C} b_i(y_i) \log \psi_{i,j}(y_i, y_j) + \log \phi_j(y_j) - \log b_j(y_j) - 1 - \lambda_i$$

To minimize we set the partial derivative to zero:

$$b_j(y_j) = \alpha \phi_j(y_j) \prod_{Y_i \in \mathcal{N}(Y_j)} \prod_{y_i \in C} \psi_{i,j}^{b_i(y_i)}(y_i, y_j)$$

where α is a normalization constant. Algorithm 3 describes the pseudocode for this fixed point iteration.

Algorithm 3 Mean Field

```
1: for each Y_i ∈ Y do {initialize message}
2:    for each y_i ∈ C do
3:       b_i(y_i) ← 1
4:    end for
5: end for
6: repeat {perform message passing}
7:    for each Y_j ∈ Y do
8:       for each y_j ∈ C do
9:          b_j(y_j) = α φ_j(y_j) ∏_{Y_i ∈ N(Y_j)} ∏_{y_i ∈ C} ψ_{i,j}^{b_i(y_i)}(y_i, y_j)
10:      end for
11:   end for
12: until all b_j(y_j) stop changing
```

Notably, one of the earliest papers on link-based classification, Chakrabarti et al. [6], used relaxation labeling although their formulation of U_{MF} was based on the Naive Bayes classifier.

In our experiments, MF always produced better results than the Rosenfeld et al. [17]'s version of relaxation labeling. Hence we report results for MF only.

4.3 Iterative Classification

Iterative Classification Algorithm assumes a neighbourhood based formulation of the input. Let $\mathcal{N}(Y)$ denote the neighbourhood relation. Let $g(y|\mathbf{v}_{\mathcal{N}(Y)}; \mathbf{w})$ denote the "local classifier" which returns the class conditional probability of assigning label y to Y conditioned on the assignments to its neighbourhood. Usually, computing the "local" class conditional probability involves computing two multi-sets: $OA(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)})$ and $LD(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})$. For every $Y_i \in \mathbf{Y}$, $OA(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)})$ denotes the set of values of the observed random variables in $\mathcal{N}(Y_i)$. OA stands for *object attributes*. $LD(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})$ is a multi-set formed by the labels of the target random variables in $\mathcal{N}(Y_i)$. Both $OA(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)})$ and $LD(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})$ are usually represented as fixed length vectors. In the case of $LD(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})$ this might require the use of an *aggregation function* ([10]).

Neville and Jensen proposed one of the earliest iterative classification algorithms. Neville and Jensen refer to $LD(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})$ as *dynamic attributes* which change after every iteration of relabeling thus necessitating recomputation to

decide on the current label. $OA(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)})$ is referred to as *static attributes* and don't change throughout the run of the classification algorithm. Algorithm 4 describes the iterative classification algorithm proposed in Neville and Jensen.

Algorithm 4 Iterative Classification Algorithm(M) [16]

for each $Y_i \in \mathbf{Y}$ **do**
  Compute $g(y_i|\mathbf{v}_{\mathcal{N}(Y_i);\mathbf{w}})$ only using $OA(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)})$ $\forall y_i \in C$.
  Set $y_i \leftarrow \text{argmax}_y\ g(y|\mathbf{v}_{\mathcal{N}(Y_i)})$
end for
for i=1 to M **do**
  for each $Y_i \in \mathbf{Y}$ **do**
    Compute $g(y_i|\mathbf{v}_{\mathcal{N}(Y_i)};\mathbf{w})\forall y_i \in C$.
    Store $p_i \leftarrow \max_y\ g(y|\mathbf{v}_{\mathcal{N}(Y_i)})$
    Store $y_i \leftarrow \text{argmax}_y\ g(y|\mathbf{v}_{\mathcal{N}(Y_i)})$
  end for
  $k \leftarrow |\mathbf{Y}|\frac{i}{M}$
  Choose the top-k p_i and assign the corresponding Y_i their new labels.
end for

Algorithm 4 begins by classifying each node using its content. Each iteration consists of recomputing each node's class label given its neighbourhood classifications. At the end of each iteration, Algorithm 4 chooses the top-k recomputed class labels corresponding to the highest posterior probabilities and discards the others. The number of iterations is fixed to M. In each subsequent iteration a greater number of relabelings are accepted. The final iteration accepts relabelings for all target random variables.

Subsequent researchers ([14]) have added an extra functionality to the basic ICA. Lu and Getoor [14] propose adding an order computation step at the beginning of the iterative step in ICA. This computes the order in which the nodes in the graph will be visited for recomputation of class labels. One can choose from a variety of different *ordering strategies*. For example, Lu and Getoor describes an ordering strategy based on the diversity of $LD(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})$. For example, given two target random variables $Y_i, Y_j \in \mathbf{Y}$, Lu and Getoor describe an ordering strategy where Y_i will be relabeled after Y_j in the current iteration if $LD(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})$ has higher diversity than $LD(Y_j|\mathbf{y}_{\mathcal{N}(Y_j)})$, the hypothesis being that target random variables with less diversity in their dynamic attributes will be easy to label. Lu and Getoor observed no visible effect of the ordering strategy on the final accuracy attained. The speed of convergence, on the other hand, did depend on the ordering strategy used. We refer the reader to Getoor [9] for a comparison of various ordering strategies.

Due to its simplicity and speed, many different researchers have used ICA in one form or the other for different applications but it isn't clear what iterative classification is trying to achieve. More precisely, when ICA relabels a target random variable $Y \in \mathbf{Y}$ it does not care to see how this relabeling is going to affect the class conditional probabilities of the target random variables in $\mathcal{N}(Y)$. This is in contrast to the other two ACCAs which we discussed. Both LBP and relaxation labeling involve some sort of a gradient term which tries to take into account the effect of changing marginal probabilities of the neighborhood while recomputing messages. ICA seems to be the most "myopic" of the four ACCAs

we discussed.

5 Learning Algorithm and Classifier Used

To compare the three ACCAs we trained three different maximum entropy classifiers, one for each ACCA. In this section we describe the learning algorithms used to learn the quantitative parts of the input to each ACCA.

5.1 Learning for LBP and MF

Both LBP and MF accept a *pairwise markov network* as input. The quantitative part of the input to a *pairwise markov network* is the set of *clique potentials* $\{\psi_c(\mathbf{v}_c)\}_{c \in C(G)}$ where $C(G)$ is the set of cliques in the pairwise markov network. Notice that the parameters are clique-specific since $\psi_c(\mathbf{v}_c)$ is the clique potential for clique c. This creates a problem for us. Recall that our aim is to learn a classifier from fully labeled training data and apply the classifier to classify unseen test data (*training with fully labeled data*). There is no guarantee that the training data and test data will share the same set of cliques, in fact, usually they are not the same. The common approach to get around this problem is to learn a *different set of parameters* instead of learning the set of clique potentials. We employ a set of features which can be used to describe any clique. Usually, each feature is a simple indicator function. One defines a parameter for each feature employed and the parameters of the classifier are thus the set of parameters $\mathbf{w}$ for all the features $\mathbf{f}$. Finally, one defines the clique potential in log-space:

$$\psi_c(y_c, \mathbf{x}) = \exp(\mathbf{w} \cdot \mathbf{f}(y_c, \mathbf{x}))$$

The maximum entropy approach to learn the parameters of a classifier is to maximize entropy while constraining the expected feature counts to be equal to the empirical feature counts. The dual of this problem is known to be the maximum likelihood estimation problem where one attempts to find the parameter values which maximize the probability of the labeled training set:

$$\max \frac{1}{Z(\mathbf{x})} \exp\left(\mathbf{w} \cdot \sum_{c \in C(G)} \mathbf{f}(y_c, \mathbf{x})\right)$$

where $Z(\mathbf{x})$ is the partition function.

Usually, one also employs a prior over the parameters to avoid overfitting. One popular choice of prior is the "shrinkage prior" which is a set of independent zero-mean gaussians one for each parameter $p(w_i) = \exp(-\lambda w_i^2)$ where λ is the regularization constant. Subsequently one learns the parameter values by maximum a posteriori (MAP) estimation.

$$l = \mathbf{w} \cdot \sum_{c \in C(G)} \mathbf{f}(y_c, \mathbf{x}) - \log Z(\mathbf{x}) - \lambda \mathbf{w} \cdot \mathbf{w}$$

To maximize l we can employ a host of gradient-based optimization algorithms (eg: conjugate gradient descent) which move in a direction computed from the gradient of the function to be optimized. The gradient of l with respect to $\mathbf{w}$ is:

$$\sum_{c \in C(G)} \mathbf{f}(y_c, \mathbf{x}) - \sum_{\mathbf{y}} p^{\mathbf{w}}(\mathbf{y}) \sum_{c \in C(G)} \mathbf{f}(y_c, \mathbf{x}) - 2\lambda \mathbf{w}$$

which can be rewritten as:

$$\sum_{c \in C(G)} \mathbf{f}(y_c, \mathbf{x}) - \sum_{c \in C(G)} \sum_{y_c} \mu_c^{\mathbf{w}}(y_c) \mathbf{f}(y_c, \mathbf{x}) - 2\lambda \mathbf{w}$$

To compute the the gradient we need to compute the marginals under the current parameter settings (viz. $\mu_c^{\mathbf{w}}(y_c)$) hence the need to perform inference. For pairwise markov networks we need to compute the marginals for each $Y_i \in \mathbf{Y}$ and $(Y_i, Y_j) \in \mathbf{E}$. For LBP, these marginals can be computed once the message passing has converged by the following equations:

$$\begin{aligned} b_{i,j}(y_i, y_j) &= \alpha \gamma_{i,j}(y_i, y_j) \prod_{Y_k \in \mathcal{N}(Y_i) \setminus Y_j} m_{k \to i}(y_i) \prod_{Y_k \in \mathcal{N}(Y_j) \setminus Y_i} m_{k \to j}(y_j) \quad \forall y_i, y_j \in C \\ b_i(y_i) &= \alpha \phi_i(y_i) \prod_{Y_j \in \mathcal{N}(Y_i)} m_{j \to i}(y_i), \quad \forall y_i \in C \end{aligned}$$

where $\gamma(y_i, y_j) = \phi_i(y_i)\phi_j(y_j)\psi_{i,j}(y_i, y_j)$.

For MF, we obtain the node marginals once the fixed point has been achieved. The edge marginals for MF are defined to be:

$$b_{i,j}(y_i, y_j) = b_i(y_i) b_j(y_j)$$

where $b_i(y_i)$ is the node marginal for Y_i being labeled with y_i.

5.2 Learning for ICA

In the case of ICA, we need to learn the parameters of the "local classifier" function $g(y|\mathbf{v}_{\mathcal{N}(Y)}; \mathbf{w})$. We partitioned $\mathbf{w}$ into $\mathbf{w}_y$, one for each label $y \in C$. For our experiments, we used the following form for the "local classifier":

$$g(y|\mathbf{v}_{\mathcal{N}(Y)}; \mathbf{w}) = \frac{\exp(\mathbf{w}_y \cdot (OA(Y|\mathbf{x}_{\mathcal{N}(Y)}), LD(Y|\mathbf{y}_{\mathcal{N}(Y)})))}{\sum_{y'} \exp(\mathbf{w}_{y'} \cdot (OA(Y|\mathbf{x}_{\mathcal{N}(Y)}), LD(Y|\mathbf{y}_{\mathcal{N}(Y)})))} \tag{1}$$

We concatenated the two fixed length vectors $OA(Y|\mathbf{x}_{\mathcal{N}(Y)})$ and $LD(Y|\mathbf{y}_{\mathcal{N}(Y)})$ to form one fixed length vector. To learn the parameters we performed the following optimization:

$$\mathbf{w} = \text{argmax}_{\mathbf{w}} \prod_{Y_i \in \mathbf{Y}} g(t(i)|\mathbf{v}^t_{\mathcal{N}(Y_i)}; \mathbf{w}) \prod_{y \in C} \exp(-\lambda ||w_y||^2)$$

where $t(i)$ denotes the label of Y_i in the training set and $\mathbf{v}^t_{\mathcal{N}(Y_i)}$ denotes the labels of the neighbours in the training set. We used gradient based optimization methods (like conjugate gradient descent) to perform this optimization. We also included a "shrinkage" prior over the parameters.

$$L = \prod_{Y_i \in \mathbf{Y}} \frac{\exp(\mathbf{w}_{t(i)} \cdot (OA^t(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)}), LD^t(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})))}{\sum_{y'} \exp(\mathbf{w}_{y'} \cdot (OA^t(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)}), LD^t(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})))} \prod_{y \in C} \exp(-\lambda ||w_y||^2)$$

Note that $OA^t(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)})$ and $LD^t(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})$ can be computed by looking at the training set and its labels.

$$
\begin{aligned}
l &= \sum_{Y_i \in \mathbf{Y}} \mathbf{w}_{t(i)} \cdot (OA^t(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)}), LD^t(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})) - \\
&\quad \sum_{Y_i \in \mathbf{Y}} \log \left[\sum_{y'} \exp(\mathbf{w}_{y'} \cdot (OA^t(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)}), LD^t(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)}))) \right] - \\
&\quad \sum_{y \in C} \lambda ||\mathbf{w}_y||^2 \\
\frac{\partial l}{\mathbf{w}_y} &= \sum_{Y_i \in \mathbf{Y} \wedge t(i)=y} (OA^t(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)}), LD^t(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})) - \\
&\quad \sum_{Y_i \in \mathbf{Y}} p^{\mathbf{w}}(y|OA^t(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)}), LD^t(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)}))(OA^t(Y_i|\mathbf{x}_{\mathcal{N}(Y_i)}), LD^t(Y_i|\mathbf{y}_{\mathcal{N}(Y_i)})) - \\
&\quad 2\lambda \mathbf{w}_y
\end{aligned}
$$

where $p^{\mathbf{w}}(y|\ldots)$ is the conditional probability of labeling Y_i with y given its neighbourhood labels $\mathbf{v}_{\mathcal{N}(Y_i)}$ under the current $\mathbf{w}$. We computed $p^{\mathbf{w}}(y|\ldots)$ using Eq. (1).

6 Experiments

We tested the performance of the three ACCAs on the task of link-based document classification by using both synthetic datasets and real world datasets. Besides the three classifiers learnt using the various inference methods we also use, as a baseline, a maximum entropy classifier which only looks at the content of each node in the graph. This *content-only* classifier does not take the links into account and has less parameters. In all our experiments (barring one) we initialized all our parameters using random numbers.

For the document classification task we have two types of random variables $\mathcal{T}_v = \{'document', 'word'\}$. A *'document'* random variable can be connected to an observed *'word'* random variable (which can be either *'present'* or *'not present'*) via a *'content'* link. Two *'document'* random variables can be connected via links of type *'hyperlinks'* or *'citations'* depending on the dataset used for the experiments. For all our experiments we dropped all the directions of the edges and experimented with undirected graphs. For training, we used fully labeled datasets where we were given the correct class labels of each document in the training corpus. For testing we used datasets where the class labels of a subset of the document random variables were unknown and required prediction (the target random variables). Note that during testing any document random variables with a known class label was treated as evidence.

6.1 Features used

For LBP and MF we used a small set of indicator functions for features. For example, we used an indicator function which indicated the presence or absence of class label c given a *'document'* random variable. A different type of feature indicated the presence or absence of a class label c and a word i from the

vocabulary given a pair of random variables connected by a link of type *'content'* (we also experimented with features that counted the number of occurrences of word i from the vocabulary in the document but this did not provide any significant improvement). The third set of features we used were indicator functions which indicated the presence or absence of a pair of class labels given a pair of connected *document* random variables.

For ICA, we used a set of features to capture dependencies between neighbouring class labels and dependencies between class labels and attribute values. Recall that we use a "local classifier" $g(y|\mathbf{v}_{\mathcal{N}(Y)}; \mathbf{w})$ which returns the returns conditional probability of labeling $Y \in \mathbf{Y}$ with label y given a fully labeled neighbourhood. Also recall that we modeled this local classifier as a selector function. We divided the set of parameters $\mathbf{w}$ into $|C|$ equal parts $\mathbf{w}_y$ $y \in C$, one for each label. To compute $pr(y|\mathbf{v}_{\mathcal{N}(Y)})$ we first select $\mathbf{w}_y$, then compute $OA(Y|\mathbf{x}_{\mathcal{N}(Y)})$ and $LD(Y|\mathbf{y}_{\mathcal{N}(Y)})$ and finally, take the dot product of $\mathbf{w}_y$ with the concatenation of $OA(Y|\mathbf{x}_{\mathcal{N}(Y)})$ and $LD(Y|\mathbf{y}_{\mathcal{N}(Y)})$. To compute $OA(Y|\mathbf{x}_{\mathcal{N}(Y)})$ we used a set of indicator functions which detected the presence/absence of words from the vocabulary. For $LD(Y|\mathbf{y}_{\mathcal{N}(Y)})$, we used the count aggregate function to count the number of each label instance in the neighbourhood.

6.2 Synthetic Data Generator

Our algorithm for generating synthetic datasets closely follows the algorithm described in Bollobas et al. [4]. The algorithm is outlined in Algorithm 5.

Algorithm 5 Synthetic data generator(*numNodes*, *alpha*, *numLabels*, *attrNoise*, *numObs*, *vocabSize*)

```
1:  Set i=0
2:  G = ∅
3:  while i < numNodes do
4:     Sample r ∈ [0, 1] uniformly at random
5:     if r <= α then
6:        connectNode(G, numLabels)
7:     else
8:        addNode(G, numLabels)
9:        i ← i + 1
10:    end if
11: end while
12: for i = 1 to numNodes do
13:    v ← i^th node in G
14:    genAttributes(v, numLabels, vocabSize, numObs, attrNoise)
15: end for
16: return G
```

The algorithm "grows" a graph from an empty set of nodes. The user is expected to supply the number of nodes the final generated graph should contain. α is a parameter which controls the number of links in the graph. Roughly, the final graph should contain $\frac{1}{1-\alpha} numNodes$ number of links. The function $chooseNewNodeClass()$ can be used to implement a set of class priors. In all our experiments with synthetic data we used a set of uniform class priors.

Algorithm 6 addNode(G, $numLabels$)

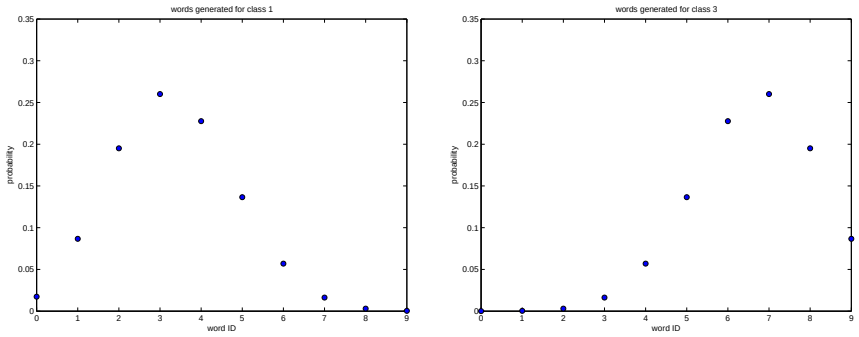
add a new node v to G
$c \leftarrow chooseNewNodeClass()$
Set $v.label \leftarrow c$
$c_n \leftarrow chooseClass(c, numLabels)$
$w \leftarrow$ select a node from G with $w.label = c_n$ and probability of selection proportional to its out-degree
introduce an edge from v to w

Algorithm 7 connectNode(G, numLabels)

$v \leftarrow$ select any node at random from G
$c \leftarrow v.label$
$c_n \leftarrow chooseClass(c, numLabels)$
$w \leftarrow$ select a node from G with $w.label = c_n$ and probability of selection proportional to its out-degree
introduce an edge from v to w

Algorithm 8 genAttributes(v, $numLabels$, $vocabSize$, $numObs$, $attrNoise$)

for i=1 to $numObs$ **do**
  sample r uniformly at random
  if $r \leq attrNoise$ **then**
    sample a word uniformly at random from the vocabulary and add it to v
  else
    sample a word-ID from the binomial distribution with $p = (1 + v.label)/(1 + numLabels)$ and $n = vocabSize$
  end if
end for



(a) Word-ids sampled for class 1 (b) Word-ids samples for class 3

Figure 1: Different binomial distributions used to sample words from for each class label by the data generator with $vocabSize = 10$ and $numLabels = 5$.

name	value
$numNodes$ for training set	3000
$numNodes$ for test set	1000
$numLabels$	5
$vocabSize$	10
$numObs$ (maximum number of words in a node)	5

Table 3: Parameter settings for our synthetic data generator

The algorithm implements a rudimentary form of *preferential attachment* where a node can choose the label of the node it wants to link to and nodes with higher out-degrees have better chance of being linked to. One could easily extend the data-generator to implement more complex types of preferential attachment. This sort of preferential attachment can be used to control the size of *closed loops* in the generated graph as we will describe later. We implemented *preferential attachment* using the function $chooseClass(c, numLabels)$ which returns a class label given a label c and the set of labels $\{0, \ldots numLabels - 1\}$, the returned class label is the label which the input label c prefers. This function is called whenever we have a node v and want to find another node w to link v to.

After generating the graph, we assign attribute values to each node as described in Algorithm 8. Figure 1 shows how different class labels sample different word-ids according to the binomial distribution in Algorithm 8 if we set $vocabSize = 10$ and $numLabels = 5$. Simply sampling words from the binomial distributions has a considerable amount of discriminative power. The *then* condition in the *if* statement in Algorithm 8 introduces some random noise to this attribute sampling procedure and the degree of randomness can be controlled by changing the parameter $attrNoise$. In some of our experiments we examine the effect of attribute noise in the word attributes on the various collective classification algorithms.

Table 3 describes our parameter settings. For each of our experiments we produced disjoint pairs of training and testing datasets.

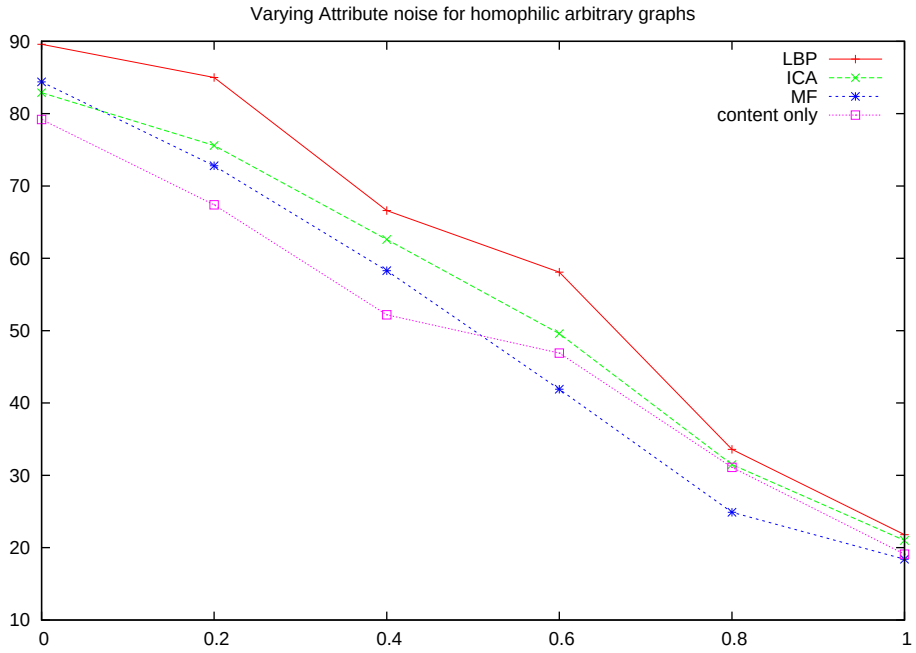


Figure 2: Classification accuracies obtained by varying $attrNoise$ for homophilic synthetic data ($\alpha = 0.2$)

6.3 Experiments with synthetic datasets

6.4 Initial experiments

Our first experiments were on synthetic datasets where nodes with label c link to other nodes with label c. Datasets with such link patterns are said to exhibit *homophily* or *encyclopedia regularity* ([28]). For our first experiment we tried to find the degree of dependance of each ACCA on the noise in the attribute values of the documents in the datasets. In this experiment, we generated synthetic datasets by varying the $attrNoise$ parameter but kept the number of links (controlled by the parameter α) constant. Intuitively, when the noise in the attribute values is low, all ACCAs should be able to classify the documents in the test set simply by looking at the attribute values. As we increase the noise in the attribute values the ACCAs should begin to exploit the link patterns in the dataset and produce results better than what the content only classifier returns.

For our experimental setup, we produced training datasets with $attrNoise = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0$ and keeping α constant at 0.2. For each setting of $attrNoise$ we also generated a test dataset. We trained the four classifiers on each of the training dataset and tested them on the corresponding test set. The classification accuracies are shown in Figure 2.

Note that in the case of the synthetic datasets produced for this experiment all runs of LBP and MF converged in a very small number of iterations (≤ 10). ICA also converged in a very small number of iterations (≤ 5). The results in

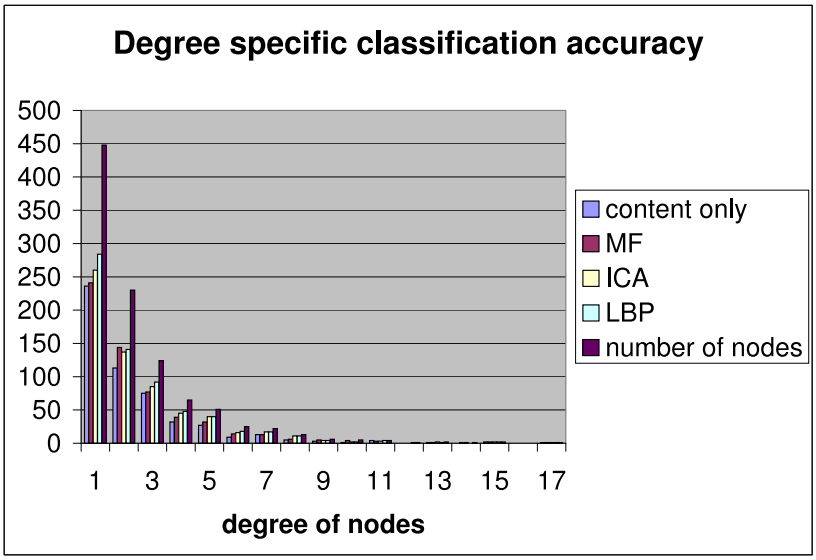


Figure 3: Degree-specific classification accuracies for the homophilic dataset generated with $attrNoise = 0.4$ and $\alpha = 0.2$. The last column in each block shows the number of nodes with a specific degree.

Figure 2 show that the three ACCAs return better results than the content only classifier when $attrNoise$ is increased from 0 to 0.4 by exploiting the correlations in the links. Even beyond $attrNoise = 0.4$, ICA and LBP manage to stay above the content only classifier. The most surprising observation for this experiment was the poor performance of MF which not only returns the lowest classification accuracies amongst the three ACCAs but returns lower classification accuracies than the content only classifier for $attrNoise = 0.6, 0.8, 1.0$. To investigate the poor performance of MF we decided to take a closer look at the classification accuracies. We divided all the nodes in the graph into sets containing nodes of equal degrees (number of neighbouring documents). Figure 3 shows one such *degree-specific classification accuracy* graph for the synthetic test dataset generated with $attrNoise = 0.4$ and $\alpha = 0.2$.

The last column in each block of Figure 3 shows the number of document nodes in each set of nodes. Figure 3 shows that although MF misclassifies more documents (compared to ICA and LBP) irrespective of its degrees, the most important contributor, by far, to the overall classification accuracy is the number of document nodes with one neighbouring document node. Recall that MF is a variational method which approximately minimizes the Mean Field Energy. A different variational method, LBP, does way better in terms of classification accuracy than MF, in fact, LBP, performs the best amongst all the classifiers. The main difference between LBP's objective function and MF's objective function is the presence of higher order terms to estimate the marginal probabilities of edges in the graph ($b_{i,j}(y_i, y_j)$). Somehow, this difference in the way the two

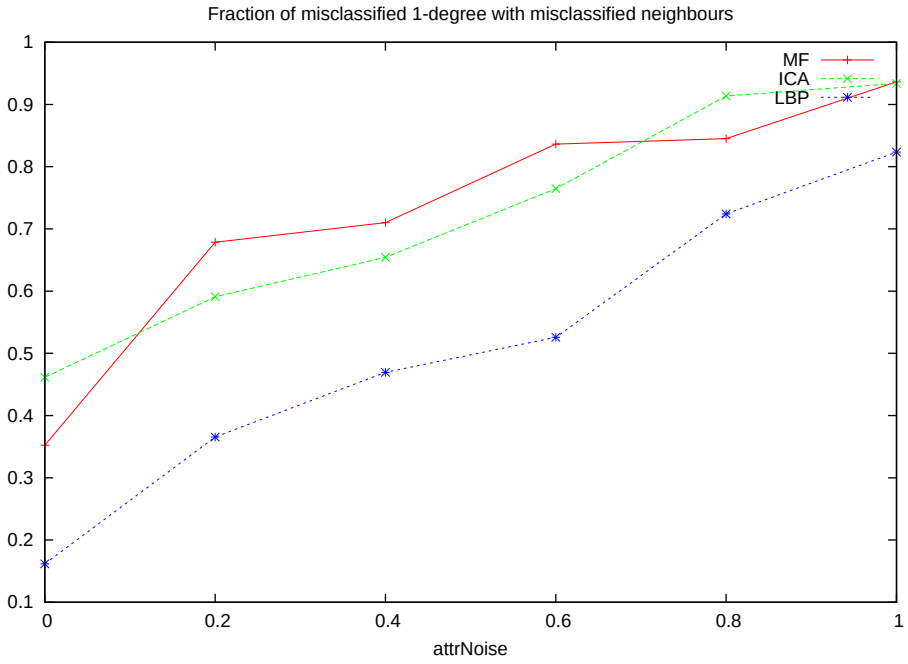


Figure 4: Fraction of misclassified 1-degree nodes with misclassified neighbours at $\alpha = 0.2$.

ACCAs handle links between two documents helps LBP achieve far more accurate results than MF. To investigate if this was indeed the case we computed the fraction of 1-degree documents with misclassified neighbours in Figure 4.

$$\frac{\#\text{ of mislcassified degree 1 nodes with a misclassified neighbour}}{\#\text{ of mislcassified degree 1 nodes}}$$

Note the wide gap between LBP's plot and MF's plot in Figure 4. Figure 4 suggests that LBP handles links (atleast in the case of documents with one document neighbour) far better than MF.

6.4.1 Cause for poor performance of MF

Previous comparisons between MF and LBP ([26]) has shown that even when the graph is singly connected and devoid of loops MF tends to gets stuck in local minima. Moreover, beliefs returned by MF tend to be "overconfident" or more extreme than the true beliefs. Our experiments also point to the fact that MF tends to get stuck in local minima and return suboptimal results than ICA and LBP. Moreover, MF's tendency to produce overconfident probabilities seems to make it more likely to propagate mistakes through neighbouring documents (as shown by the evidence in Figure 4).

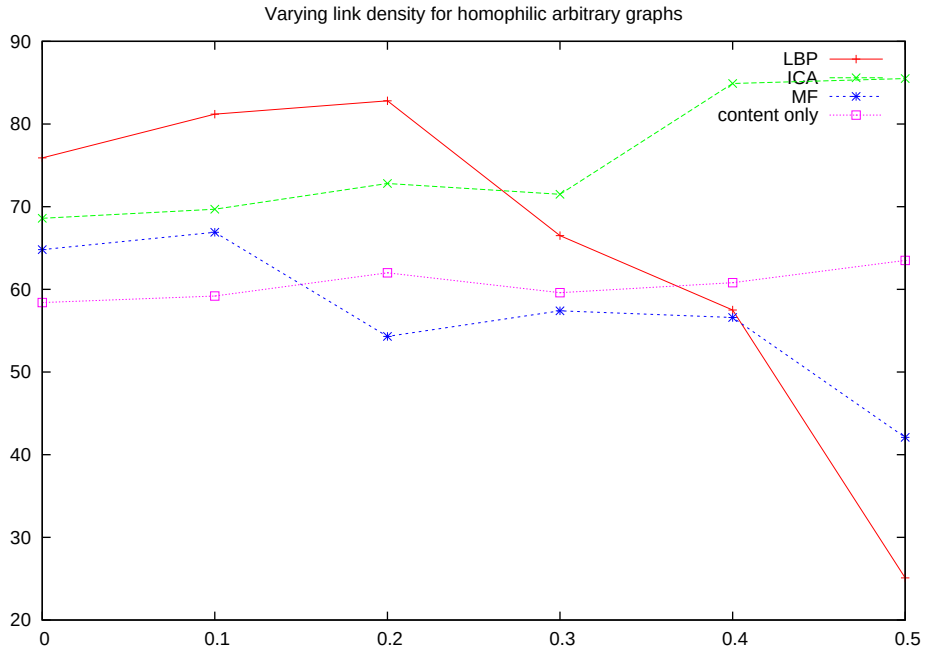


Figure 5: Varying link density for homophilic graphs ($attrNoise = 0.3$)

6.4.2 Effect of varying graph characteristics

When dealing with irregular graphs, one can expect to see a wide variety of graphs with different characteristics. We tried to find out if MF's poor performance is affected by varying these characteristics. One such characteristic is varying the *link density* of the graph. Increasing link density will introduce more *closed loops* in the graph making it more difficult for message-passing ACCAs like LBP and MF to converge. However, we were more interested in determining that even in the cases that MF and LBP do converge, does increasing link density increase their chances of getting stuck in a local minima?

We produced training datasets with $\alpha = 0.0, 0.1, 0.2, 0.3, 0.4, 0.5$ and keeping $attrNoise$ constant at 0.3. For each setting of α we also generated a test dataset. We trained the four classifiers on each of the training dataset and tested them on the corresponding test set. The classification accuracies are shown in Figure 5.

Note that in the case of the synthetic datasets produced for this experiment all runs of LBP and MF converged in a very small number of iterations ($\leq$ 10). ICA also converged in a very small number of iterations ($\leq$ 5). Figure 5 shows that MF tends to produce deteriorating results as α increases showing an increasing tendency to get stuck in a local minima. Even LBP shows signs of getting stuck in a local minima once α increases beyond $\alpha = 0.2$. ICA seems to be quite robust in this respect showing improving results throughout, especially beyond $\alpha = 0.3$.

Figure 5 shows that in the case of LBP, convergence does **not** guarantee good results. Previous investigations ([15]) reported that, usually, when LBP

converges it converges to very good approximations of the marginal probabilities. In our experiment, for the test dataset generated with $\alpha = 0.5$, LBP converged in 5 iterations but produced far worse results than ICA. Recent research ([29]) has thrown more light on the case when LBP converges but to wrong approximations of the marginal probabilities. Yedidia et al. [29] lists some desirable qualities of the *Bethe Free Energy* (*maxent-normal*) which increase chances of convergence to good approximations. When the *Bethe Free Energy* does not possess these qualities it is likely that even when the message passing converges it will converge to wrong values of the marginal probabilities.

6.4.3 Effect of different link patterns

In the last set of experiments we generated synthetic datasets with *homophily*. It is conceivable that homophily is not the only link pattern that we will encounter while applying link-based classification. In the next set of experiments we try to find out if the different ACCAs are affected by the type of link patterns.

Message-passing algorithms like LBP and MF are affected by the number of loops in the graph. When the graph is singly-connected, *Bethe Free Energy* is an exact formulation and so LBP will return exact values of the marginals. Intuitively, when the number of loops in the graph is small then LBP should perform well. Homophily tends to give rise to graphs with very short loops. For example, imagine a document node A of class label l with degree 2, this node is likely to be linked to another document node B of the same class l. A is also likely to be labeled to another node C of the same class label l. Now, since B and C are of the same class label there is a good chance that B is linked to C (homophily) introducing a closed-loop of size 3.

Consider a different link pattern where a document node of label l links to nodes with label $l-1$ and nodes with label $l+1$ (chosen uniformly at random). Imagine a node A of class label 2 with degree 2, this node is likely to be linked to two nodes B and C such that one of the following holds true:

- B belongs to class 1 and C belongs to class 1.
- B belongs to class 3 and C belongs to class 3.
- B belongs to class 1 and C belongs to class 3.

Notice that in none of the cases do B and C have labels which make it probable for them to link up. This link pattern discourages short closed loops like homophily does. In this set of experiments we experiment with synthetic datasets with such a link pattern.

Our first experiment we observed the effect of varying $attrNoise$. We produced training datasets with $attrNoise = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0$ and keeping α constant at 0.2. For each setting of $attrNoise$ we also generated a test dataset. We trained the four classifiers on each of the training dataset and tested them on the corresponding test set. The classification accuracies are shown in Figure 6. Figure 7 shows the degrees-pecific classification accuracies for the test dataset generated with $attrNoise = 0.4$ and $\alpha = 0.2$. Note the improvement in performance of MF for nodes of almost all degrees returning accuracies equal to, if not better than, ICA.

Figure 6 shows an improvement in the performance of MF. MF returns results comparable to ICA's in almost all the settings.

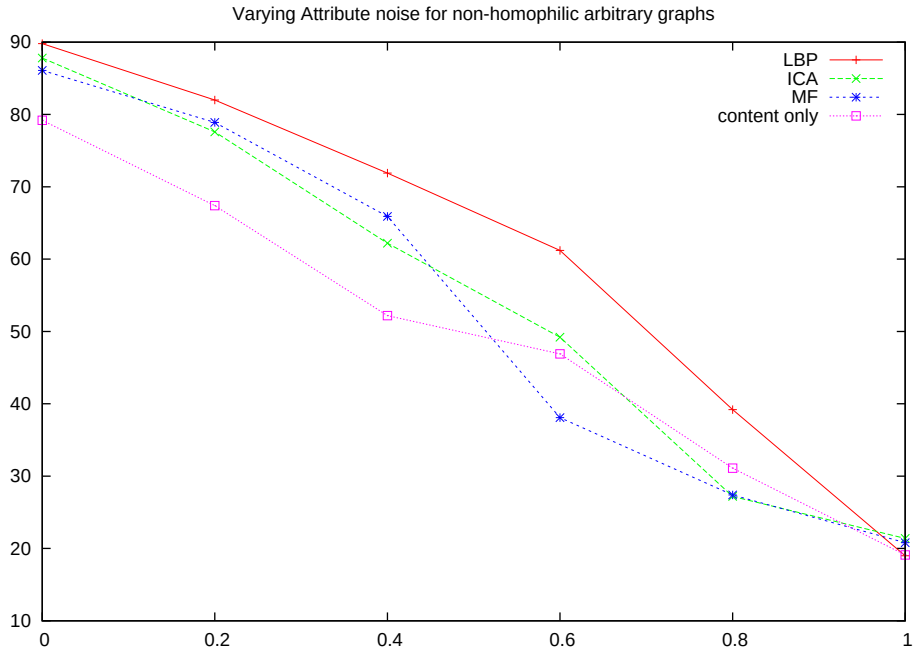


Figure 6: Varying $attrNoise$ for synthetic data ($\alpha = 0.2$) where nodes with label l link to nodes with label $l-1$ and $l+1$

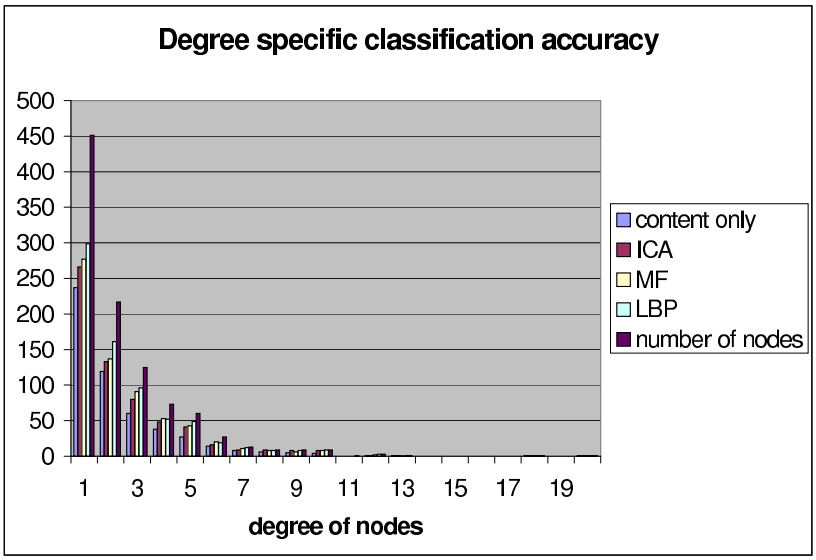


Figure 7: Degree-specific classification accuracies for the dataset with the new link pattern generated with $attrNoise = 0.4$ and $\alpha = 0.2$. The last column in each block shows the number of nodes with a specific degree.

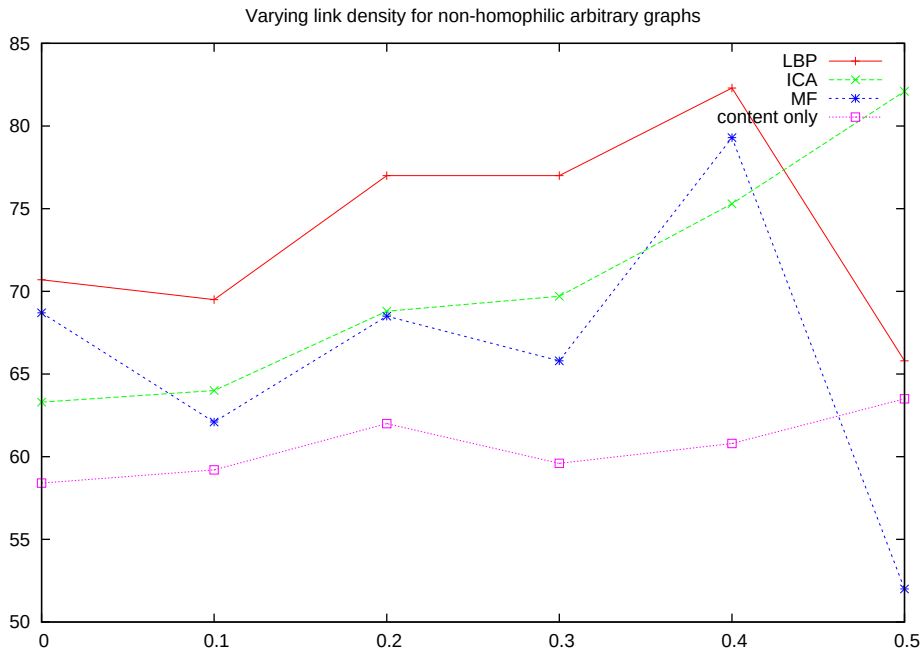


Figure 8: Varying link density for cyclic graphs ($attrNoise = 0.3$)

We also varied the link density in the synthetic datasets with the new link pattern. We produced training datasets with $\alpha = 0.0, 0.1, 0.2, 0.3, 0.4, 0.5$ and keeping $attrNoise$ constant at 0.3. For each setting of α we also generated a test dataset. We trained the four classifiers on each of the training dataset and tested them on the corresponding test set. The classification accuracies are shown in Figure 8.

These results show that the link patterns present in the dataset to be classified has an effect in determining the size and number of loops in the graph. The choice of ACCA should be made depending on the type of link pattern exhibited.

6.5 Experiments with real-world datasets

We tested the different ACCAs on three real-world datasets.

The WebKB dataset is a hypertext dataset collected from various universities. There are four splits (after removing the "other" split), each corresponding to webpages from a university: wisconsin, washington, texas or cornell. There are 5 class labels: *Course, Student, Faculty, Project* and *Staff*. From the "university" splits we selected only those documents which either link to or are linked by atleast one other document in the dataset and extracted a corpus of 877 documents. After stemming and stop word removal we collected a vocabulary of 1703 distinct words. There are 1608 links in our corpus. We did four fold cross validation using the "university" splits training on 3 and testing on the 4^{th}.Note that documents belonging to one university do not link to another university's documents. Thus documents belonging to one university form one

	content only	ICA	MF	LBP
Accuracy (micro averaged)	81.69	84.39	84.40	**84.42**
Precision (macro averaged)	78.07	**79.65**	78.63	78.71
Recall (macro averaged)	60.73	**64.71**	64.45	64.11
F1-measure (macro averaged)	62.49	**65.99**	65.65	65.13

Table 4: WebKB results averaged across 4 university splits

document graph completely disconnected from other university graphs. This has an effect while testing because we can only depend on the words of the documents for evidence since there are no documents with known class labels in the test set.

The other two real-world datasets we experimented on were both bibliographic datasets. The *Cora* dataset contains a number of Machine Learning papers. The papers are labeled from one of *Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning* and *Theory*. We chose documents so that each document is either cited or cites one of the other documents in our corpus. Finally, we divided our corpus into 3 splits. To create the splits we chose documents at random to make the split. This has the effect of creating many "across-split" links which turn out to be extremely useful while testing. Stop words were removed and words with document frequency of less than 10 were also removed. Finally, there were 2708 documents in our corpus with a vocabulary of 1433 words and 5429 links.

The CiteSeer dataset was extracted from the CiteSeer database. We extracted our splits from the CiteSeer corpus in exactly the same way as we did with Cora. Each document in our corpus either cites or gets cited from another document in the corpus. We divided the dataset into three splits by choosing documents randomly from the corpus. Stop words were removed and words with document frequency of less than 10 were also removed. Our final corpus contains 3312 documents with a vocabulary of 3703 distinct words and 4732 links. The documents are labeled with one of the following class labels *Agents, AI, DB, IR, ML* and *HCI*.

The results for the WebKB dataset are shown in Table 4. The numbers indicate that using the link structure helps achieve improvements in classification accuracy. We also provide the degree-specific accuracy distribution results in Appendix A. The degree distribution much more skewed in WebKB than any of the synthetic datasets we generated. In fact, there is a document in the washington split which links to 122 other documents. For WebKB, the content only classifier works reasonably well. We also performed a paired t-test to test the significance of the results. These results are shown in Appendix A.

The results for the experiments on the bibliographic datasets are shown in Table 5 and Table 6. In the case of *Cora*, LBP does much better than the other inference procedures. Even though LBP shows improvements over ICA and MF for Cora, the results are not significant. The results for the significance tests are shown in Appendix A. The degree-specific classification accuracies for the *Cora* splits are shown in Appendix A.

On *CiteSeer*, all three ACCAs show almost equal behaviour improving on the content only classification accuracy. The results of the paired t-test are shown in Appendix A and the degree-spevific classification accuracies are shown in

	content only	ICA	MF	LBP
Accuracy (micro averaged)	70.71	78.35	82.61	**84.49**
Precision (macro averaged)	73.84	84.06	82.27	**84.82**
Recall (macro averaged)	63.88	71.14	78.52	**80.87**
F1-measure (macro averaged)	66.51	74.99	79.70	**82.48**

Table 5: Cora results averaged across 3 splits

	content only	ICA	MF	LBP
Accuracy (micro averaged)	68.56	72.71	72.67	**72.94**
Precision (macro averaged)	61.93	67.88	**68.09**	67.98
Recall (macro averaged)	61.72	**62.46**	61.92	62.18
F1-measure (macro averaged)	59.77	62.46	**62.91**	62.64

Table 6: CiteSeer results averaged across 3 splits

Appendix A.

In both WebKB and CiteSeer, the difference between the average classification accuracies returned by content-only and LBP is around $3 - 4.5\%$ whereas in the case of Cora it is an order of magnitude higher, around 13.5% which seems to suggest that the link structure provides most information in the case of Cora. Only in the case of Cora does LBP show any consistent improvement over the other collective classification methods. Even then the results are not significant which is very disappointing. This hints to the fact that LBP is not sufficient for dealing with the sort of irregular graphs one encounters for link-based classification. One reason for this could be the skewed degree distribution in real world datasets. With such skewed degree distributions, one is likely to find a lot of short closed loops than one would expect. Given its simplicity, ICA performs very well producing results very close to LBP in the case of WebKB and CiteSeer.

Table 7 compares the times required by our implementations of the various ACCAs. Training with LBP requires much larger durations than training with MF or ICA. Training times with MF and ICA are comparable. Testing with all three ACCAs is very fast.

6.5.1 Problems encountered while training

For both the bibliographic datasets, we faced a lot of problems training the classifiers using LBP and MF. This is slightly disconcerting because, for Maximum-Entropy models, given fully labeled training data, learning is supposed to be a concave function and one should be able to get to the global optimum given any

	content only	ICA	MF	LBP
Training	53.03	62.43	79.59	588.49
Testing	0.08	0.17	0.20	0.23

Table 7: Run times measured in seconds on a Xeon 2.8 GHz dual processor on a 3000 node dataset generated with $\alpha = 0.2$ and $attrNoise = 0.2$

starting point. An essential step in the learning procedure we used is to compute the gradient of the objective function. As we discussed in Section 5, the gradient is computed using the marginals under the current parameter settings. If we could perform exact inference then we wouldn't have faced problems in computing the gradient but the datasets we encountered have tens of thousands of nodes and exact inference is not possible. Instead we employ approximate inference mechanisms like LBP and MF. These approximate inference mechanisms seem to return very poor estimates of the marginals when learning for datasets like Cora and CiteSeer and thus the computed gradient is wrong which causes problems while learning. A good set of initial parameters which are close to the optimum values goes a long way to alleviate this problem. In atleast one case, we had to use the parameters learned by ICA to initialize the learning procedure for LBP. That means that the oft repeated fact that parameter estimation for Maximum Entropy models is a concave function is of little practical importance when dealing with arbitrary graphs for link-based classification using approximate inference mechanisms like LBP and MF and there will be cases when we might need to initialize our parameter values carefully.

7 Related Work

7.1 Prior work on performance comparisons

A number of different ACCAs have been proposed but, to the best of our knowledge, there hasn't been too many head-to-head performance comparisons especially comparing their performance on classification with arbitrary graphs. In contrast there has been a number of performance studies describing the performance of various classifiers ([8, 28]) and usefulness of various features ([11]). Most papers proposing new classifiers for link-based classification also provide a brief comparison with existing methods ([18, 19, 23]).

Murphy et al. [15] provided a study describing the performance of LBP on a few well known networks (eg: QMR-DT). Murphy et al. do not compare LBP against other ACCAs. They found that LBP converged to good approximations of the marginals in all but the most complicated graph they considered. On the QMR-DT network LBP failed to converge. Most of the graphs they experimented on were very small graphs which could be solved for exactly by using the *Junction Tree* algorithm. They attributed the poor performance of LBP on the QMR-DT network to small priors. They also found that when LBP converged it did so to very good approximations. This finding led Murphy et al. to suggest that convergence of LBP could be used as a test to judge whether LBP is the appropriate choice for a problem.

Weiss [26] provides a comparison between MF and LBP on some regular graphs most of which were singly connected. Weiss found that MF is very prone to get stuck at a local minima. Weiss does not provide a study to judge the efficacy of LBP, LBP's performance is used as a benchmark to compare results returned by MF.

None of the above work compare ACCAs on the task of classification with irregular graphs. Our aim was to provide a study which could be used to judge which ACCA is appropriate for the various applications of link-based classification. Moreover, recent research has found that many of the above

findings are not true. For example, it is now known that LBP can converge but to very poor approximations ([29]) thus using convergence as a test to judge the efficacy of LBP is not a good idea.

7.2 Applications of link-based classification

Link-based classification is a very general formulation and has been successfully applied to many application domains. In this section we briefly list the various domains link-based classification has been applied to. A domain which has seen a lot of activity is document classification. Chakrabarti et al. [6] was one of the first to apply link-based classification to a corpora of patents linked via hyperlinks and reported that considering attributes of neighbouring documents actually hurts classification performance. Slattery and Craven [18] also considered the problem of document classification by constructing features from neighbouring documents using an *Inductive Logic Programming* rule learner. Yang et al. [28] conducted an in-depth investigation over mutliple datasets commonly used for document classification experiments and identified different patterns.

Document classification is by no means the only domain for applying the link-based classification formulation. Taskar et al. [24] applies this formulation to **classify hyperlinks** between hypertext documents. One of the ideas used here was that, if two hyperlinks emanate from the same *section* in a webpage, they are likely to be classified with the same label. Taskar et al. also applies link-based classification to the problem of predicting friendship links between people. One of the interesting ideas investigated in this problem is whether a person's friendship link with one person affects the presence/absence of a friendship link with another person. Lafferty et al. [12] apply link-based classification to the *Natural Language Processing* task of **part-of-speech** tagging and proposed *Conditional Random Fields*, a novel classifier for sequence data which is based on maximum entropy principles. Taskar et al. [22] proposed *Maximum Margin Markov Networks* for link-based classification, which is an extension of *Support Vector Machines* and provided a generalization bound relating the error rate on the training set to the error-rate on the test set. Taskar et al. showed experiments on the task of *optical character recognition* and *hypertext classification*. Optical Character Recognition can be posed as a sequence modeling task where the image of each character can be used as features, the true character is the label and the word is the labeling for the complete sequence. In other words, we can use the correlation between various characters occurring next to each other to improve classification. Taskar et al. [20] applied a specific form of link-based classification, a generalized version of the *Potts model*, to the problem of document classification and hypertext classification which allowed them to model the problem as a *Integer Linear Program* which is guaranteed to return integral solutions. Taskar et al. [21] used the link-based classification formulation for link prediction to predict *disulphide bonds* in proteins. Anguelov et al. [3] applies link-based classification to segmentation of 3D scan data. Carvalho and Cohen [5] classifies email "speech acts" (eg: request for something, commitment by sender to perform some task etc.) by exploiting the sequential correlation information that exists in emails belonging to the same thread. Chen et al. [7] uses collective classification techniques to resolve entities detected by a sensor network. On a slightly different note, LBP has been used extensively in *Digital Communication* tasks like iterative decoding of *Turbo* codes and *Low-Density*

Parity-Check codes.

7.3 Extensions to LBP

In this report, we compared three *approximate collective classification algorithms*. All three algorithms began as heuristics but encouraging empirical results directed reasearchers to search for theoretical justifications. Both LBP and Relaxation Labeling have a large body of work devoted to their justification, modifications and extensions. Because of its ability to handle higher order interactions (eg: edge marginals) and as shown by our experiments, LBP is preferable to relaxation labeling and here we will briefly describe the recent work done to extend the basic LBP algorithm.

Yedidia et al. [30] justified LBP as a message passing algorithm whose fixed points are stationary points of the *Bethe Free Energy*. Since Yedidia et al.'s original result, a considerable amount of research has been devoted to improving LBP's convergence and accuracy. A natural way to improve the accuracy of basic LBP is to go beyond the pairwise interactions and define larger regions of interactions because this will make the entropy term in *Bethe Free energy* closer to the correct term. In basic LBP, one needs to consider edges and nodes involving the target random variables. Aji and McEliece [1] extend this idea to regions larger than edges and nodes by proposing the use of *Junction Graphs*. Yedidia et al. also described the *Cluster Variational* method which is another way to include larger regions and proposed the *Generalized Belief Propagation* (GBP) by extending the basic LBP. In the *cluster variational method*, one introduces intersections of regions as new regions with the hope that this will improve accuracy of the estimated marginal probabilities. Yedidia et al. [29] illustrated the connection between Aji and McEliece and Yedidia et al. by proposing *Region Graph* method which subsumes both *Junction Graph* and *cluster variational* methods. Note that given a collection of random variables and a set of interactions amongst different subsets of random variables, we could apply any of the above approaches to create a different graph composed of regions. Given so many approaches to generate different region graphs it would be desirable to devise an algorithm which, given an original set of interactions amongst random variables, computes the optimal set of regions. Welling [27] proposes such an algorithm involving splitting, merging and "dying" of regions.

In this report, we discussed three of the simplest and most popular collective classification methods which are, by no means, the only collective classification algorithms available. One problem with all three methods discussed here is that none of them have any proof of convergence. Recent research has developed provably convergent methods for collective classification. A.L.Yuille [2] describes an iterative method which is a provably convergent alternative LBP. Wainwright et al. [25] develops methods where the graph with cycles is reduced to a number of trees (graphs without cycles). Performing inference in trees is easy because they are devoid of cycles and convergence to optimal solutions is guaranteed. Wainwright et al. develops methods which have good convergence properties by approximating the distribution on the original graph by distributions on trees and performing inference on trees which is guaranteed to converge to the optimal solution if there exists one.

8 Discussion and Future Work

Our experiments showed that the choice of the ACCA used for link-based classification can make a difference in the results obtained and this choice needs to made on the basis of various critera.

Of the three ACCAs we compared, LBP uses higher order edge marginals to approximate the multivariate probability distribution. Of the other two ACCAs we considered, MF uses only node marginals whereas there isn't a clear understanding of what ICA is doing. Even then it is conceivable that ICA does not use edge marginals judging by its simplicity. Thus one might think that LBP's higher order variables will help it achive the best results in all cases. Our experiments showed that this was not the case. LBP, like its simpler cousin MF, has a tendency to get stuck in local minima. This problem of getting stuck in local minima is aggravated by two factors:

- Increasing link density in the graph.
- Decreasing the size of closed loops and increasing the number of short loops.

Moreover, the type of link patterns exhibited by the graph can affect the size and number of closed loops thus affecting the performance of LBP.

MF shows the tendency of getting stuck in a local minima to a higher degree. Moreover, MF shows a tendency to propagate misclassifications through documents which are neighbours which brings down the classification accuracy drastically.

In contrast, ICA can be a good choice when LBP and MF fail. ICA seems to be much more robust that LBP or MF to increasing link density. ICA actually improved classification accuracies with increasing link density. ICA even shows consistently good results with varying link patterns showing robustness towards loops of varying sizes.

Judging by the results of our experiments, it seems a good idea to run LBP when the graph is sparsely linked and shows large loops. If the graph is highly linked then one should adopt ICA as the ACCA of choice. ICA has received very little attention in terms of justifying it theoretically, given the results of our experiments it seems worthwhile to search for justifications of ICA and extending the basic algorithm to use edge marginals and compare its results with LBP.

References

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

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

[3] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmenta-

tion of 3d scan data. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

[4] B. Bollobas, C. Borgs, J. Chayes, and O. Riordan. Directed scale-free graphs. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms.*, 2003.

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

[6] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *International Conference on Management of Data.*, pages 307 – 318, 1998.

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

[8] M. Fisher and R. M. Everson. When are links useful? experiments in text classification. In *Proceedings of the European Conference on IR Research.*, pages 41–56, 2003.

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

[10] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research.*, 2002.

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

[12] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning.*, pages 282 – 289, 2001.

[13] S. Li, H. Wang, and M. Petrou. Relaxation labeling of markov random fields. In *In Proceedings of International Conference Pattern Recognition*, volume 94, pages 488–492, 1994.

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

[15] K. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence.*, pages 467–475, 1999.

[16] J. Neville and D. Jensen. Iterative classification in relational data. In *Proceedings of AAAI.*, 2000.

[17] A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. In *IEEE Transactions on Systems, Man and Cybernetics*, 1976.

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

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

[20] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proceedings of the International Conference on Machine Learning.*, 2004.

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

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

[23] B. Taskar, E. Segal, and D. Koller. Probabilistic clustering in relational data. In *International Joint Conference on Artificial Intelligence.*, 2001.

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

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

[26] Y. Weiss. Comparing the mean field method and belief propagation for approximate inference in mrfs. In *Advanced Mean Field Methods, Saad and Opper (ed), MIT Press*, 2001.

[27] M. Welling. On the choice of regions for generalized belief propagation. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence.*, 2004.

[28] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems.*, 18, 2002.

[29] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. In *IEEE Transactions on Information Theory*, pages 2282–2312, 2005.

[30] J. Yedidia, W.T.Freeman, and Y. Weiss. Generalized belief propagation. In *Neural Information Processing Systems.*, volume 13, pages 689–695, 2000.

Appendix

A Results of the paired t-tests and Degree specific classification accuracies for real world data

	LBP vs. MF	LBP vs. ICA	MF vs. ICA	LBP vs. content only	MF vs. content only	ICA vs. content only
Accuracy	0.047	0.0644	0.086	1.55	**1.70**	**1.71**
Precision	0.083	-1.10	-1.53	0.70	0.43	1.10
Recall	-0.25	-0.65	-0.35	1.42	1.55	**1.65**
F1-measure	-0.30	-0.75	-0.454	1.42	1.60	**1.71**

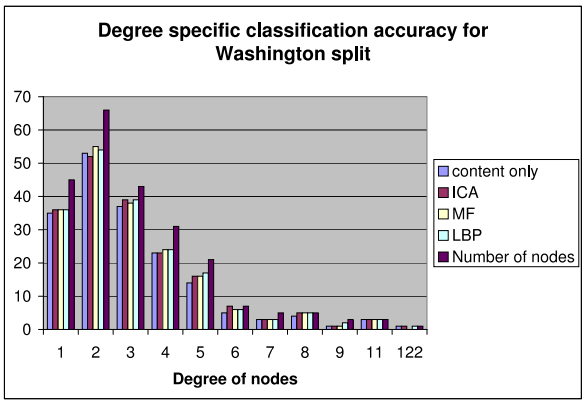
Table 8: Result of paired t-tests on WebKB, numbers in bold font indicate comparisons which are above 90% significance level.

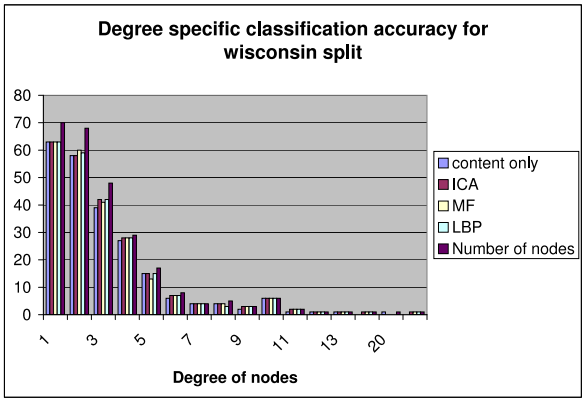
	LBP vs. MF	LBP vs. ICA	MF vs. ICA	LBP vs. content only	MF vs. content only	ICA vs. content only
Accuracy	1.03	1.28	1.19	1.39	1.41	1.28
Precision	1.192	0.82	-1.14	1.39	1.40	1.40
Recall	0.861	1.30	1.36	1.38	1.40	1.25
F1-measure	1.044	1.29	1.30	1.38	1.40	1.31

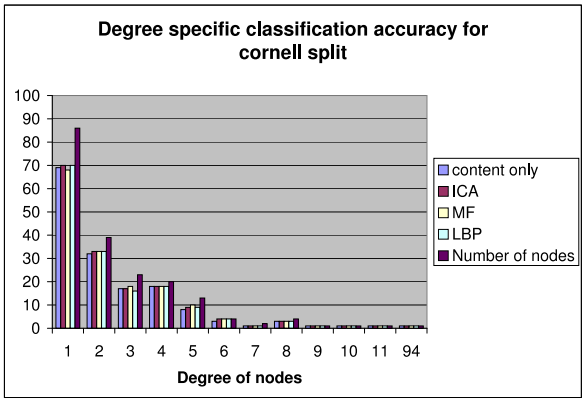
Table 9: Result of paired t-tests on Cora

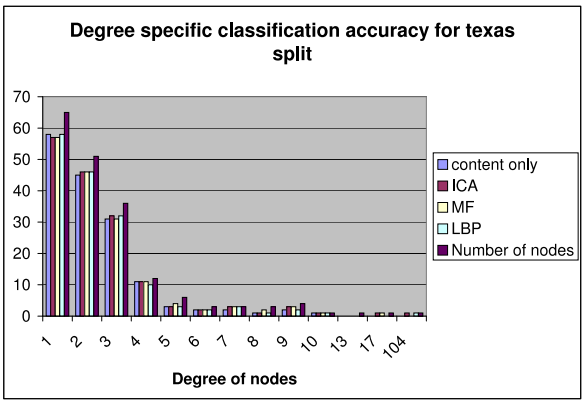
	LBP vs. MF	LBP vs. ICA	MF vs. ICA	LBP vs. content only	MF vs. content only	ICA vs. content only
Accuracy	1.29	0.49	-0.09	1.32	1.30	1.36
Precision	-0.07	0.07	0.12	1.24	1.14	1.32
Recall	0.83	-0.28	-0.43	0.33	0.12	0.64
F1-measure	-1.17	0.17	0.46	1.32	1.35	1.39

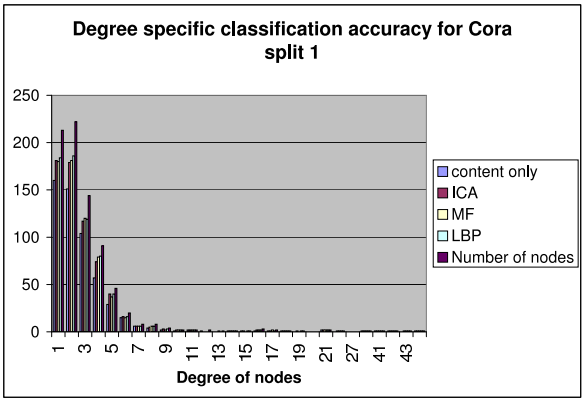
Table 10: Result of paired t-tests on CiteSeer

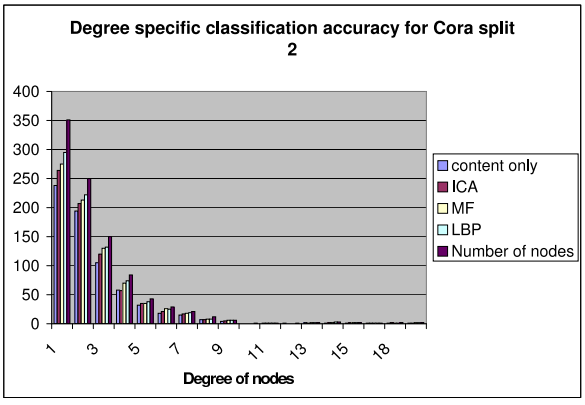

Degree specific classification accuracy for Washington split
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 11 122
Degree of nodes
content only
ICA
MF
LBP
Number of nodes

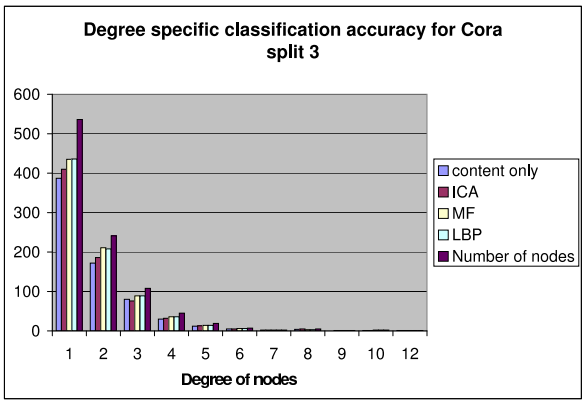

Degree specific classification accuracy for wisconsin split
80
70
60
50
40
30
20
10
0
1 3 5 7 9 11 13 20
Degree of nodes
content only
ICA
MF
LBP
Number of nodes

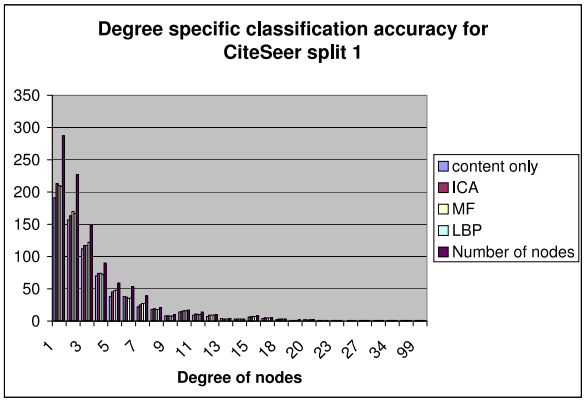

Degree specific classification accuracy for cornell split
100
90
80
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11 94
Degree of nodes
content only
ICA
MF
LBP
Number of nodes

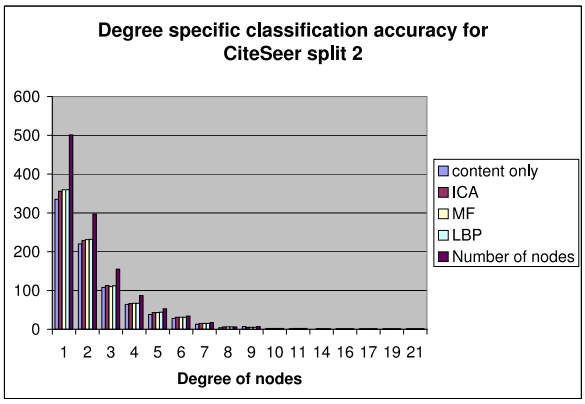

Degree specific classification accuracy for texas split
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
13
17
104
Degree of nodes
content only
ICA
MF
LBP
Number of nodes

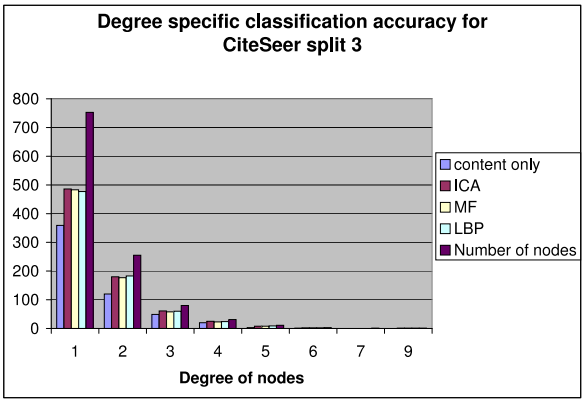

Degree specific classification accuracy for Cora split 1
250
200
150
100
50
0
1
3
5
7
9
11
13
15
17
19
21
27
41
43
Degree of nodes
content only
ICA
MF
LBP
Number of nodes


Degree specific classification accuracy for Cora split 2
400
350
300
250
200
150
100
50
0
1
3
5
7
9
11
13
15
18
Degree of nodes
content only
ICA
MF
LBP
Number of nodes


Degree specific classification accuracy for Cora split 3
600
500
400
300
200
100
0
1 2 3 4 5 6 7 8 9 10 12
Degree of nodes
content only
ICA
MF
LBP
Number of nodes


Degree specific classification accuracy for CiteSeer split 1
350
300
250
200
150
100
50
0
1 3 5 7 9 11 13 15 18 20 23 27 34 99
Degree of nodes
content only
ICA
MF
LBP
Number of nodes


Degree specific classification accuracy for CiteSeer split 2
600
500
400
300
200
100
0
1 2 3 4 5 6 7 8 9 10 11 14 16 17 19 21
Degree of nodes
content only
ICA
MF
LBP
Number of nodes


Degree specific classification accuracy for CiteSeer split 3
800
700
600
500
400
300
200
100
0
1
2
3
4
5
6
7
9
Degree of nodes
content only
ICA
MF
LBP
Number of nodes