Collective Classification Using Heterogeneous Classifiers

Zehra Cataltepe[1], Abdullah Sonmez[1], Kadriye Baglioglu[1], and Ayse Erzan[2]
{cataltepe,sonmezab,baglioglu,erzan}@itu.edu.tr

[1] Istanbul Technical University, Computer Engineering Dept., Maslak, Istanbul, Turkey 34469
[2] Istanbul Technical University, Physics Dept., Maslak, Istanbul, Turkey 34469

Abstract. Collective classification algorithms have been used to improve classification performance when network training data with content, link and label information and test data with content and link information are available. Collective classification algorithms use a base classifier which is trained on training content and link data. The base classifier inputs usually consist of the content vector concatenated with an aggregation vector of neighborhood class information. In this paper, instead of using a single base classifier, we propose using different types of base classifiers for content and link. We then combine the content and link classifier outputs using different classifier combination methods. Our experiments show that using heterogeneous classifiers for link and content classification and combining their outputs gives accuracies as good as collective classification. Our method can also be extended to collective classification scenarios with multiple types of content and link.

1 Introduction

In most pattern recognition applications, the observed and unobserved instances are assumed to be drawn independently from the same distribution. Classification problems are solved using instances' features (content) and labels. Connections/dependencies/relations between instances are not taken into consideration. On the other hand, learning problems with network information, where for each node its features and relations with other nodes are available, become more common in our lives. Examples include social [33], semantic [31], financial [1], communication [8] and gene regulatory [3] networks. Classification of nodes or links in the network, discovery of links or nodes which are not yet observed or identification of essential nodes or links, are some of the research areas on networked data.

While in the traditional classification problems the instances/nodes are usually independent and identically distributed, networked data contain instances which are dependent on each other. Link-based classification takes into consideration the links between the objects in order to improve the classification performance. Attributes of objects and links together can be considered as node features. However, when two linked samples are not yet classified, they require each other's labels to decide on their own label. Collective classification methods have been devised to classify test instances in a network simultaneously, based on each other as well as training data.

Collective classification [6, 15, 27] algorithms aim to classify networked data when the test nodes and their links to other test nodes and training nodes are known. In collec-

tive classification, first a base classifier is trained using both content and link information in training data. Then, using a collective inference method, test nodes are iteratively labeled, based on their content and neighbor information. Especially when there is class autocorrelation among the neighboring nodes in the network, test nodes are able take advantage of their neighbors' class information and collective classification improves classification accuracy [11]. Iterative Classification Algorithm (ICA), Gibbs Sampling and Relaxation Labeling [15, 27] are common methods of collective inference. Collective inference methods have been studied in detail in the works of [15] and [27].

Different choices of base classifiers that are able to use content and neighbors' link information, such as naive Bayes, logistic regression, decision trees, k-nearest neighbors, have been used in the literature [27, 21, 11, 19, 17]. The base classifier takes as input, usually, the content features of the node being classified and relational features, which are usually an aggregation of the class labels of the other linked instances [27, 19, 17]. However, when content and relational features show different characteristics it may not be optimal to have a single classifier to combine all of those features. Also, if there are multiple content types, such as text, images and audio on a web page or link types such as direct or co-citation links on a web page, citation or bibliography links on scientific papers, SMS or call links in call detailed record (CDR) data or family, work, friend links on a social web site, it is hard to input all those features into a single classifier while still obtaining a good generalization performance. Certain content or link types may be better suited to identify certain classes and putting all of them into a single feature vector would harm that discrimination ability. Moreover, for certain types of content, certain choices of local classifier may be known to perform better, for example SVMs have been known to perform well for text categorization [12], while hidden Markov models are used for speech [24], and therefore it could be better to use different choices of classifiers for each content type.

In this paper, we investigate different methodologies of base classifier construction for collective classification. First of all, in order to be able to handle multiple content or link types or content types that require a certain classifier, we suggest that a different classifier is trained for each content type and link type. Then, we investigate different methods of combining these classifiers, namely taking the average, maximum and locally weighted averaging, in order to get good training and test accuracies. We show that, classifier combination increases classification accuracy and can achieve results as good as ICA. We investigate the performance of our algorithms and the performance of content only, link only classification and the traditional ICA algorithm which uses content appended by the aggregated neighbor labels. For performance comparison, we use both real and synthetic datasets.

The rest of the paper is organized as follows. In Section 2, we introduce the notation used in the paper and also the collective classification algorithms in general and ICA (Iterative Classification Algorithm) algorithm in particular. Section 3 gives details on the classification and classifier combination methods we introduce in this paper. Related work is given in Section 4. Section 5 describes the real and synthetic datasets used in the experiments and their properties and experimental setup. The results of the experiments are given in Section 6. The discussions are in Section 7.

2 Background

2.1 Notation

Before we give details of the algorithms we use for classification of networked data, in this section, we first give the notation.

We assume that we are given a networked dataset represented by a graph $G = (V, E)$ with nodes (vertices) V and undirected and unit weight links (edges) $L = \{u, v\}, u, v \in V$.

We assume that there is a classification problem with C classes. Each node u has a C dimensional label vector $\mathbf{r}(u) \in \{0,1\}^C$ which uses 1-of-K representation and shows the class of the node. Some of the vertices are in the training set V_{train} whose labels are known, while the rest are in the test set V_{test} whose labels will be predicted. Note that, $V_{train} \cap V_{test} = \emptyset$ and $V_{train} \cup V_{test} = V$. $L_{train} \subset L$ contains the links which are between two training nodes, while L_{test} contains links between test nodes as well as links between training and test nodes.

Each node $u \in V$ (whether it is in the training or test set) also has an m dimensional feature vector $\mathbf{x}(u) \in \{0,1\}^m$.

In traditional machine learning, the classification problem to be solved would be: Given the independent and identically distributed feature vectors of the training nodes and their labels, $\mathbf{x}(u)$ and $\mathbf{r}(u)$, $u \in V_{train}$, find a mapping $g(\mathbf{x}(u)) : \{0,1\}^m \rightarrow \{0,1\}^C$, which best approximates the relationship between feature vectors and labels so that the expected accuracy (acc) of g on any unseen test node v is maximized. It is assumed that both the inputs for training data and the test data come from the same distribution and they are independent. Since the classifier $g(\mathbf{x}(u))$ uses only the input features, we will call it the content only classifier $g(\mathbf{x}(u)) = g_{CO}(\mathbf{x}(u))$.

On the other hand, in transductive learning [32], we assume that we are given a specific test set and our goal is to perform as well as possible on that specific test set and not necessarily on all possible test inputs. The goal of transductive learning can be stated as the maximization of the test classification accuracy:

$$acc(g, V_{test}) = \frac{1}{|V_{test}|} \sum_{v \in V_{test}} 1 - \delta[g_{CO}(\mathbf{x}(v)), \mathbf{r}(v)]. \quad (1)$$

Here $\delta[p, q]$ returns 1 if two vectors p and q differ in at least one position. Transductive learning algorithms try to find the test output assignments that minimize the accuracy on the test set, which is a simpler problem than computing a target function that minimizes the expected test accuracy for all possible test inputs [32]. Note that, in transductive learning, we do not need a classifier g_{CO} but only the estimated labels for the test set.

When not only the training node features, but also links between them are given, the link information can be used for classification. Usually link information of not the whole graph but only the neighbors of a specific node are taken into account, therefore we need to define the concept of a neighborhood. Let $SP_G(u, v)$ denote the number of edges (hops) on the shortest path between two nodes u and $v \in V$, and assign $SP_G(u, u) = 0$ and if u and v are not connected, then $SP_G(u, v) = \infty$. For each node $u \in V$, the h-neighborhood function $N_h(u)$ returns a set of nodes which according to

the links L are neighbors of the node u that are at most h hops away from u:

$$N_h(u) = \{v : SP_G(u, v) \leq h\}. \tag{2}$$

We use the shortened notation of $N(u)$ when $h = 1$ or is specified in advance. The neighborhood function returns a set of nodes which may be empty. We also define a label aggregation function that takes the set of labels of neighbors of the node and returns a C dimensional real vector. Among different aggregation functions available, in this paper, we use the count aggregation and define the aggregated labels of a node u's neighbors as:

$$\mathbf{r}_{N_h}(u) = \sum_{v \in N_h(u)} \mathbf{r}(v). \tag{3}$$

Based on the labels of the neighbors only, a classifier, which we call the link only (LO) classifier $g_{LO}(\mathbf{r}_{N_h}(u))$ can be trained on the training data. Since the training nodes' labels and links are given, the aggregated labels can be computed for the whole training data. On the other hand, when the test nodes need to be classified, the actual test labels are not known. Collective classification algorithms such as Iterative Classification Algorithm (ICA) or Gibbs Sampling [15, 27] let test nodes iteratively label and relabel each other until a stable solution is achieved.

When both node features and links are known, a classifier that uses both the content features of the node and labels of the neighbors have been used in [27]. We will call this classifier the content and link classifier $g_{CO,LO}([\mathbf{x}(u)\ \mathbf{r}_{N_h}(u)])$ which has $m + C$ features consisting of the m node features and the C dimensional aggregated label vector of the neighbors.

2.2 Collective Classification

In order to determine a node's label, collective classification uses three types of information about it: The node's observed attributes (content), observed attributes of the node's neighbors, observed labels of the node's neighbors [27]. First of all a base classifier using link (relational) and content features is trained on training data. In order to create fixed size feature vector, content features are appended with an aggregation of link features around the node (usually labels of neighbors) and used as input to the base classifier. Test nodes are first labeled using the content information. Then they are relabeled using the base classifier and predicted labels for their neighbors, until labels converge.

Iterative classification algorithm (ICA) is a popular and simple approximate collective inference algorithm [27, 15]. Despite its simplicity, ICA was shown to perform as well as the other algorithms such as Gibbs Sampling [26]. Pseudocode for the ICA algorithm (based on [27]) is given in Algorithm 1. In the pseudo code, $\tilde{\mathbf{r}}(u)$ stands for temporary label assignment of instance u in the test set. $g_{CO,LO}([\mathbf{x}(u)\ \mathbf{r}_{N_h}(u)])$ is the base classifier which is first trained on training nodes and their neighbors from the training set. The base classifier uses the estimated labels of the neighbors if they are test nodes. O is a random ordering of test nodes.

Algorithm 1 $\tilde{\mathbf{r}}(V_{test}) = \mathrm{ICA}(G, V_{train}, V_{test}, g_{CO,LO}())$

for all $u \in V_{test}$ **do**
  Compute $\tilde{\mathbf{r}}_{N_h}(u)$ using only neighbors in V_{train}
  Set $\tilde{\mathbf{r}}(u) \leftarrow g_{CO,LO}([\mathbf{x}(u)\ \tilde{\mathbf{r}}_{N_h}(u)])$
end for
repeat
  Generate ordering O over nodes in V_{test}
  for all $u \in O$ **do**
    Compute $\tilde{\mathbf{r}}_{N_h}(u)$ using current label assignments to nodes in $N_h(u)$
    Set $\tilde{\mathbf{r}}(u) \leftarrow g_{CO,LO}([\mathbf{x}(u)\ \tilde{\mathbf{r}}_{N_h}(u)])$
  end for
until all labels are stabilized or threshold number of iterations

3 Collective Classification Using Heterogeneous Classifiers

Collective classification algorithms use a base classifier which is trained on training content and link data. The base classifier inputs usually consist of the content vector concatenated with an aggregation vector of neighborhood class information. In this paper, we evaluate different methodologies of base classifier construction for collective classification.

These methodologies are summarized in Table 1. In Table 1, $g_{CO}, g_{LO}, g_{CO,LO}$ are classifiers (such as logistic regression, kNN, SVM) which return the estimated class labels for a given input node. In Content Only (CO) classification g_{CO} classifier and in link only (LO) classification g_{LO} classifier are trained only on content and link information respectively. $g_{CO,LO}$ is the classifier frequently used [27] for ICA which takes the content features of the node appended by its aggregated neighbor classes as inputs. $g_{CO,LO}$ is just one of the possibilities of using both content and link information in classification. Especially when content and link features are classified better by different classifiers, it may not make sense to just join the feature vectors and give them as input to a single classifier as in ICA.

We propose that the content (g_{CO}) and link (g_{LO}) classifiers should be first trained on training data. The link only classifier (g_{LO}) uses an aggregation of neighbor labels, which may not be available for the test nodes. Therefore, we perform ICA using g_{LO} only and obtain estimates for the test node labels using the link information. Test node labels can also be directly estimated using the content only classifier g_{CO}. These two estimates can now be combined using different classifier combination [14] techniques.

In this paper we use three different classifier combination methods:

– $wAVE$ (weighted Average) method computes a weighted average of the classifier g_{CO} and g_{LO}'s outputs. Different classifers may perform better for different neighborhood (in link or content) of nodes. In order to be able to take into account classifier's performance for each node separately, we introduce $\alpha_{CO}(u) \in R$ and $\alpha_{LO}(u) \in R$ which are local classifier weights for content and link only classifiers. The weights $\alpha_{LO}(u)$ can be determined locally, based on the correct classification rate of the g_{LO} classifier in the neighborhood of u according to edges in the link graph $G_{LO} = G$. In order to compute $\alpha_{LO}(u)$, we first find nodes which are in the

h-neighborhood of u in graph G_{LO}, $N_{h,G_{LO}}(u)$. (See Equation 2. We introduce G_{LO} in $N_{h,G_{LO}}(u)$, so that it is clear that the neighborhood is according to the G_{LO} graph.) Then we compute the local average accuracy of the classifier g_{LO} within $N_{h,G_{LO}}(u)$ as:

$$\alpha_{LO}(u) = \overline{acc(g_{LO}, u)} = \frac{1}{|N_{h,G_{LO},train}(u)|} \sum_{v \in N_{h,G_{LO},train}(u)} acc(g_{LO}, v). \quad (4)$$

Here $N_{h,G_{LO},train}(u)$ denotes nodes from training data which are in h-neighborhood of node u in G_{LO} graph.

Similarly, in order to determine $\alpha_{CO}(u)$, we first create a graph G_{CO} based on the content similarities of the nodes. We use cosine similarity and match similarity in this work, however other similarity measures could also be used. In the content graph, we join nodes whose similarity are above a threshold whose value is chosen so that the average degree of the content graph G_{CO} is as close as possible to the average degree of the link only graph G. Once the content graph is produced, $\alpha_{CO}(u) = \overline{acc(g_{CO}, u)}$ is produced similar to Equation 4, but using G_{CO} and g_{CO} instead of their LO counterparts.

If a test node is too far away from training data in G there may not be any training nodes within its h neighborhood. The same could happen for a test node too far away from training data in the content graph G_{CO}. When that is the case, instead of the local weights $\alpha_{CO}(u)$ or $\alpha_{LO}(u)$, their averages over the whole G_{CO} and G graphs are used.

In order to compute the $\alpha_{CO}(u)$ and $\alpha_{LO}(u)$ estimates, we need to determine the values the number of hops, h^*_{CO} and h^*_{LO}, that will be examined in $\alpha_{CO}(u)$ and $\alpha_{LO}(u)$ computations. We can determine these values based on the training data as follows: We compute the correlation (Pearson Correlation Coefficient, to be precise) between accuracy of a classifier on a node and also the local average accuracy of the classifier within the h neighborhood of the node. We choose the number of hops to be examined as the number of hops which maximizes this correlation.

- AVE (Average) method returns the average of the outputs of the g_{CO} and g_{LO} classifiers on a node.
- MAX (Maximum) method takes the maximum of the class probabilities produced by g_{CO} and g_{LO} for each class to be the probability of that class. The class which has the maximum among these combined estimates is chosen as the label for a node.

Table 1: Base Classifier Construction Methods

Method	Abbreviation	Formula
Content Only	CO	$g_{CO}(\mathbf{x}(u))$
Link Only	LO	$g_{LO}(\mathbf{r}_{N_h(u)}(u))$
Iterative Classification Alg.	ICA	$g([\mathbf{x}(u)\ \mathbf{r}_{N_h(u)}(u)])$
Average	AVE	$AVE(g_{CO}(\mathbf{x}(u)), g_{LO}(A(N(u))))$
Maximum	MAX	$MAX(g_{CO}(x(u)), g_{LO}(A(N(u))))$
Local Average	wAVE	$AVE(\alpha_{CO}(u)g_{CO}(\mathbf{x}(u)), \alpha_{LO}(u)(\mathbf{r}_{N_h(u)}(u)))$

4 Related Work

There have been studies, mostly using relational classifiers, that combine a number of classifiers for collective classification.

A local (content) classifier together with an ensemble of relational classifiers have been used by [23]. The aim of [23] is to produce a generic relational ensemble model that can incorporate both relational and local attributes for learning. They address issues related to heterogeneity, sparsity and multiple relations. They introduce a new method called PRNMultiHop which tries to handle the sparsity problem. Instead of considering only the directly linked nodes, they consider two nodes as linked if they can be reached at most in a certain threshold number of hops. They compare their results with RBC, RPT and RDN. On Cora and CompuScience datasets they show that their method PRN2MultiHop outperforms those three methods. Ensemble classification works as follows: They train a base classifier using the local features and a relational classifier for each type of relational feature (link type), then they combine the results of these classifiers using stacking or voting. Stacking gives better results than voting.

Local methods of classifier evaluation have been used to improve collective classification in a number of studies. [2] points out the fact that label autocorrelation may be different in different regions of the graph. They compute the global and node neighborhood autocorrelation using Pearson's corrected contingency coefficient. They also compute the probability of each label (output) for a node given the neighborhood of a node globally on the whole graph and locally around the specific node for which a label will be computed. They use a linear combination of these two probabilities as the label probabilities. In order to compute the weight of the local model, they use the number of labeled neighbors of the node. [16] utilize local and global relevance of a node in order to identify functionally important nodes. For global relevance they use the distribution of the shortest path lengths averaged over destination persons from a source person. Clustering coefficient of a node is used to compute its local relevance.

Stacked graphical models [9, 13] which use a different method of base classifier construction is also related to our work. In stacked graphical models, first a base classifier is trained using content features, then the content features are appended with relational features which are produced using class estimates for the related instances using the learned model. In [9] it is shown that stacked graphical models perform better than traditional collective classification local models, because instead of using aggregation of actual labels they use predicted labels and have smaller bias.

McDowell and colleagues' work on ICA with meta classifier (ICA_{MC})[18] is also related to our work. In (ICA_{MC}), first a single (node) classifier is trained for collective classification. Then meta-features that try to capture the classifier performance for each node are produced and another meta-classifier is trained on these features. Using feature selection on meta-features, [18] obtains better accuracies than ICA.

5 Experimental Setup

5.1 Datasets

In this section, we give details on the CoRA and CiteSeer scientific publication datasets and synthetic datasets which are used in the experiments below.

Cora and Citeseer Datasets The Cora and Citeseer datasets have been used in collective classification literature [27]. These datasets are downloaded from the Statistical Relational Learning Group web site (http://www.cs.umd.edu/projects/linqs/projects/lbc) at the University of Maryland. Both datasets consist of information on scientific papers. As features, the words that occur at least 10 times are used. For each paper, whether or not it contains a specific word, which class it belongs to, which papers it cites and which papers it is cited by are known.

Synthetic Datasets In order to create synthetic networked data for collective classification, we propose a method that allows varying content and link relevances with the class label and varying dependence (redundancy) between content and link. As in the "content based" networks of [4], we generate content and link bits, and based on their link similarity we connect the nodes in the network.

We assume that the data generated consist of a graph $G = (V, E)$ with nodes V and vertices E. For any node in V, class label is assigned based on the complete knowledge of a vector of length m with elements from a certain set $U = \{1, 2, \ldots, C\}$ where $C \geq 2$ is the number of classes. For a certain node $u \in V$, the complete m-element feature vector is denoted by $z(u)$ and its ith element is denoted by $z(u, i)$. Based on the complete feature vector each node is assigned to one of C classes according to the mode of the complete feature vector, i.e. $r(u) = k$ where $k = mode\ z(u, i)$, ties broken randomly if there are multiple modes.

A number of elements, $0 \leq m_c, m_l, m_s \leq m$ are designated for content, link and shared (between content and link) features respectively and these satisfy $m_c + m_l - m_s = m$. Content features $x(u)$ are the first m_c elements of $z(u)$. Link features are determined as $l(u) = [z(u, m_c)z(u, m_c + 1), \ldots, z(u, m - 1)]$. (Please see Figure 1.) The lengths of content and link feature vectors determine their relevance. Note that by using different portions of $z(u)$ for different content and link features, the synthetic data generation algorithm can be extended to multiple views.

Content features $x(u)$ are produced for each node u. In order to produce links between nodes, a similarity measure between their link features is needed. In this paper an integer power of inverse normalized hamming distance is used. For any two nodes $u, v \in V$, their link feature similarity is defined as:

$$sim_l(u, v) = \left(1 - \frac{\sum_{i=1}^{m_l} [l(u, i) = l(v, i)]}{m_l}\right)^a \tag{5}$$

where $a > 0$ is an integer which is used to control the degree distribution of the graph produced, and $[TRUE] = 1$ and $[FALSE] = 0$.

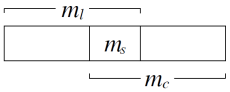

Fig. 1: Partitioning of the elements of the complete feature vector for synthetic data generation.

In order to create a networked dataset with N nodes we use Algorithm 2. In the $RandomizeElements()$ algorithm, each bit is chosen as 1 with probability P_{bias} and 0 with probability $1 - P_{bias}$. Different values of P_{bias} can be used to control the difficulty of the classification problem.

Algorithm 2 $G = \text{Build}(N)$

$V \leftarrow \{\}$
for $i \leftarrow 1 \ldots N$ **do**
  $v \leftarrow \text{CreateVertex}()$
  $v \leftarrow \text{RandomizeElements}(v)$
  $v \leftarrow \text{DetermineLabel}(v)$
  $V \leftarrow V \cup \{v\}$
end for
$E \leftarrow \{\}$
for all $(v, w) \in V \times V, v \neq w$ **do**
  if $\text{Random}() < \text{Similarity}(v, w)$ **then**
    $E \leftarrow E \cup \{(v, w)\}$
  end if
end for
$G \leftarrow (V, E)$

For the synthetic datasets used in this paper, elements come from $U = \{1, 2\}$ and hence there are two classes. The number of content and link bits are all chosen to be $m_c = m_l = 32$. For three different synthetic datasets (Synthms0, Synthms16, Synthms32), the number of shared bits are $ms = 0, 16, 32$, resulting in the total number of bits of $m = m_c + m_l - m_s = 64, 48, 32$ respectively. Datasets produced have $N = |V| = 1000$ nodes and $|E| = 3000$ links. The value of P_{bias} used was 0.75.

Table 2 shows the total number of features, nodes, links and classes for each dataset. We also compute different graph properties for both the original $G = G_{LO}$ graph and the constructed G_{CO}graph.

5.2 Sampling

Class distribution based random sampling, which tries to preserve class distribution of the dataset as much as possible, is used during sampling of training, validation and test sets in the experiments.

5.3 Classification Methods

A base classifier which is trained on node features and local connectivity information is needed for collective classification. In this paper, we use logistic regression (LR), Support Vector Machine (SVM), Naive Bayes (NB), Bayes Net (BN), k-Nearest Neighbor (kNN, k=3) for the g_{CO}, g_{LO} and $g_{CO,LO}$ classifiers. For all of the methods Weka implementations with default parameters (unless otherwise noted) have been used.

6 Experimental Results

6.1 Analysis of Average Local Accuracy Values

In Figure 2, we show the correlation between the accuracy $acc(u)$ of a node u and the average accuracy of its neighbors $\alpha(u)$ for the Cora dataset. Since accuracy depends on the classifier used, we show the correlations when logistic regression and Bayes net classifiers are used. For link only classification, the average local accuracy values are more correlated with accuracy for logistic regression classifier. On the other hand, for content only classification, the correlation is higher for the Bayes net classifier. As it can be seen in the figures, usually the correlation between the accuracy of a node and its neighbors decreases as the size of the neighborhood (h in $N_h(u)$) increases. Except, with the content only classifier, the correlation is maximized for a neighborhood of size 2.

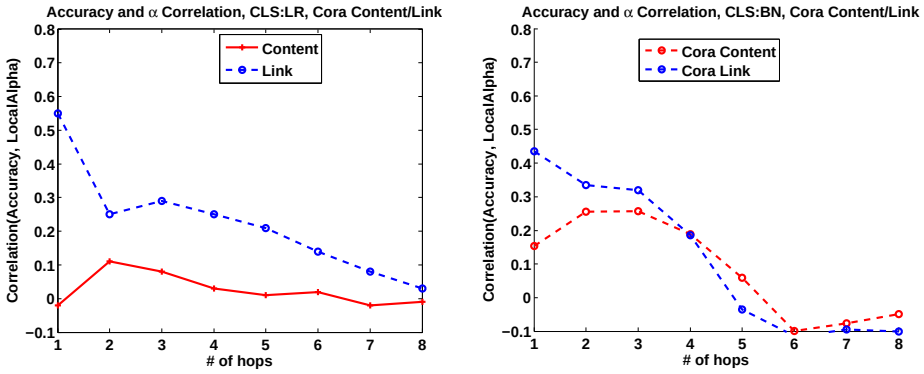

Fig. 2: Correlation between accuracy of the classifier at a node and its accuracy within the h-hop-neighborhood of the node for Cora dataset and using logistic regression classifier.

6.2 Performance of Different Classifiers

First of all, we conducted experiments on the synthetic datasets Synthms0, Synthms16, Synthms32, using Logistic Regression (LR), Support Vector Machine (SVM), Naive Bayes (NB), Bayes Net (BN), k-Nearest Neighbor (kNN) classifiers. Table 3 shows the accuracies obtained when content only (CO), link only (LO) classifiers and ICA

are used with a specific classification method for each dataset. For these experiments, for α_{CO} and α_{LO} computations the number of hops to explore for neighborhood was chosen to be $h^*_{CO} = h^*_{LO} = 2$.

For CO classification, while LR, SVM, BN and NB give similar accuracies, kNN usually gives worse accuracies. On the other hand, for LO all classifiers perform similarly.

As the number of shared bits (m_s) increases the total number of bits m decreases and the information in content and link views increase. For this reason as m_s increases, classification accuracies of both CO and LO methods increase. The LO classifier performs usually worse than the CO classifier, because instead of all m_l bits, it has access to only the aggregated neighbor labels as input. We also see how ICA is affected from the dependency between CO and LO views in Table 3. For the Synthms0 dataset, the number of shared bits is zero and therefore those two views have minimum correlation. Therefore, ICA performs better than both CO and LO classification. Using ICA with LR and SVM still results in some accuracy increase for Synthms16 and no less accuracy than CO or LO for Synthms32. However, with the other classifiers ICA does not necessarily result in an accuracy increase. Therefore, we conclude that, in addition to as recognized by other authors, homophily, correlation between CO and LO views are also an important factor in determining the success of ICA. The less correlated the CO and LO views, the more chances of ICA resulting in better classification accuracy.

Table 4 shows the accuracies obtained when different classifiers are used on Cora and CiteSeer datasets. While for the synthetic datasets the number of input bits m was low and most classifiers performed similarly, this is not the case for the Cora and CiteSeer datasets, whose input dimensionalities are in the thousands (see Table 2). For the CO classification, SVM, NB and BN classifiers usually performed better than the others. We think that this is due to the high input dimensionality of the content features. On the other hand, for LO classification LR outperformed the other methods. Since the LO homophily of Cora dataset is higher than the CiteSeer, the LO accuracies are also higher. For the Cora dataset, instead of using thousands of features in CO classifier, simply using the aggregated class neighbors in LO classifier results in a better accuracy. This is expected since both datasets have high homophily and LO (and ICA) benefits from homophily. For both real datasets, BN method gives the best results for ICA and ICA performs better than CO methods. However, again due to high link graph homophily, ICA performs just a little better than LO accuracies.

6.3 Performance of Classifier Combination

For the synthetic dataset, classification accuracies for the same type of CO and LO classifiers and using the three different classifier combination methods are given in the last three columns of Table 3. All three combination methods result in accuracies at least as good and mostly better than ICA.

For the Cora and Citeseer datasets, some classifiers were shown to be better than the others for CO and LO classification. Therefore, classifier combination results for different types of classifiers and for each classification method are given in Table 5 for the Cora dataset. In the first row of the table, when the best performing classifiers for CO and LO classification, BN and LR respectively, are used, classifier combination

results in as good accuracy as that of ICA. When the LO classifier is changed to BN (second row of the table), classifier combination accuracies are still close to that of ICA. In the last row of the table, classifier combination results when the G_{CO} graph is produced according to match similarity (instead of cosine) are shown. The $wAVE$ method performance is significantly reduced when the content graph is produced using the match similarity, which points out to the fact that the similarity measure used for content graph construction affects the graph produced and neighbors found for each node, therefore content similarity measure is quite important for the performance of the wAVE classifier combination method.

Table 2: Graph Properties for Synthetic Data, Citeseer and Cora

	Synthms0	Synthms16	Synthms32	Cora	Citeseer
Dataset Size	1000	1000	1000	2708	3312
# of Content Features	32	32	32	1433	3703
# of Classes	2	2	2	7	6
# of Links(Link)	2072	2207	2230	5429	4591
# of Links(Content)	2065	2070	2230	31873	21212
Average Degree(Link)	4,14	4,41	4,46	3,898	2,7391
Average Degree(Content)	4,13	4,14	4,46	23,54	12,809
Homophily(Link)	0,732	0,757	0,803	0,8252	0,7099
Homophily(Content)	0,75	0,751	0,803	0,0787	0,2719
Clustering Coefficient(Link)	0,0967	0,1024	0,1136	0,2931	0,2429
Clustering Coefficient(Content)	0,0873	0,0961	0,1136	0,884	0,8805

Table 3: Accuracies Obtained on Synthetic Datasets using Different Classifiers

DataSet-Cls	CO	LO	ICA	AVE	MAX	wAVE
Synthms0-LR	0,88 ± 0,01	0,79 ± 0,02	0,93 ± 0,01	0,93 ± 0,01	0,93 ± 0,01	0,93 ± 0,01
Synthms0-SVM	0,88 ± 0,01	0,80 ± 0,02	0,93 ± 0,01	0,94 ± 0,01	0,94 ± 0,01	0,93 ± 0,01
Synthms0-NB	0,88 ± 0,01	0,78 ± 0,02	0,92 ± 0,01	0,92 ± 0,01	0,92 ± 0,01	0,94 ± 0,01
Synthms0-BN	0,87 ± 0,01	0,78 ± 0,02	0,92 ± 0,01	0,92 ± 0,01	0,92 ± 0,01	0,93 ± 0,01
Synthms0-kNN	0,82 ± 0,01	0,79 ± 0,02	0,84 ± 0,01	0,89 ± 0,01	0,89 ± 0,01	0,89 ± 0,01
Synthms16-LR	0,92 ± 0,01	0,80 ± 0,01	0,94 ± 0,01	0,95 ± 0,01	0,95 ± 0,01	0,94 ± 0,01
Synthms16-SVM	0,92 ± 0,01	0,81 ± 0,01	0,93 ± 0,00	0,95 ± 0,01	0,95 ± 0,01	0,94 ± 0,01
Synthms16-NB	0,93 ± 0,01	0,81 ± 0,01	0,92 ± 0,01	0,92 ± 0,01	0,92 ± 0,01	0,94 ± 0,01
Synthms16-BN	0,93 ± 0,01	0,81 ± 0,01	0,92 ± 0,01	0,92 ± 0,01	0,92 ± 0,01	0,94 ± 0,01
Synthms16-kNN	0,85 ± 0,01	0,81 ± 0,01	0,85 ± 0,01	0,90 ± 0,01	0,89 ± 0,01	0,89 ± 0,01
Synthms32-LR	0,97 ± 0,01	0,87 ± 0,01	0,97 ± 0,01	0,97 ± 0,01	0,97 ± 0,01	0,97 ± 0,01
Synthms32-SVM	0,97 ± 0,01	0,88 ± 0,01	0,97 ± 0,01	0,97 ± 0,01	0,97 ± 0,01	0,97 ± 0,01
Synthms32-NB	0,97 ± 0,01	0,86 ± 0,01	0,94 ± 0,01	0,94 ± 0,01	0,94 ± 0,01	0,95 ± 0,01
Synthms32-BN	0,97 ± 0,01	0,86 ± 0,01	0,93 ± 0,01	0,93 ± 0,01	0,93 ± 0,01	0,95 ± 0,01
Synthms32-kNN	0,90 ± 0,01	0,87 ± 0,01	0,87 ± 0,01	0,89 ± 0,01	0,89 ± 0,01	0,89 ± 0,01

7 Discussion

In this paper, we have shown that for link only or content only classification, based on the characteristics of inputs, different classifiers may perform better than the others for

either link or content. We have also shown that instead of using ICA on a classifier trained with content features appended by link features, simply combining the content only and link only classifiers may result in as good or sometimes better performance.

We experimented with local evaluation of content and link only classifiers and determining how they should be combined. Local average neighbor accuracy is one possible method of classifier assessment. We have seen that the similarity measure used in content graph generation plays an important role in weighted average combination result. Determinining the optimal similarity measure, the one that maximizes the homophily in the content only graph, is an interesting problem that we are planning to investigate in the near future.

Synthetic data experiments have shown that, in addition to homophily, correlation between content and link views also plays an important role in the collective classification performance. Just like classifier combination [14] which benefits from diversity and accuracy of classifiers, both collective classification and combination of content only and link only classifiers also benefit from both accuracy and diversity of content only and link only views. We have seen that when content and link are not correlated, both collective classification and classifier combination gave better results than using link only or content only classifiers. On the other hand, when both content and link views carried the same information, ICA actually resulted in worse accuracies.

Generation of synthetic graphs which exhibit different graph properties (such as degree distribution, clustering coefficient, homophily) using different content generation mechanisms and similarity computations and determining performance of different learning algorithms on them is another important future research direction.

Finally, when there are a number of different types of content or link, training separate classifiers for them and then combining them would allow efficient use of multiple link and content types for classification. Experimenting on real multiview and multilink data is another possible future research direction.

Table 4: Accuracies Obtained on Cora and CiteSeer Datasets Using Different Classifiers

D. No	Acc(CO)	Acc(LO)	Acc(ICA)
Cora-LR	0,63 ± 0,01	0,85 ± 0,01	0,72 ± 0,01
Cora-SVM	0,73 ± 0,01	0,64 ± 0,02	0,76 ± 0,01
Cora-NB	0,73 ± 0,01	0,70 ± 0,02	0,80 ± 0,01
Cora-BN	0,73 ± 0,01	0,78 ± 0,01	0,86 ± 0,01
Cora-KNN	0,50 ± 0,01	0,81 ± 0,01	0,57 ± 0,01
Citeseer-LR	0,58 ± 0,02	0,68 ± 0,02	0,61 ± 0,02
Citeseer-SVM	0,75 ± 0,01	0,66 ± 0,02	0,75 ± 0,01
Citeseer-NB	0,71 ± 0,01	0,59 ± 0,03	0,74 ± 0,01
Citeseer-BN	0,71 ± 0,01	0,65 ± 0,02	0,77 ± 0,01
Citeseer-KNN	0,35 ± 0,02	0,65 ± 0,02	0,37 ± 0,02

Table 5: Cora Dataset Classifier Combination Accuracies for different experiments.

Experiment	CO	LO	ICA	AVE	MAX	wAVE
CO-BN,LO-LR, cos	BN	LR	BN			
$h^*_{CO} = h^*_{LO} = 2$	0,73 ± 0,01	0,78 ± 0,02	0,86 ± 0,02	0,87 ± 0,01	0,86 ± 0,00	0,87 ± 0,01
CO-BN,LO-BN, cos	BN	BN	BN			
$h^*_{CO} = h^*_{LO} = 1$	0,73 ± 0,01	0,78 ± 0,01	0,86 ± 0,01	0,84 ± 0,01	0,83 ± 0,01	0,82 ± 0,01
CO-BN,LO-BN, match	BN	BN	BN			
$h^*_{CO} = h^*_{LO} = 1$	0,73 ± 0,01	0,79 ± 0,01	0,86 ± 0,01	0,85 ± 0,01	0,83 ± 0,01	0,78 ± 0,01

Acknowledgements

Authors Cataltepe, Sonmez and Erzan are supported by Tubitak (The Scientific and Technological Research Foundation of Turkey) research project 109E052. Authors would like to thank Eser Aygün of Istanbul Technical University for providing the java code for the synthetic graph generation.

References

1. A. A. Bernstein, S. Clearwater, S. Hill, C. Perlich, and F. Provost. Discovering knowledge from relational data extracted from business news. In *Proceedings of the Workshop on Multi-Relational Data Mining at KDD-2002*, pages 7–22, 2002.
2. P. Angin and J. Neville. A shrinkage approach for modeling non-stationary relational autocorrelation. In *SNA/KDD*, 2008.
3. A. Awan, H. Bari, F. Yan, S. Moksong, S. Yang, S. Chowdhury, Q. Cui, Z. Yu, E. Purisima, and E. Wang. Regulatory network motifs and hotspots of cancer genes in a mammalian cellular signalling network. *IET Syst. Biol.*, 1(5):292–297, 2007.
4. D. Balcan and A. Erzan. Random model for rna interference yields scale free network. *Eur. Phys. J. B*, (38):253–260, 2004.
5. K. Buza, A. Nanopoulos, and L. Schmidt-Thieme. Graph-based model-selection framework for large ensembles. In *Hybrid Artificial Intelligence Systems, Lecture Notes in Computer Science, Volume 6076/2010*, pages 557–564, 2010.
6. S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD*, 1998.
7. O. Chapelle, A. Zien, and B. Scholkopf. *Semi-supervised learning*. MIT Press, 2006.
8. K. Dasgupta, R. Singh, B. Viswanathan, D. Chakraborty, S. Mukherjea, A. A. Nanavati, and A. Joshi. Social ties and their relevance to churn in mobile telecom networks. In *EDBT08*, 2008.
9. A. Fast and D. Jensen. Why stacked models perform effective collective classification. In *Eighth IEEE International Conference on Data Mining*, pages 785–790, 2008.
10. L. Goodman. Snowball sampling. *Annals of Mathematical Statistics*, 32:148–170, 1961.
11. D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *University of Massachusetts, Technical Report 04-27*, 2004.
12. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML*, 1998.
13. Z. Kou and W. W. Cohen. Notes on stacked graphical learning for efficient inference in markov random fields. In *CMU Technical Report, CMU-ML-07-101*, 2007.

14. L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.
15. S. A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study, May 2007.
16. Y. Maeno and Y. Ohsawa. Node discovery problem for a social network, 2007.
17. L. McDowell, K. Gupta, and D. Aha. Cautious collective classification. *Journal of Machine Learning Research*, 10:2777–2836, 2009.
18. L. McDowell, K. Gupta, and D. Aha. Meta-Prediction for Collective Classification. 2010.
19. L. McDowell, K. M. Gupta, and D. W. Aha. Cautious inference in collective classification. In *AAAI*, pages 596–601. AAAI Press, 2007.
20. J. Neville, B. Gallagher, and T. Eliassi-Rad. Evaluating statistical tests for within-network classifiers of relational data. In *ICDM*, 2009.
21. J. Neville and D. Jensen. Iterative classification in relational data. In *Workshop on Statistical Relational Learning, AAAI*, 2000.
22. A. Popescul and L. H. Ungar. Statistical relational learning for link prediction. In *IJCAI Workshop on Learning Statistical Models from Relational Data, 2003*, 2003.
23. C. Preisach and L. Schmidt-Thieme. Ensembles of relational classifiers. *Knowl. Inf. Syst*, 14(3):249–272, 2008.
24. L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):275–286, 1989.
25. P. Sen and L. Getoor. Empirical comparison of approximate inference algorithms for networked data. In *ICML workshop on Open Problems in Statistical Relational Learning (SRL2006)*, 2006.
26. P. Sen and L. Getoor. Link-based classification. In *UM Computer Science Department, Technical Report, CS-TR-4858*. University of Maryland, 2007.
27. P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3), 2008.
28. B. Senliol, A. Aral, and Z. Cataltepe. Feature selection for collective classification. In *International Symposium on Computer and Information Sciences (ISCIS 2009)*. IEEE, 2009.
29. B. Senliol, Z. Cataltepe, and A. Sonmez. Feature and node selection for collective classification. In *International Symposium on Computer and Information Sciences (ISCIS 2010)*, 2010.
30. U. o. M. Statistical relational learning group.
31. V. Tresp, M. Bundschus, A. Rettinger, and Y. Huang. Towards machine learning on the semantic web. In *Uncertainty Reasoning for the Semantic Web I, Lecture Notes in AI*. Springer, 2008.
32. V. N. Vapnik. *Estimation of dependences based on empirical data*. Birkhuser, 2006.
33. R. Xiang, J. Neville, and M. Rogati. Modeling relationship strength in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 981–990. ACM, 2010.