Improving Relational Classification Using Link Prediction Techniques

Cristina Pérez-Solà[1] and Jordi Herrera-Joancomartí[1,2]

[1] Dept. d'Enginyeria de la Informació i les Comunicacions
Universitat Autònoma de Barcelona
08193 Bellaterra, Catalonia, Spain
{cperez,jherrera}@deic.uab.cat
[2] Internet Interdisciplinary Institute (IN3) - UOC

Abstract. In this paper, we address the problem of classifying entities belonging to networked datasets. We show that assortativity is positively correlated with classification performance and how we are able to improve classification accuracy by increasing the assortativity of the network. Our method to increase assortativity is based on modifying the weights of the edges using a scoring function. We evaluate the ability of different functions to serve for this purpose. Experimental results show that, for the appropriated functions, classification on networks with modified weights outperforms the classification using the original weights.

1 Introduction

Relational classification deals with the problem of classifying networked data, that is, data containing a set of entities that are interlinked with each other. Networked data can be found almost everywhere: from authorship networks, that link authors sharing a common paper, to the now very popular Online Social Networks, where users are mainly linked by friendship. Some traditional machine learning techniques, that deal with independent entities, have been adapted to handle networked datasets, and new algorithms have also been proposed to manage this kind of data. Classification is not an exception. In the last years, many algorithms have been proposed to take advantage of the linked nature of these datasets in order to perform classification [1–4].

In this paper, we build on these existing techniques and propose a method to increase assortativity mixing according to the node class labels. Prior works [5, 6] have suggested that assortativity with respect to class labels is an indicator of the level of performance that a relational classifier is able to achieve. So after proposing a method to increase assortativity, we will evaluate to what extents this statement is true. We will conduct a systematic analysis of the performances obtained when classifying different datasets with multiple configurations of the classifier, and we will show how these performances correlate with the assortativity obtained in both the original graphs and those modified to increase assortativity. Assortativity has been proposed as a metric to perform automatic edge selection [5] because preliminary results showed that choosing those edges

for which higher assortativity was obtained resulted in higher classification performance. However, this preliminary study already showed that the procedure does not always lead to the best possible performance. It is thus interesting to evaluate to what extent assortativity is positively correlated with classification performance.

The contribution of this paper is threefold. First, we propose a method to increase both node and edge assortativity by modifying the weights of the edges. This method is based on the usage of scoring functions. We investigate several scoring functions abilities to increase assortativity for different datasets. Second, we evaluate the correlation between the level of assortativity found in a graph and the obtained performance when trying to classify nodes of that graph. We evaluate correlation for datasets modeling different entities and relationships and for multiple relational classifiers. Third, we compare the classification results of the increased assortativity graphs with the original graphs and analyze the performance improvement.

The rest of the paper is organized as follows. Section 2 describes our proposal for increasing assortativity by modifying the weights of the edges of the graph. Then, Section 3 presents the experimental results supporting our claims. First, Section 3.1 presents the datasets used in the experiments. Then, Section 3.2 demonstrates how the proposed method is able to increase assortativity. In Section 3.3, we define the classification problem that we are facing in order to show, in Section 3.4, how assortativity is positively correlated with classification performance. After that, Section 3.5 demonstrates the effects of using the proposed method on classification performance. Finally, Section 4 reviews the related work and Section 5 presents the conclusions and lines for further work.

2 Modifying Edges' Weight to Increase Assortativity

This section describes the proposed procedure for increasing assortativity. After defining the notation and the concept of assortativity, we present the set of scoring functions that we use to test our technique. Then, we show how to compute the new weights taking into account the results of the scoring functions.

2.1 Notation

Given a graph $G = (V, E_w)$, the set of vertexes V represents the entities in the networked dataset and the set of edges E_w represents the relationships between those entities. Since we are dealing with weighted graphs, edges are pairs of vertexes with an associated weight $e = (v_i, v_j, w_{ij})$ s.t $(v_i, v_j) \in V \times V$ and $w_{ij} \in \mathbb{R}$. Because we are dealing with undirected graphs, symmetry is assumed, $e = (v_i, v_j, w_{ij}) = (v_j, v_i, w_{ji})$. Let us denote by $\Gamma(v_i)$ the set of adjacent nodes of v_i, that is, $\Gamma(v_i) = \{v_j \in V \text{ s.t. } \exists e = (v_i, v_j, w_{ij}) \in E \text{ with } w_{ij} \neq 0\}$. Finally, we will use the words entities, nodes, or vertexes interchangeably through the rest of this paper, as we will do with edges, relationships, and links.

Classification is one of the basic techniques in data mining processes. Classification problems consist on assigning labels to entities for which the label is initially unknown. Given a set of labeled samples, the goal is to assign labels to the rest of the samples in the dataset. More formally, we denote by $\mathfrak{C} = \{\mathfrak{c}_k, \text{ for } k = 1, \cdots, m\}$ the set of all possible categories an entity can be labeled with. Then, there exist a set of nodes $V_l \subset V$ for which the mapping $A : V_l \to \mathfrak{C}$ is known before classification takes place, and a set of nodes $V_{nl} = V \setminus V_l$ for which the mapping is unknown.[1] The goal of the classification process is to discover this latter mapping, or a probability distribution over it. Notice that with this definition, the only uncertainty introduced is the class membership of the nodes in V_{nl}.

2.2 Assortativity

Assortativity mixing is the tendency for entities in a network to be connected to other entities that are like them in some way [7]. This phenomenon has been much studied for social networks, where users show a preference to link, follow, or listen to other users who are like them. When dealing with social networks, assortativity is usually known as homophily. Assortativity (or dissortativity, the tendency of nodes to be linked to other nodes that are not like them) has been reported in many kinds of networks. For instance, degree dissortativity has been observed in protein networks, neural networks, and metabolic networks [7].

Assortativity mixing can be computed according to an enumerative characteristic or a scalar characteristic. In the latter case, degree assortativity is of special interest because of its consequences on the structure of the network. In this paper, we are interested on the first alternative, assortativity according to an enumerative characteristic, where assortativity will be related to the class label of the nodes for which the classification will take place. From now on, we will refer to the assortativity regarding the class labels as merely assortativity.

The first hypothesis that we want to test is if it is possible to increase the assortativity of a graph with respect to the class labels assigned to its nodes without knowing these class labels. That is, given a graph $G = (V, E_w)$ for which all class labels are unknown, we want to see if it is possible to design a process that results in a new graph $G' = (V, E'_w)$ that presents higher assortativity than G. This scenario is even more restrictive than the usual within-network node classification scenario, where some of the labels will be known in advance. Note that although the described process does not need any class label, we make use of these class labels to evaluate its performance (i.e. to compute assortativity).

In order to compute edge assortativity [7] for a given graph $G = (V, E)$ for which the mapping $A : V \to \mathfrak{C}$ is known for all V, an edge assortativity matrix e of size $|\mathfrak{C}| \times |\mathfrak{C}|$ is constructed. Each cell e_{ij} contains the fraction of all edges that link nodes of class $\mathfrak{c}_i$ to nodes of class $\mathfrak{c}_j$, normalized such that $\sum_{\forall i,j} e_{ij} = 1$. Values a_i and b_i are defined as the fraction of each type of end of an edge that

[1] Note that l stands for *labeled* and nl stands for *not labeled*.

is attached to vertexes of type $\mathfrak{c}_i$: $a_i = \sum_{\forall j} e_{ij}$ and $b_i = \sum_{\forall i} e_{ij}$. The (edge) assortativity coefficient A_E is then defined as:

$$A_E = \frac{\sum_{\forall i} e_{ii} - \sum_{\forall i} a_i b_i}{1 - \sum_{\forall i} a_i b_i}$$

Because A_E measures assortativity across edges and not across nodes, a node assortativity metric, A_N, is defined in [5]. A_N is computed in the same way, now using the node assortativity matrix e^* instead of the edge assortativity matrix e. There are also weighted versions of these metrics that take into account not only if there exists an edge between two nodes but also the weight of that edge. Through the rest of the paper, we make use of these weighted versions.

2.3 Scoring Functions

In order to increase both node and edge assortativity in a graph, our proposal is to modify the weights of the edges of the graph, so that the new weight is able to better quantify the strength of the relationship that the edges represent. So we need to find functions that quantify this strength. We make use of functions that receive as input an unweighted unlabeled graph $G = (V, E)$ and return a symmetric score, $s(v_i, v_j) = s(v_j, v_i)$, for every pair of nodes in V, such that it quantifies, somehow, the strength of the relationship between nodes v_j and v_i. Surely, strength is a very general word and, as a consequence, many functions meet the requirements to be used as scoring functions.

The set of scoring functions chosen to test our hypothesis was inspired from those used to solve the link prediction problem in Online Social Networks (OSN). OSN are very dynamic by nature. Over time, new members join the network and new relationships are created both between new and old members. The link prediction problem for OSN consists on inferring which new links are more likely to appear in the future in a network given only its current state [8]. One of the approaches that has been followed to deal with this problem is to define functions that evaluate how likely it is, for a given pair of nodes, to create a new link. After applying these functions to every pair of nodes in the network, the algorithm predicts that those pairs of nodes for which the function returns higher values are the ones who are going to create a new link in the near future. The used functions try to evaluate the proximity or similarity of the nodes, with the idea in mind that two nodes that are proximal are more likely to create a connection in the future than two distant nodes. Depending on which metric is used to define proximity, many link prediction models are created.

The set of metrics that are used to define proximity in the link prediction problem meets all the requirements for our scoring functions. What follows is a short summary of the metrics we have chosen to experiment with.

Number of Common Neighbors (CN): Proximity is usually understood in terms of describing the common neighborhood. The most direct metric to measure the common neighborhood is the number of common neighbors, that is, the cardinal of the intersection between each of the nodes' neighbors sets:

$$score_{CN}(v_i, v_j) = |\Gamma(v_i) \cap \Gamma(v_j)|$$

This measure captures how many neighbors two nodes have in common, but it does not take into account how many non shared neighbors do these nodes have. In order to also include this information, Jaccard Index is defined.

Jaccard Index (JI): JI is defined as the size of the intersection between the two nodes neighborhoods divided by the size of the union of the neighborhoods:

$$score_{JI}(v_i, v_j) = \frac{|\Gamma(v_i) \cap \Gamma(v_j)|}{|\Gamma(v_i) \cup \Gamma(v_j)|}$$

In a similar fashion, we could want to give higher score to nodes that share low degree neighbors. Intuitively, it is more difficult that these low degree nodes have the two evaluated nodes as neighbors than it is for higher degree nodes.

Adamic-Adar (AA): The adaptation to the link prediction model for the Adamic-Adar metric [9] would take into account the degree of the shared neighbors:

$$score_{AA}(v_i, v_j) = \sum_{v_k \in \Gamma(v_i) \cap \Gamma(v_j)} \frac{1}{\log(|\Gamma(v_k)|)}$$

However, other studies point metrics that do not follow this line of thought. Instead of rewarding connections between low degree nodes, some models assume that high degree nodes tend to create more new links.

Preferential Attachment (PA): The preferential attachment model postulates that the probability that a node v_i creates a new link in the network is proportional to the current degree of v_i. Then, the probability that a new link between two nodes is formed depends on the current degrees of these two nodes:

$$score_{PA}(v_i, v_j) = |\Gamma(v_i)||\Gamma(v_j)|$$

Apart from looking at the degree of the neighbors, we can also take into account the density of the common neighbors subgraph.

Clustering Coefficient (CC): The CC of the common neighborhood captures the number of links existing between the common neighbors, taking into account how many of those links could exist:

$$score_{CC}(v_i, v_j) = \frac{2\,|\{e = (v_k, v_l) \in E \text{ s.t. } v_k, v_l \in \Gamma(v_i) \cap \Gamma(v_j)\}|}{|\Gamma(v_i) \cap \Gamma(v_j)|(|\Gamma(v_i) \cap \Gamma(v_j)| - 1)}$$

Note that all the proposed metrics are based on analyzing the common neighborhood that any two nodes may share. Apart from these metrics, other topological measures have been proposed to be used in link prediction. These measures take into account distances between nodes, paths among them, or similarity. A review of some of these metrics can be found in [8].

2.4 Modifying Edges' Weight

Once we have a set of functions evaluating the strength of a relationship, we need to define how to modify the original graph, which already has weights, so that it includes the results of the scoring functions. We propose to modify each weight by directly multiplying it by the result of the scoring function:

$$w'_{ij} = score_{func}(v_i, v_j) * w_{ij}$$

By doing so, we attain two different goals. On one hand, we ensure that no new edges are created. Recall that the scoring function is defined for every pair of nodes of the graph, whether they share a link or not. By multiplying the result of the scoring function by the original weight, we guarantee that all nodes that do not share a link in the original graph (and thus have $w = 0$) will not share a link on the modified graph. On the other hand, we allow all scoring functions to eliminate non-relevant edges by assigning them a score of 0.

3 Experimental Results

This section describes the methodology used to evaluate the proposed techniques as well as the results of the experiments performed in order to do this evaluation.

3.1 Datasets

Table 1. Original datasets

Dataset	$\|\mathfrak{C}\|$	Edge set	$\|V\|$	$\|E\|$
WebKB Cornell	7	Cocitations	351	26832
WebKB Cornell	7	Links	351	1393
WebKB Texas	7	Cocitations	338	32988
WebKB Texas	7	Links	338	1002
WebKB Washington	7	Cocitations	434	30462
WebKB Washington	7	Links	434	1941
WebKB Wisconsin	7	Cocitations	354	33250
WebKB Wisconsin	7	Links	354	1155
IMDb	2	All	1441	48419
IMDb	2	Prodco	1441	20317
Industry	12	Pr	2189	13062
Industry	12	Yh	1798	14165
Cora	7	All	4240	71824
Cora	7	Cite	4240	22516

This paper's experiments are based on several relational datasets which have already been used in the past by the relational learning community. This allows

us to compare our results directly with those found on prior studies while providing a set of diverse graphs coming from different environments to prove our claims.

All the experiments described in this paper are made using essentially 4 different datasets. For each of the datasets, various graphs can be created attending on the kind of relationships taken into account to define the edges or the source of information used to create the graph. This results in a total of 14 different graphs to experiment with. Table 1 presents a short summary of the key properties of each dataset. Note that these datasets are of very different nature and that the differences between graphs constructed using different edges or different datasets are strongly pronounced. The fact that our assumptions hold for most of the presented datasets is thus a good indicator of the soundness of the presented techniques. The original datasets used in this paper can be found in [10] together with a more detailed description of their content.

3.2 Assortativity Measurements

Table 2. Edge assortativity

Graph	Original	AA	CC	CN	JI	PA
$\text{Cornell}_{\text{cocite}}$	0.22701	0.19305	0.21925	0.18095	**0.24969**	0.13277
$\text{Cornell}_{\text{link}}$	0.05404	**0.05860**	**0.09348**	**0.11501**	**0.12689**	−0.25756
$\text{Texas}_{\text{cocite}}$	0.46064	**0.47667**	0.45137	0.45240	**0.61685**	0.29227
$\text{Texas}_{\text{link}}$	−0.03256	**0.25315**	**0.29175**	**0.29279**	**0.50357**	−0.22091
$\text{Washignton}_{\text{cocite}}$	0.30070	0.27731	0.29330	0.25166	**0.36886**	0.19694
$\text{Washington}_{\text{link}}$	0.08401	**0.19725**	0.05016	**0.15769**	**0.43920**	−0.29734
$\text{Wisconsin}_{\text{cocite}}$	0.57683	**0.65363**	**0.58620**	**0.64662**	**0.74448**	0.44479
$\text{Wisconsin}_{\text{link}}$	0.16045	**0.45262**	**0.38430**	**0.50690**	**0.54182**	**0.21701**
IMDb_{all}	0.30519	**0.39482**	**0.33020**	**0.38908**	**0.44831**	0.24412
$\text{IMDb}_{\text{prodco}}$	0.50085	**0.52631**	0.49038	**0.53462**	**0.50723**	**0.52579**
$\text{Industry}_{\text{pr}}$	0.44210	**0.54537**	**0.47248**	**0.54325**	**0.53394**	**0.48832**
$\text{Industry}_{\text{yh}}$	0.44061	**0.47978**	0.41910	**0.45753**	**0.51627**	0.38919
$\text{Cora}_{\text{cite}}$	0.73664	**0.81468**	**0.81058**	**0.80629**	**0.84720**	0.65804
Cora_{all}	0.65627	0.65103	0.64375	0.64624	**0.67744**	0.58648

Table 2 shows the obtained edge assortativity (A_E) values for the original graph as well as for the graphs modified using the scoring functions (in bold type those of which assortativity improves w.r.t. the original graph). The first thing to notice is that original graphs present very different edge assortativity values, and even one of the graphs presents a negative value, although it is close to 0. So we are dealing with graphs that do not show any kind of assortativity nor dissortativity together with graphs that show very high assortativity (for

instance, $\text{cora}_{\text{cite}}$ presents a value of 0.74). When analyzing the success of the different scoring functions in increasing edge assortativity, we can observe that using the Jaccard Index (JI) leads to an increase on A_E for all graphs. Then, there is a set of three graphs ($\text{Cornell}_{\text{cocite}}$, $\text{Washington}_{\text{cocite}}$, and Cora_{all}) for which none of the other scoring functions are able to increase A_E. Apart from Jaccard Index, both the Adamic-Adar metric (AA) and the size of the common neighborhood (CN) are also quite successful, with 11 out of 14 and 10 out of 14 graphs showing an increase on assortativity, respectively. Finally, Clustering Coefficient leads to an increase of A_E on just half of the graphs, while Preferential Attachment is able to do so for only 3 graphs.

The magnitude of the assortativity growth also differs depending on the used scoring function. While JI usually leads to the biggest growth, that is not true for all the cases. For instance, both CN and AA are able to surpass JI for the $\text{IMDB}_{\text{prodco}}$ and $\text{Industry}_{\text{pr}}$ graphs.

Table 3. Node assortativity

Graph	Original	AA	CC	CN	JI	PA
$\text{Cornell}_{\text{cocite}}$	0.15595	**0.17092**	0.15571	**0.16393**	**0.20798**	0.12103
$\text{Cornell}_{\text{link1}}$	0.03999	**0.08155**	0.03542	**0.12417**	**0.12070**	−0.12177
$\text{Texas}_{\text{cocite}}$	0.39393	**0.44062**	0.37213	**0.42317**	**0.55223**	0.28926
$\text{Texas}_{\text{link1}}$	0.04574	**0.24626**	**0.21532**	**0.29777**	**0.48132**	−0.12948
$\text{Washington}_{\text{cocite}}$	0.16165	**0.19945**	0.14190	0.17674	**0.21560**	0.15828
$\text{Washington}_{\text{link}}$	0.02381	**0.13928**	**0.03976**	**0.10134**	**0.36904**	−0.14483
$\text{Wisconsin}_{\text{cocite}}$	0.45537	**0.55342**	**0.46367**	**0.55227**	**0.60544**	0.39855
$\text{Wisconsin}_{\text{link}}$	0.19886	**0.41702**	**0.32907**	**0.47819**	**0.50172**	**0.20973**
IMDb_{all}	0.29626	**0.38699**	**0.32643**	**0.38093**	**0.44384**	0.23210
$\text{IMDb}_{\text{prodco}}$	0.50011	**0.52696**	0.49147	**0.53516**	**0.50827**	**0.52552**
$\text{Industry}_{\text{pr}}$	0.38282	0.38206	0.35325	0.37290	0.38263	0.38222
$\text{Industry}_{\text{yh}}$	0.38541	0.35086	0.37761	0.32570	**0.42881**	0.24248
$\text{Cora}_{\text{cite}}$	0.72968	**0.81079**	**0.81299**	**0.80202**	**0.84906**	0.65219
Cora_{all}	0.64420	0.63709	0.63393	0.63092	**0.67066**	0.55912

Table 3 shows the obtained values for node assortativity (A_N). In this case, there is a graph ($\text{Industry}_{\text{pr}}$) for which none of the modified graphs are able to surpass the original graph assortativity. Nonetheless, A_N does not decrease substantially for any of the modified graphs, so no negative consequences will appear by using the modifications. Leaving aside this graph, results for A_N are similar than those showed for A_E. Graphs modified using JI exhibit higher A_N than the original ones for all datasets, and both the AA and the CN are able to increase A_N for most of the graphs (11 out of 14 and 10 out of 14, respectively). Graphs modified using CC and PA do not show an increase on A_N for most of the graphs.

We have shown that it is possible to increase both edge and node assortativity without knowing the node class labels. Using Jaccard Index as a scoring function results in a general increase on (node and edge) assortativity. The usage of the CN and AA as scoring functions also leads to an increase on assortativity for most of the graphs, although this increase can not be observed for all them. In these cases where assortativity does not increase, it is worth to note that the magnitude of the decrease is small. The graphs modified using CC as scoring function do not show a significant increase in assortativity, so this metric does not seem to be a good alternative to use with general graphs. Lastly, the use of PA as a scoring function must be discarded, as it does not show any improvement over the non-modified graph.

The poor performance of PA in increasing assortativity may be explained by the fact that preferential attachment is a model of network growth, that is, it explains how likely it is for a node to get new links, but, unlike the other scoring functions, it does not quantify the strength of the created link in any manner. On the contrary, the relationships involving very high degree nodes (which get high scores when using PA), will most likely be very weak connections. Note that all the other scoring functions, although they can be used to predict the creation of non existing links, also quantify, in some way, the strength of the relationship between any two nodes.

3.3 Classification Algorithms

We use the Netkit toolkit [5] as the relational classification framework. By using Netkit, we are able to systematically test different classifiers and compare the results. Classifiers in Netkit are comprised by a local classifier (LC), a relational classifier (RL), and a collective inference procedure (CI). Each of the different modules can be instantiated with many components. In our experiments, we allow the LC to be instantiated with either classpriors (`cp`) or uniform (`unif`); the RL component can be instantiated with Weighted-Vote Relational Neighbor Classifier (`wvrn`), its Probabilistic version (`prn`), the Class-distribution Relational Neighbor Classifier[2] (`cdrn-norm-cos`), and Network-Only Bayes Classifier (`no-bayes`); the IC module can be specified with Relaxation Labeling (`relaxLabel`), Iterative Classification (`it`), or without any inference method (`null`).[3] This give us $2 \times 4 \times 3 = 24$ different full classifiers. For the rest of the paper, we will use the term *full classifier* (*fc*) to refer to a specific instantiation of the three modules (LC-RC-CI).

In order to measure classification accuracy or performance of each classifier we use the percentage of (initially unlabeled) nodes in the test set that the classifier is able to correctly classify. Since our datasets contain the class labels for all nodes, we are able to compute this accuracy by taking the labels as the ground of truth.

[2] With Normalized values of neighbor-class and using the cosine distance metric.

[3] Readers can refer to the original Netkit paper [5] for a full explanation of these modules.

3.4 Correlation Between Assortativity and Performance

Once we have shown in Section 3.2 that it is possible to increase assortativity, we have to analyze if this increase in assortativity leads to an increase on classification performance. Intuitively, this is almost tautological for some relational classifiers [5], but the relation is not so obvious for some other classifiers. In order to test our second hypothesis, namely, that assortativity is positively correlated with classification performance, we compute assortativity as in Section 3.2 and classification performances as described in Section 3.3.

We are interested in analyzing the correlation between assortativity and classification performance. We expect that when assortativity increases, classification performance also increases. So we want to discover if the function that describes the relationship between these two variables is monotonically increasing. However, we are not concerned on finding the exact function that describes this relationship.

Spearman's rank correlation coefficient is a measure of statistical dependence between two variables that assesses how well this relationship can be described using a monotonic function [11]. The Spearman's coefficient can take values between -1 and 1, with -1 describing a perfect decreasing monotonic function and 1 characterizing a perfect increasing monotonic function.[4] So we can use the Spearman's rank correlation coefficient to asses whether assortativity is positively correlated with performance.

In the interest of comparing classification performance between different datasets, we use the notion of relative error reduction as defined in [5]:

$$ER_{REL}(fc, D, r) = \frac{base\_error(D) - error(fc, D, r)}{base\_error(D)}$$

The base error for a given dataset D is the error committed when predicting that all samples belong to the most prevalent class. The error for a given dataset D, a full classifier fc, and a labeled ratio r is the error committed when trying to classify the $1 - r\%$ remaining samples with the specific configuration described by fc. Note that although the error reduction metric is not bounded, its value is inside the $[0, 1]$ interval when $base\_error(D) \geq error(fc, D, r)$, which is the most common scenario.

Although classification performance increased with the labeled set ratio (as we will see in Section 3.5), no significant differences where observed on the correlation between performance and assortativity for different r values. Table 4 shows the Spearman's rank correlation coefficient between the node and edge assortativity of each of the graphs and the error reduction achieved when classifying those graphs with the different full classifiers. Results presented on the table correspond to the experiments with r set to 35%. Each of the values represents the correlation between the 84 graphs[5] assortativity values and the 100−run

[4] When data does not contain repeated values.

[5] Notice that the total number of graphs tested comes from the 14 original graphs plus the ones obtained using each of the 5 scoring functions.

mean performance obtained when classifying those graphs. As it was expected, we found a positive correlation between both edge and node assortativity for all full classifiers, with the Spearman's rank coefficient ranging between 0.44 and 0.73 for node assortativity and between 0.44 and 0.71 for edge assortativity.

Table 4. Spearman's rank correlation coefficient between error reduction and assortativity (r=0.35)

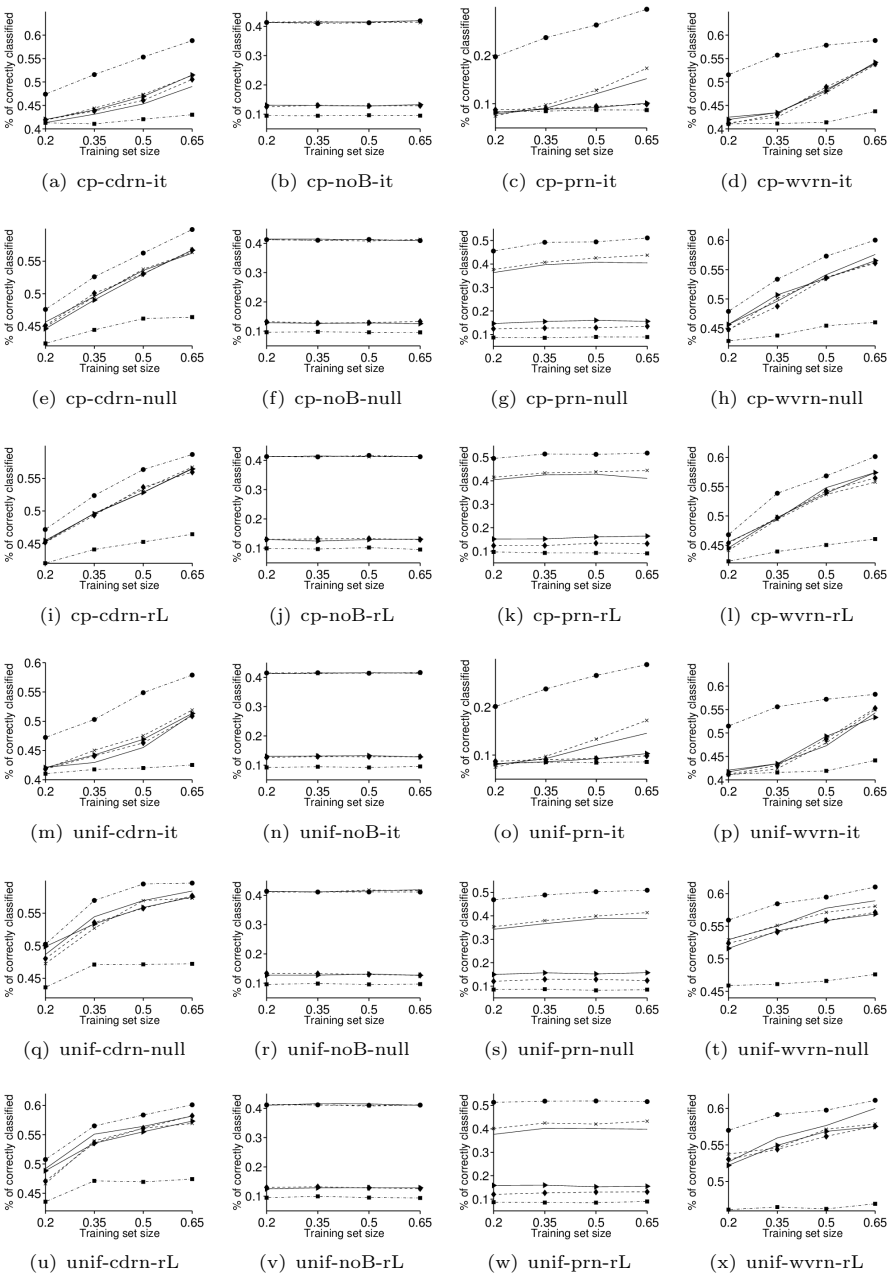
Full classifier	A_N	A_E	Full classifier	A_N	A_E
`cprior-wvrn-it`	0.6264	0.6315	`unif-wvrn-it`	0.5986	0.6049
`cprior-prn-it`	0.4481	0.4474	`unif-prn-it`	0.4448	0.4417
`cprior-nobayes-it`	0.4949	0.5054	`unif-nobayes-it`	0.5002	0.5121
`cprior-cdrn-norm-it`	0.7362	0.7175	`unif-cdrn-norm-it`	0.6819	0.6676
`cprior-wvrn-relaxLabel`	0.5213	0.5171	`unif-wvrn-relaxLabel`	0.5355	0.5415
`cprior-prn-relaxLabel`	0.4534	0.4831	`unif-prn-relaxLabel`	0.4423	0.4757
`cprior-nobayes-relaxLabel`	0.5100	0.5357	`unif-nobayes-relaxLabel`	0.5205	0.5471
`cprior-cdrn-norm-relaxLabl`	0.4863	0.5015	`unif-cdrn-norm-relaxLabl`	0.4894	0.4848
`cprior-wvrn-null`	0.5318	0.5304	`unif-wvrn-null`	0.5390	0.5491
`cprior-prn-null`	0.4627	0.4893	`unif-prn-null`	0.4669	0.5016
`cprior-nobayes-null`	0.5103	0.5342	`unif-nobayes-null`	0.5431	0.5644
`cprior-cdrn-norm-null`	0.4963	0.5063	`unif-cdrn-norm-null`	0.4956	0.4903

The Spearman's rank correlation coefficient is positive and greater than 0.44 for all the classifiers, which denotes that there exists a positive correlation between both node and edge assortativity and classification performance. The strength of this correlation varies depending on the specific classifier configuration. However, the values are quite high considering that different datasets are compared together. Although relative error reduction is used instead of classification accuracy, which already tries to compensate the differences between base errors on the different datasets, the different nature of the used graphs introduces additional complexity. When evaluating the different datasets independently[6], we found that the correlation was almost perfect for some datasets and worse for some other datasets. For instance, $\text{Cornell}_{\text{cocite}}$, $\text{Texas}_{\text{cocite}}$, and IMDb_{all} showed a correlation of 0.9429 (node assortativity and relative error reduction for the `cp-wvrn-it` configuration), while other datasets such as the four university ones with *link* edges showed very low correlation, or even a negative one.

3.5 Increasing Classification Performance

Once we have showed that we are able to increase assortativity using our scoring functions and that assortativity is positively correlated with performance, we want to observe the results of our third hypothesis, namely, that using scoring functions to correct weights can improve relational classification. In order to evaluate the degree in which using scoring functions improves networked classification, we use the 24 different full classifiers with all the available graphs. Since

[6] We omit these individual results due to space constraints.



(a) cp-cdrn-it (b) cp-noB-it (c) cp-prn-it (d) cp-wvrn-it

(e) cp-cdrn-null (f) cp-noB-null (g) cp-prn-null (h) cp-wvrn-null

(i) cp-cdrn-rL (j) cp-noB-rL (k) cp-prn-rL (l) cp-wvrn-rL

(m) unif-cdrn-it (n) unif-noB-it (o) unif-prn-it (p) unif-wvrn-it

(q) unif-cdrn-null (r) unif-noB-null (s) unif-prn-null (t) unif-wvrn-null

(u) unif-cdrn-rL (v) unif-noB-rL (w) unif-prn-rL (x) unif-wvrn-rL

Fig. 1. Performances comparison for all the classifiers and the different graph variations for the Cornell$_{\text{cocite}}$ dataset: Original (—), AA (-▶-), CC (-×-), CN (-♦-), JI (-●-), PA (-■-)

we have 14 original graphs and 5 different variations of each of these graphs can be obtained by using the different scoring functions, all the experiments are done with 14×6 graphs. For each graph and classifier, we repeat the process of selecting new train and test sets 100 times and define the performance of the full classifier with respect to a given graph and a labeled ratio r as the mean of these 100 different runs. We repeated the process for different labeled ratios (train set sizes): 20%, 35%, 50%, and 65%.

Figure 1 shows the results of this classification for $\text{Cornell}_{\text{cocite}}$ dataset. First, we can see that for all the classifiers but those using Network-Only Bayes (`noB`), classification accuracy increases as the training set size grows. As the labeled ratio increases, best models can be built and more correct information is available to do the predictions.

Second, we can appreciate that the performance offered by the scoring functions strongly depends on the specific relational classifier (RL) used. Graphs showing the results for the same RL instantiation and different LC and IC components present very similar curves.

For `cdrn-cos` and `wvrn`, the graph modified with JI leads to the best performance; the graphs modified with AA, CC and CN give similar results than the original graph, sometimes showing slightly better performance than the original graph; PA modifications offer the worst results, not being able to increase performance over the original graph. Performance when using `prn` is also consistent when varying the LC and IC components: the graph modified with JI always offers the best accuracy, sometimes increasing performance over 10%; CC is slightly better than the original graph; and AA, CN, and PA do not overcome the performance achieved with the original graph. Network-Only Bayes results are the same for all the LC-IC variations.

Independently of the selected full classifier, the graph modified with Preferential Attachment always offers worse performance than all the other graphs. This is consistent with the results showed in Section 3.2, where we could observe that the assortativity values always decreased when using PA as scoring function. This is also true for the graphs modified with JI, where we could see that assortativity always increased along with performance.

Due to space constrains, we are not able to include the results for all the datasets. The results for the other datasets showed the same consistency when using the same relational classifier and varying the LC and IC components. JI also regularly performed better than all the other alternatives for all *fc* when testing $\text{Washington}_{\text{cocite}}$, $\text{Wisconsin}_{\text{cocite}}$, $\text{Texas}_{\text{cocite}}$, and IMDb_{all}.[7] JI was overcome by CC for some specific full classifiers for $\text{Texas}_{\text{link}}$ and $\text{Cornell}_{\text{link}}$ datasets, and sometimes for other modified graphs or even for the original graph for the $\text{Washington}_{\text{link}}$ and $\text{Wisconsin}_{\text{link}}$ datasets. However, for both Cora and Industry datasets, the graph modified with JI did not show a significant improvement over the original graph.

[7] The exceptions were 2 out of 24 *fc* in $\text{Texas}_{\text{cocite}}$ for which the original graph performed better than JI, and some specific r values and classifiers in IMDb_{all}.

4 Related Work

The problem of classifying networked data has been a recent focus of activity in the machine learning research community, with special interest on adapting traditional machine learning techniques to networked data classification.

In [5], the authors present a relational classifier toolkit. Beyond the actual toolkit itself and by describing each of its modules, the authors review different algorithms that can be used to classify networked data.

Many algorithms for relational classifiers have been proposed in the past.

In [1] the authors present the Relational Neighbor (RL) classifier based on the principle of homophily, where the probability of a sample belonging to a given class is proportional to the number of neighbors of that sample belonging to the same class.

The Weighted Vote Relational Classifier (WVRN) estimates class-membership of a node as the weighted mean of the class-membership probabilities of the neighbors of that node.

The Class-Distribution Relational Neighbor classifier (CDRN) is presented in [5], where the probability of class membership of a node is estimated by the similarity of its class vector with the class reference vector. The class vector of a node is defined as the vector of summed linkage weights to the various classes and the class reference vector for a given class is the average of the class vectors for nodes known to be of that class.

Network Only Bayes classifier (nBC) [2] uses naive Bayes classification based on the classes of the nodes' neighbors to classify hyperlinked documents.

In [4] the Network-Only Link-Based classification (nLB) is presented, which uses regularized logistic regression models to classify networked data.

Since in relational classification problems entities are interlinked, the predicted class of a specific node may have consequences on the prediction of another node's class. For this reason, the method of independently classifying entities, which may be of use in traditional machine learning approaches, may not be the best way to deal with interlinked data. The process of simultaneously classifying a set of linked entities is known as collective inference. It has been shown that collective inference improves classification accuracy [12].

Collective inference may improve probabilistic inference in networked data [12]. Many CI methods are used in relational learning: Gibbs sampling [13], relaxation labeling [2], and iterative classification [4, 14] are the most used.

Relational classification has been applied to email classification [15], with a dataset of mails being linked only by parent-children relationships; to topic classification of hypertext documents [2]; to predict movie success with IMDb data, linking movies with a shared production company [1, 5]; to sub-topic prediction in machine learning papers [5]; to age, gender, and location prediction of bloggers [16]; and many other networked data classification problems.

5 Conclusions

We have showed that it is possible to increase the assortativity of a graph according to the node class labels with a very simple technique based on the usage of scoring functions. We have evaluated different scoring functions and demonstrated that using Jaccard Index (JI) always results in an increase on edge assortativity and, on all datasets but one, also in node assortativity. The usage of Common Neighbors (CN) and Adamic-Adar (AA) also leads to an increase on both node and edge assortativity for most of the tested datasets.

Although we have showed that there is a positive correlation between an increase on assortativity and an increase on classification performance, this correlation is not perfect (which supports preliminary tests done in [5]). Note that while we are dealing with a single assortativity value for each graph, many variables are involved in the performance obtained when classifying: from the specific configuration that the classifier adopts to the effect of choosing a concrete split of the training and test samples. So each assortativity value is compared against multiple classification performance results obtained when using different full classifiers.

Regarding the performance improvements achieved when using the modified graphs, the experiments showed that using Jaccard Index to modify the weights of the edges results in a general improvement of classification performance, although not for absolutely all the possible classifier configurations and datasets. The performance improvement when using CC, AA, and CN as scoring functions strongly depended on the selected dataset and, in a lesser extent, on the relational classifier instantiated. This opens an interesting line for future work: trying to identify the set of graph properties that determine which classifier (and scoring function) will lead to the best performance results.

Moreover, in this paper we were focused on evaluating different scoring functions and their effect on assortativity and performance. However, no specific efforts were devoted to construct the best possible scoring function. In this sense, a combination of the scoring functions that offered the best results may lead to a higher increase on classification performance. Trying to find the best possible scoring function is left as future work.

The techniques described in this paper can be applied to directed graphs following the naive procedure of computing the scoring functions over the underlying undirected graph obtained when omitting the direction of the edges. Since afterwards the results of the scoring functions are multiplied by the original weight in order to compute the new weight, edges between the same nodes differing only on the direction could be able to obtain different modified weights. Although the procedure seems feasible, it will be interesting to think about other techniques improving this naive approach.

Acknowledgments

This work was partially supported by the Spanish MCYT and the FEDER funds under grants TIN2011-27076-C03 “CO- PRIVACY”, TSI2007-65406-C03-

03 “E-AEGIS”, CONSOLIDER CSD2007-00004 “ARES“, TIN2010-15764 “N-KHRONOUS”, and grant FPU-AP2010-0078.

References

1. Macskassy, S., Provost, F.: A simple relational classifier. In: Proceedings of the 2nd Workshop on Multi-Relational Data Mining, KDD2003. (2003) 64–76
2. Chakrabarti, S., Dom, B., Indyk, P.: Enhanced hypertext categorization using hyperlinks. In: Proceedings of the ACM SIGMOD International Conference on Management of data. Volume 27., New York, NY, USA, ACM Press (1998) 307–318
3. Perlich, C., Provost, F.: Distribution-based aggregation for relational learning with identifier attributes. Machine Learning **62**(1-2) (February 2006) 65–105
4. Lu, Q., Getoor, L.: Link-based classification using labeled and unlabeled data. In: Proceedings of the ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data. (2003)
5. Macskassy, S.A., Provost, F.: Classification in networked data: A toolkit and a univariate case study. Journal of Machine Learning Research **8** (May 2007) 935–983
6. Bilgic, M., Getoor, L.: Effective label acquisition for collective classification. In: Proceedings of the International Conference on Knowledge discovery and data mining. (2008) 43–51
7. Newman, M.E.J.: Mixing patterns in networks. Physical Review E **67** (Feb 2003) 026126
8. Liben, D., Kleinberg, J.: The link prediction problem for social networks. In: Proceedings of the International Conference on Information and knowledge management. (2003) 556–559
9. Adamic, L., Adar, E.: Friends and neighbors on the Web. Social Networks **25**(3) (2003) 211–230
10. Macskassy, S., Provost, F.: NetKit-SRL - network learning toolkit for statistical relational learning
11. Spearman, C.: The proof and measurement of association between two things. The American journal of psychology **15**(1) (1904) 72–101
12. Jensen, D., Neville, J., Gallagher, B.: Why collective inference improves relational classification. In: Proceedings of the International Conference on Knowledge discovery and data mining. (2004) 593–598
13. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. Pattern Analysis and Machine Intelligence, IEEE Transactions on **PAMI-6**(6) (November 1984) 721–741
14. Neville, J., Jensen, D.: Iterative classification in relational data. In: AAAI-2000 Workshop on Learning Statistical Models from Relational Data. (2000)
15. Carvalho, V., Cohen, W.: On the collective classification of email speech acts. In: Proceedings of the International Conference on Research and development in information retrieval. (2005) 345–352
16. Bhagat, S., Rozenbaum, I., Cormode, G.: Applying link-based classification to label blogs. In: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis. (2007) 92–101