An Experimental Study of Graph-Based Semi-Supervised Classification with Additional Node Information

B. Lebichot.[a,*], M. Saerens[a],

[a]*Machine Learning Group - ICTEAM & LSM, Université catholique de Louvain Place des Doyens 1, B-1348 Louvain-la-Neuve, Belgium.*

Abstract

The volume of data generated by internet and social networks is increasing every day, and there is a clear need for efficient ways of extracting useful information from them. As this information can take different forms, it is important to use all the available data representations for prediction. This is often referred to multi-view learning in pattern recognition and machine learning. In this paper, we focus our attention on supervised classification using both regular, plain, tabular, data and structural information coming from a network structure. In this context, 16 techniques are investigated and compared in this study and can be divided in three families: the first one uses only the plain data to fit a classification model, the second uses only the network structure and the last combines both information sources. The relative performances in these three settings are investigated. Furthermore, the effect of using a network embedding and well-known indicators in spatial statistics are also studied. Possible applications are automatic classification of web pages or other linked documents, of people in a social network or of proteins in a biological complex system, to name a few. Based on our comparison, we draw some general conclusions and advice to tackle this particular classification task: it is observed that some dataset labelings can be better explained by their graph structure (graph-driven), or by their features set (features-driven).

Keywords: Graph and network analysis, semi-supervised classification, network data, graph mining, multi-view learning.

*Corresponding author

Email addresses: bertrand.lebichot@uclouvain.be (B. Lebichot.), marco.saerens@uclouvain.be (M. Saerens)

1. Introduction

Nowadays, with the increasing volume of data generated, for instance by internet and social networks, there is a need for efficient ways to infer useful information from those network-based data. Moreover, these data can take several different forms and, in that case, it would be useful to use these alternative views in the prediction model. This is the purpose of multi-view learning [1, 2]. In this paper, we focus our attention on supervised classification using both regular tabular data defined on nodes and structural information coming from graphs or networks[1].

Of course, as discussed in [3] (see, e.g., [4] for a survey), many different approaches have been developed for information fusion in machine learning, pattern recognition and applied statistics. This includes [3] simple weighted averages (see, e.g., [5, 6]), Bayesian fusion (see, e.g., [5, 6]), majority vote (see, e.g., [7, 8, 9]), models coming from uncertainty reasoning [10] (see, e.g., [11]), standard multivariate statistical analysis techniques such as correspondence analysis [12], maximum entropy modeling (see, e.g., [13, 14, 3]), multi-view learning [1, 2] etc. In this work, we investigate several ways of combining the structural information provided by a network (or graph, both terms will be used interchangeably) of interactions between objects (for instance interactions between members of an online social network) and information associated to these different objects (for instance the gender of the person, her age, etc) for solving objects (nodes) classification problems.

As well-known, the goal of classification is to automatically label data to predefined classes. This is also called supervised learning since the method uses known labels (the desired prediction of an instance) to fit the classification model. One interesting variant is to use semi-supervised learning based on a network structure [15, 16, 17, 18, 19, 20, 21, 22, 23, 24].

Indeed, traditional pattern recognition, machine learning or data mining classification methods require large amounts of labeled training instances – which are often difficult to obtain – to fit accurate models. Semi-supervised learning methods can reduce the effort by including unlabeled samples. This name comes from the fact that the used dataset is a mixture of supervised and unsupervised data (it contains training samples that are unlabeled). Then, the classifier takes advantage from both the supervised and unsupervised data. The advantage here is that unlabeled data are often much less costly than labeled data. This technique often allows to reduce the amount

[1] Graph and network will be used interchangeably.

of labeled instances needed to achieve the same level of classification accuracy [24, 16]. In other words, exploiting the distribution of unlabeled data during the model fitting process can prove helpful.

Semi-supervised classification comes in two different settings: inductive and transductive [24]. The goal of the former setting is to predict the labels of future test data, unknown when fitting the model, while the second is to classify (only) the unlabeled instances of the training sample. Some often-used semi-supervised algorithms include: expectation-maximization with generative mixture models, self-training, co-training, transductive support vector machines, and graph-based methods [22, 23, 24, 23]. In this work, we focus on these graph-based methods to label nodes.

The structure of the data can also be of different types. This work focuses on a particular data structure: we assume that our dataset takes the form of a network with features associated to the nodes. Nodes are the samples of our dataset and links between these nodes represent a given type of interaction or relation between these samples (like a friendship relation on Facebook). For each node, a number of features or attributes characterizing it is also available (see Figure 1 for an example). Other data structures exist but are not studied in this paper; for instance:

- Different types of nodes can be present, with different types of features sets describing them.
- Different types of relations can link the different nodes.

This problem has numerous applications such as classification of individuals in social networks, categorization of linked documents (e.g. patents or scientific papers), or protein function prediction, to name a few. In this kind of application (as in many others), unlabeled data are usually available in large quantities and are easy to collect: friendship links can be recorded on Facebook, text documents can be crawled from the internet and DNA sequences of proteins are readily available from gene databases.

In this work, we investigate experimentally various models combining information on the nodes of the graph and the graph structure. Indeed, it has been shown that network information can improve significantly prediction accuracy in a number of contexts [25, 20]. A total of 16 classification algorithms using various combinations of data sources, mainly described in [17], are compared. The different considered algorithms are detailed in Section 4.

A standard support vector machine (SVM) classifier is used as a baseline algorithm, but we also investigated the ridge logistic regression classifier.

The results and conclusions obtained with this second classification model were similar to the SVM and are therefore not reported in this paper.

In short, the main questions investigated in this work are:

- Does the combination of features on nodes and network structure works better than using the features only?
- Does the combination of features on nodes and network structure works better than using the graph structure only?
- Which classifier performs best on network structure alone, without considering features on nodes?
- Which classifier performs best when combining information, that is, using network structure with features on nodes?

Finally, this comparison leads to some general conclusions and advices when tackling classification problems on network data with node features.

In summary, this work has four main contributions:

- The paper reviews different algorithms used for learning from both a graph structure and node features, mainly following [17]. Some algorithms are inductive while some others are transductive.
- An empirical comparison of those algorithms is performed on ten real-world datasets.
- We investigate the effect of extracting features from the graph structure (and some well-known indicators in spatial statistics) in a classification context.
- Finally, this comparison leads to some general conclusions and advices to tackle graph-based classification tasks.

The remaining of this paper is organized as follows. Section 2 provides some background and notation. Section 3 investigates related work. Section 4 introduces the investigated classification methods. Then, Section 5 presents the experimental methodology and the results. Finally, Section 6 concludes the paper.

2. Background and notation

This section aims to introduce the necessary theoretical background and notation used in the paper. Consider a weighted, undirected, strongly connected, graph or network G (with no self-loop) containing a set of n vertices $\mathcal{V}$ (or nodes) and a set of edges $\mathcal{E}$ (or arcs, links). The $n \times n$ **adjacency matrix** of the graph, containing non-negative affinities between nodes, is denoted as $\mathbf{A}$, with elements $a_{ij} \geq 0$.

We also introduce the **Laplacian matrix L** of the graph, defined in the usual manner:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{1}$$

where $\mathbf{D} = \mathbf{Diag}(\mathbf{Ae})$ is the diagonal (out)degree matrix of the graph G containing the $a_{i\bullet} = \sum_{j=1}^{n} a_{ij}$ on its diagonal and $\mathbf{e}$ is a column vector full of ones. One of the properties of $\mathbf{L}$ is that its eigenvalues provide useful information about the connectivity of the graph [26]. The smallest eigenvalue of $\mathbf{L}$ is always equals to 0, and the second smallest one is equals to 0 only if the graph is composed of at least two connected components. This last value is called the algebraic connectivity.

Moreover, a **natural random walk** on G is defined in the standard way. In node i, the random walker chooses the next edge to follow according to transition probabilities

$$p_{ij} = \frac{a_{ij}}{\sum_{j'=1}^{n} a_{ij'}} \tag{2}$$

representing the probability of jumping from node i to node $j \in \mathcal{S}ucc(i)$, the set of successor nodes of i. The corresponding $n \times n$ **transition probabilities matrix** will be denoted as $\mathbf{P}$ and is stochastic. Thus, the random walker chooses to follow an edge with a likelihood proportional to the affinity (apart from the sum-to-one normalization), therefore favoring edges with a large affinity.

Moreover, we will consider that each of the nodes of G has the same set of m features, or attributes, with no missing values. The column vector $\mathbf{x}_i$ contains the values of the m features of node i and x_{ij} states for the value of feature j taken by node i. Moreover, $\mathbf{X}_{\text{features}}$, or simply $\mathbf{X}$, will refer to the $n \times m$ data matrix containing the elements x_{ij}.

Finally we define $\mathbf{y}$ as the column vector containing the class labels of the nodes. More precisely, $\mathbf{y}^c$ is a binary vector indicating whether or not each node belongs to class number c. That is, $\mathbf{y}_i^c$ is equal to one if node i belongs to class c, and zero otherwise.

Recall that the purpose of the classification tasks will to predict the class of the unlabeled data (in a transductive setting), or to predict new test data

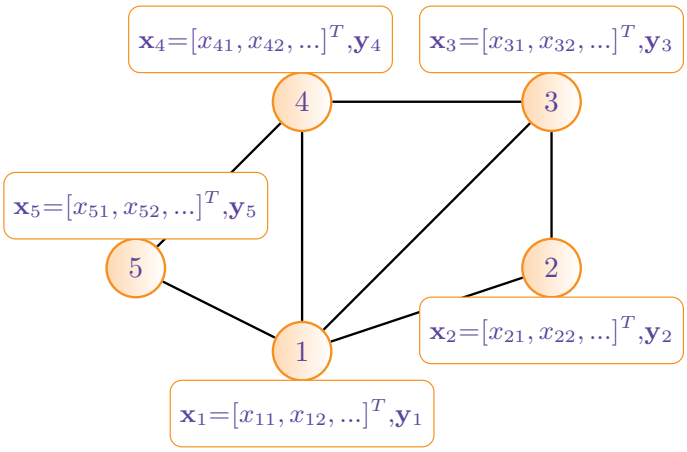

Figure 1: Example of graph with additional node information. Each node is characterized by a feature vector and a class label.

(in an inductive setting), while knowing the structure of the graph G, the values of the features $\mathbf{X}$ for all the nodes of G and the class labels $\mathbf{y}^c$ on the *labeled* nodes only for each c. Our baseline classifier based on features only will be a linear support vector machines (SVM).

3. Some related work

The 16 investigated models are presented in the next Section 4. In addition to those models, other approaches exist.

For example [27, 28] use a standard ridge regression model complemented by a Laplacian regularization term, which has been called the Laplacian regularized least squares. This option was investigated but provided poor results compared to reported models (therefore not reported).

Note that using a logistic ridge regression as the base classifier (instead of a support vector machine) was also investigated in this work but results are not reported here for conciseness as it provided performances similar to the SVM.

Laplacian support vector machines (LapSVMs) extend the SVM classifier in order to take the structure of the network into account. They exploit both the information on the nodes and the graph structure in order to categorize the nodes with the use of its Laplacian matrix (see Section 2). To this end, [27] proposed to add a graph Laplacian regularization term to the

traditional SVM cost function in order to obtain a semi-supervised version of this model. A Matlab toolbox for this model is available but provided poor results in terms of performance and tractability. This model was therefore not included in our comparisons.

Chakrabarti et al. [29] developed, in the context of patents classification, a naive Bayes model in the presence of structural autocorrelation. The main idea is to use a naive Bayes classifier combining both feature information on the nodes and structural information by making some independence assumptions. However, we found that this procedure is very time consuming, even for small-size networks, and decided to not include it in the present work as results were impossible to obtain on the larger datasets.

Also note that various semi-supervised classifiers based on network data only (features on nodes are not available) were also developed [20, 23]. The interested reader is invited to consult, e.g., [30, 15, 16, 18, 21, 22, 23, 24] (and included references), focused on this topic, for comprehensive reviews. Finally, an interesting survey and a comparative experiment of related methods, but more focused on relational learning, can be found in [20].

Finally, our problem is a particular case of the more general multi-view learning framework: Multi-view learning is an emerging direction in machine learning which considers learning with multiple views to improve the generalization performance [1, 2]. Also known as data fusion, it learns from multiple feature sets. In our particular case, we study the problem of learning from two information sources: on the one hand features and on the other hand a single graph.

Multi-view learning methods are divided into three major categories : co-training style algorithms, co-regularization style algorithms and margin-consistency style algorithms [1, 2]. Some of these multi-view algorithms will be investigated.

- Co-training is historically the first family of multi-view algorithms: the classifier is trained alternately on two distinct views with confident labels for the unlabeled data (see [31] for details). Examples are co-EM, co-testing and robust co-training [2].

- Co-regularization algorithms make sure that data from multiple views are consistent by adding a regularization term in the classifier objective function: the disagreement between the discriminant functions of the two views. Examples are sparse multi-view SVMs, multi-view TSVMs, multi-view Laplacian SVMs and multi-view Laplacian TSVMs [2].

- Margin-consistency style algorithms make use of the latent consistency

of classification results from multiple views by using the framework of maximize entropy discrimination (MED, see [1, 2] for details).

4. Description of relevant classification methods

The different classification methods compared in this work are briefly presented in this section, which is largely inspired by [17]. For a more thorough presentation, see the provided references to the original works or [17]. The classification models are sorted into different families: graph embedding-based classifiers, extensions of feature-based classifiers, graph-based classifiers and multi-view learning.

All these methods rely on a standard, strong, assumption about the distribution of the labels in the graph: it is assumed that *neighboring nodes are likely to belong to the same class* and thus are likely to share the same class label (see, e.g., [19, 17, 21]). This assumption is often called *homophily*, *associativity*, *local consistency*, or *structural autocorrelation*. As described now, this hypothesis can be tested by using some autocorrelation measures widely used in spatial statistics. Moreover, the experiments show that if the assumption is not verified, the methods exploiting the graph structure do not bring any useful information to classify the nodes.

4.1. Graph embedding-based classifiers

A first interesting way to combine information from the features on the nodes and from the graph structure is to perform a *graph embedding* projecting the nodes of the graph into a low-dimensional space (an embedding space) preserving as much as possible its structural information, and then use the coordinates of the projected nodes as *additional features* in a standard classification model, such as a logistic regression or a support vector machine.

This procedure has been proposed, e.g., in the field of spatial statistics for ecological modeling [32, 33, 34], but also more recently in data mining [35, 36, 37, 38]. While many graph embedding techniques could be used, [33] suggests to exploit Moran's or Geary's index of spatial autocorrelation in order to compute the embedding.

Let us briefly develop their approach by closely following [17] (see this reference for more information). Moran's I and Geary's c (see, e.g., [39, 40, 41, 42]) are two coefficients commonly used in spatial statistics in order to test the hypothesis of spatial autocorrelation of a numerical quantity defined on the nodes. This interesting property will be investigated on the datasets used in the experimental section, in the context of semisupervised

classification (see, e.g., Table 8). Four different possibilities will be considered to extract features from the graph structure: maximizing Moran's I, minimizing Geary's c, local principal component analysis and maximizing the bag-of-path (BoP) modularity.

4.1.1. Maximizing Moran's I

Moran's I [43, 44] is given by

$$I(\mathbf{x}) = \frac{n}{a_{\bullet\bullet}} \frac{\sum_{i,j=1}^{n} a_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_{i'=1}^{n}(x_{i'} - \bar{x})^2} \tag{3}$$

where x_i and x_j are the values observed on nodes i and j respectively, for a considered quantity defined on the nodes (for instance the age of the person in a social network). The column vector $\mathbf{x}$ is the vector containing the values x_i on all nodes and $\bar{x}$ is the average value of $\mathbf{x}$. Then, $a_{\bullet\bullet}$ is simply the sum of all entries of $\mathbf{A}$ – the volume of the graph.

$I(\mathbf{x})$ can be interpreted as a correlation coefficient similar to the Pearson correlation coefficient [39, 40, 41, 42]. The numerator is a measure of covariance among the neighboring x_i in G, while the denominator is a measure of variance. It is a common misconception that I is in the interval $[-1, +1]$. Instead, the upper and lower bound depends on n, $a_{\bullet\bullet}$, but also on the maximum and minimum eigenvalues of $\mathbf{A}$ (see [45] for details). A value close to $I_0 = -1/(n-1) \approx 0$ [45] indicates no evidence of autocorrelation, a larger value indicates positive autocorrelation and a smaller value indicates negative autocorrelation (autocorrelation means that neighboring nodes tend to take similar values).

In matrix form, Equation (3) can be rewritten as

$$I(\mathbf{x}) = \frac{n}{a_{\bullet\bullet}} \frac{\mathbf{x}^T\mathbf{HAHx}}{\mathbf{x}^T\mathbf{Hx}} \tag{4}$$

where $\mathbf{H} = (\mathbf{I} - \mathbf{E}/n)$ is the centering matrix [46] and $\mathbf{E}$ is a matrix full of ones. Note that the centering matrix is idempotent, $\mathbf{HH} = \mathbf{H}$.

The objective is now to find the scores $\mathbf{x}$ that achieve the largest autocorrelation, as defined by Moran's index. This corresponds to the values that most explain the structure of G. It can be obtained by setting the gradient equal to zero; we then obtain the following generalized eigensystem:

$$\mathbf{HAHx}' = \lambda\mathbf{x}', \text{ and then } \mathbf{x} = \mathbf{Hx}' \tag{5}$$

The idea is thus to extract the first eigenvector $\mathbf{x}_1$ of the centered adjacency matrix (5) corresponding to the *largest* eigenvalue λ_1 and then to

compute the second-largest eigenvector, $\mathbf{x}_2$, orthogonal to $\mathbf{x}_1$, etc. The eigenvalues λ_i are proportional to the corresponding explained Moran's $I(\mathbf{x})$.

The p largest centered eigenvectors of (5) are thus extracted and then used as additional p features for a supervised classification model (here a SVM). In other words, $\mathbf{X}_{\text{Moran}} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p]^T$ is a new data matrix, capturing the structural information of G, that can be concatenated to the feature-based data matrix $\mathbf{X}_{\text{features}}$, forming the extended data matrix $[\mathbf{X}_{\text{features}}, \mathbf{X}_{\text{Moran}}]$.

4.1.2. Minimizing Geary's c

On the other hand, Geary's c [47] is another estimate of autocorrelation given by

$$c(\mathbf{x}) = \frac{(n-1)}{2\, a_{\bullet\bullet}} \frac{\sum_{i,j=1}^{n} a_{ij}(x_i - x_j)^2}{\sum_{i'=1}^{n}(x_{i'} - \bar{x})^2} \tag{6}$$

and is related to Moran's I. However, while Moran's I considers a covariance between neighboring nodes, Geary's c considers distances between values on pairs of neighboring nodes. Once again, lower and upper bounds are often assumed to be respectively 0 and 2 with 0 indicating perfect positive autocorrelation and 2 indicating perfect negative autocorrelation [34, 40, 42]. However, as for Moran's I, [45] shows that the bounds are more complex and actually depend on n, $a_{\bullet\bullet}$, and the maximum and minimum eigenvalues of $\mathbf{A}$. $c = 1$ indicates no evidence of autocorrelation.

In matrix form, Geary's c can be rewritten as

$$c(\mathbf{x}) = \frac{(n-1)}{2a_{\bullet\bullet}} \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{H}\mathbf{x}}. \tag{7}$$

This time, the objective is to find the score vector minimizing Geary's c. By proceeding as for Moran's I, we find that minimizing $c(\mathbf{x})$ aims to compute the p *lowest* non-trivial eigenvectors of the Laplacian matrix:

$$\mathbf{L}\mathbf{x} = \lambda\mathbf{H}\mathbf{x} \tag{8}$$

and then use these eigenvectors as additional p features in a classification model. We therefore end up with the problem of computing the lowest eigenvectors of the Laplacian matrix, which also appears in spectral clustering (ratio cut, see, e.g., [48, 17, 49]).

Geary's c has a computational advantage over Moran's I: the Laplacian matrix is usually sparse, which is not the case for Moran's I. Moreover, note that since the Laplacian matrix $\mathbf{L}$ is centered, any non-trivial solution of $\mathbf{L}\mathbf{x} = \lambda\mathbf{x}$ is also a solution of Equation (8).

4.1.3. Local principal component analysis

In [50, 51], the authors propose to use a measure of local, structural, association between nodes, the **contiguity ratio** defined as

$$cr(\mathbf{x}) = \frac{\sum_{i=1}^{n}(x_i - m_i)^2}{\sum_{i'=1}^{n}(x_{i'} - \bar{x})^2}, \text{ with } m_i = \sum_{j \in \mathcal{N}(i)} p_{ij} x_j. \tag{9}$$

and m_i is the average value observed on the neighbors of i, $\mathcal{N}(i)$. As for Geary's index, the value is close to zero when there is a strong structural association. However, there are no clear bounds indicating no structural association or negative correlation [51].

The numerator of Equation (9) is the mean squared difference between the value on a node and the average of its neighboring values; it is called the local variance in [51]. The denominator is the standard sample variance. In matrix form,

$$cr(\mathbf{x}) = \frac{\mathbf{x}^T(\mathbf{I} - \mathbf{P})^T(\mathbf{I} - \mathbf{P})\mathbf{x}}{\mathbf{x}^T\mathbf{H}\mathbf{x}}. \tag{10}$$

Proceeding as for Geary and Moran's indexes, minimizing $cr(\mathbf{x})$ aims to solve

$$(\mathbf{I} - \mathbf{P})^T(\mathbf{I} - \mathbf{P})\mathbf{x} = \lambda\mathbf{H}\mathbf{x} \tag{11}$$

Here again, eigenvectors corresponding to the smallest non-trivial eigenvalues of the eigensystem (11) are extracted. This procedure is also referred to as **local principal component** analysis in [51].

4.1.4. Bag-of-path modularity

For this algorithm, we also compute a number of structural features, but now derived from the modularity measure (which was introduced by Newman and co-workers in [52, 53, 49]) redefined in the bag-of-path (BoP) framework [54], and concatenate them to the node features, $[\mathbf{X}_{\text{features}}, \mathbf{X}_{\text{BoPMod}}]$. Again, a SVM is then used to classify all unlabeled nodes. Indeed, it has been shown that using the dominant eigenvectors of the BoP modularity matrix provides better performances than using the eigenvectors of the standard modularity matrix [54]. The results for the standard modularity matrix are therefore not reported in this work.

It can be shown (see [54] for details) that the BoP modularity matrix is equal to

$$\mathbf{Q}_{\text{BoP}} = \mathbf{Z} - \frac{(\mathbf{Z}\mathbf{e})(\mathbf{e}^T\mathbf{Z})}{\mathbf{e}^T\mathbf{Z}\mathbf{e}} \tag{12}$$

where $\mathbf{Z}$ is the fundamental bag-of-path $n \times n$ matrix and $\mathbf{e}$ is a length n column vector full of ones. Then as for Moran's I and Geary's c, an eigensystem

$$\mathbf{Q}_{\text{BoP}}\mathbf{x} = \lambda\mathbf{x} \tag{13}$$

must be solved and the largest eigenvectors are used as new, additional, structural, features.

4.2. Extensions of standard feature-based classifiers

These techniques rely on extensions of standard feature-based classifiers (for instance a logistic regression model or a support vector machine). The extension is defined in order to take the network structure into account. As before, the discussion is based on [17].

4.2.1. The AutoSVM: taking autocovariates into account

This model is also known as the **autologistic** or **autologit** model [55, 56, 57, 58], and is frequently used in the spatial statistics and biostatistics fields.

Note that, as a SVM is used as base classifier in this work (see Section 1), we adapted this model (instead of the logistic regression in [57]) in order to take the graph structure into account. The method is based on the quantity $ac_i^c = \sum_{j \in \mathcal{N}(i)} p_{ij}\hat{y}_j^c$, where $\hat{y}_j^c$ is the predicted membership of node j, called the **autocovariate** in [57] (other forms are possible, see [56, 57]). It corresponds to the weighted averaged membership to class c within the neighborhood of i: it indicates to which extent neighbors of i belong to class c. The assumption is that node i has a higher chance to belong to class c if its neighbors also belong to that class.

However, since the predicted value $\hat{y}_j^c$ depends on the occurrence of the predicted value on other nodes, fitting the model is not straightforward. For the autologistic model, it goes through the maximization of the (pseudo-) likelihood (see for example [59, 55]), but we will consider a simpler alternative [57] which uses a kind of expectation-maximization-like heuristics (EM, see, e.g. [60, 61]), and is easy to adapt to our SVM classifier.

Following [17], here is a summary of the estimation procedure proposed in [57]:

1. At $t = 0$, initialize the predicted class memberships $\hat{y}_i^c(t = 0)$ of the unlabeled nodes by a standard SVM depending on the feature vectors only, from which we disregard the structural information (the information about neighbors' labels). For the labeled nodes, the membership values are of course not modified.

2. Compute the current values of the autocovariates, $ac_i^c = \sum_{j \in \mathcal{N}(i)} p_{ij} \hat{y}_j^c(t)$, for all nodes.
3. Train a so-called **autoSVM** model based on these current autocovariate values as well as the features on nodes, providing parameter estimates $\hat{\mathbf{w}}^c$.
4. Compute the new predicted class memberships $\hat{y}_i^c(t+1)$ of the set of unlabeled nodes from the fitted autoSVM model. After having considered all the unlabeled nodes, we have the new predicted values $\hat{y}_i^c(t+1)$.
5. Steps 2 to 4 are iterated until convergence of the predicted membership values $\hat{y}_i^c(t)$.

4.2.2. Double kernel SVM

Here, we describe another simple way of combining the information coming from features on nodes and graph structure. The basic idea ([62, 17]) is to

1. Compute a $n \times n$ kernel matrix based on node features [63, 64], for instance a linear kernel or a gaussian kernel.
2. Compute a $n \times n$ kernel matrix on the graph [65, 17, 66, 64], for instance the regularized commute-time kernel (see Subsection 5.2.2).
3. Fit a SVM based on these two combined kernels.

Then, by using the kernel trick, everything happens as if the new data matrix is

$$\mathbf{X}_{\text{new}} = [\mathbf{K}_{\mathbf{A}}, \mathbf{K}_{\mathbf{X}}] \tag{14}$$

where $\mathbf{K}_{\mathbf{A}}$ is a kernel on a graph and $\mathbf{K}_{\mathbf{X}} = \mathbf{X}\mathbf{X}^T$ is the kernel matrix associated to the features on the nodes (see [17] for details). Then, we can fit a SVM classifier based on this new data matrix and the labeled nodes.

4.2.3. A spatial autoregressive model

This model is a spatial extension of a standard regression model [67] and is well known in spatial econometrics. This extended model assumes that the predicted vector of class memberships $\hat{\mathbf{y}}^c$ is generated in each class c according to

$$\hat{\mathbf{y}}^c = \rho \mathbf{P} \hat{\mathbf{y}}^c + \mathbf{X}\mathbf{w}^c + \boldsymbol{\epsilon} \tag{15}$$

where $\mathbf{w}^c$ is the usual parameter vector, ρ is a scalar parameter introduced to account for the structural dependency through $\mathbf{P}$ and $\boldsymbol{\epsilon}$ is an error term. This model is somehow related to the previously introduced AutoSVM model. Obviously if ρ is equal to zero, there is no structural

dependency and the model reduces to a standard linear regression model. Lesage's Econometrics Matlab toolbox was used for the implementation of this model [67]; see this reference for more information.

4.3. Graph-based classifiers

We also investigate some semi-supervised methods based on the graph structure only (no node feature exists or features are simply not taken into account). We selected the techniques performing best in a series of experimental comparisons [65, 30, 68]. As already discussed, they rely on some strong assumptions about the distribution of labels: that neighboring nodes (or "close nodes") are likely to share the same class label [16].

4.3.1. The bag-of-paths group betweenness

This model [30], inspired by [69, 70], considers a bag containing all the possible paths between pairs of nodes in G. Then, a Boltzmann distribution, depending on a temperature parameter T, is defined on the set of paths such that long (high-cost) paths have a low probability of being drawn from the bag, while short (low-cost) paths have a high probability of being drawn.

The **bag-of-paths (BoP) probabilities**, $P(s = i, e = j)$, providing the probability of drawing a path starting in i and ending in j, can be computed in closed form and a betweenness measure quantifying to which extend a node is in-between two node is defined. A node receives a high betweenness if it has a large a posteriori probability of appearing on paths connecting two arbitrary nodes of the network. A group betweenness between classes is defined as the sum of the contribution of all paths starting and ending in a particular class, and passing through the considered node. Each unlabeled node is then classified according to the class showing the highest group betweenness. More information can be found in [30].

4.3.2. A sum-of-similarities based on the regularized commute time kernel

We also investigate a classification procedure based on a simple alignment with the regularized commute time kernel (RCT) $\mathbf{K}$, a **sum-of-similarities** defined by $\mathbf{K}\mathbf{y}^c$, with $\mathbf{K} = (\mathbf{D} - \alpha\mathbf{A})^{-1}$ [71, 65, 17]. This expression quantifies to which extend each node is close (in terms of the similarity provided by the regularized commute time kernel) to class c. This similarity is computed for each class c in turn. Then, each node is assigned to the class showing the largest sum of similarities. It corresponds to a variant of the k nearest neighbors classifier when dealing with a similarity matrix instead of distances.

Element i, j of this kernel can be interpreted as the discounted cumulated probability of visiting node j when starting from node i. The (scalar) parameter $\alpha \in [0, 1]$ corresponds to a killed random walk where the random walker has a $(1 - \alpha)$ probability of disappearing at each step. Other graph kernels could be used in a sum-of-similarities setting [17] but this one consistently provided good results in comparative studies of graph-based semi-supervised classification techniques [65, 68, 69].

4.4. Multi-view learning

Finally, Multi-view learning is also considered. The three classes of Multi-view learning were recalled in Section 3. The original Co-training algorithms [31] was re-implemented based on SVMs. For Co-regularization algorithms, a kernel canonical correlation [72] tool was kindly provided in Matlab by one of the authors. However, for Margin-consistency algorithms, we failed to find an efficient Matlab implementation.

4.4.1. Co-training

Given a set $\mathcal{L}$ of labeled samples/nodes and a set $\mathcal{U}$ of unlabeled samples/nodes, the algorithm iterates the following two-steps procedure. First, use $\mathcal{L}$ to train two distinct classifiers: here one is based on the features only and the second is based on a kernel extracted from the graph only. Second, allow each of the classifier to label a small subset of $\mathcal{U}$ with the highest posteriors provided by both views (here, distances to the hyperplane were used instead), then update $\mathcal{L}$ and $\mathcal{U}$. When all unlabeled nodes have been labeled, the procedure stops. See [31] for more details.

4.4.2. Kernel canonical correlation analysis

Kernel canonical correlation analysis [72] is an kernel extension of standard canonical correlation analysis. The idea is to project the data in a new space, and constrain the multiple transformed feature sets to be as close as possible, while regularizing the self covariance of each transformed feature sets to be small enough. The goal is to find projection vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ such that

$$\frac{\text{cov}(\mathbf{X}\mathbf{w}_1, \mathbf{Y}\mathbf{w}_2)}{\sqrt{\text{var}(\mathbf{X}\mathbf{w}_1)\text{var}(\mathbf{Y}\mathbf{w}_2)}} \tag{16}$$

is maximal. var() and cov() are respectively the variance and covariance measures and $\mathbf{X}$ and $\mathbf{Y}$ are two kernel-based views. In our case, $\mathbf{X}$ corresponds to the regular features and $\mathbf{Y}$ corresponds to a kernel built from the graph (here, we chose the RCT kernel with $\alpha = 0.85$). See [2] or [72] for

Table 1: Class distribution of the four *WebKB* datasets.

Class	**Cornell** (DB1)	**Texas** (DB2)	**Washington** (DB3)	**Wisconsin** (DB4)
Course	42	33	59	70
Faculty	32	30	25	32
Student	83	101	103	118
Project + staff	38	19	28	31
Total	195	183	230	251
Majority class (%)	42.6	55.2	44.8	47.0
Number of features	1704	1704	1704	1704

more details. Notice that this method is related to the double kernel SVM of Subsection 4.2.2.

5. Experiments

In this section, the different classification methods will be compared on semi-supervised classification tasks and several datasets. The goal is to classify unlabeled nodes and to compare the results obtained by the different methods in terms of classification accuracy.

This section is organized as follows. First, the datasets used for semi-supervised classification are described in Subsection 5.1. Then, the compared methods are recalled in Subsection 5.2. The experimental methodology is explained in Subsection 5.3. Finally, results are presented and discussed in Subsection 5.4.

5.1. Datasets

All datasets are described by (i) the adjacency matrix $\mathbf{A}$ of the underlying graph, (ii) class vectors $\mathbf{y}^c$ (to predict) and (iii) a number of features on nodes gathered in the data matrix $\mathbf{X}_{\text{features}}$. Using a chi-square test, we kept only the 100 most significant features for each dataset. The datasets are available at `http://www.isys.ucl.ac.be/staff/lebichot/research.htm`.

For each of these dataset, if more than one connected component is present, we only use the largest connected component, deleting all the others nodes, features and target classes. Also, we choose to work with undirected graphs for all datasets: if a graph is directed, we used $\mathbf{A} = (\mathbf{A}^T + \mathbf{A})/2$ to introduce reciprocal edges.

Table 2: Class distribution of the three *Ego facebook* datasets.

Class	**FB 107** (DB5)	**FB 1684** (DB6)	**FB 1912** (DB7)
Main group	524	568	737
Other groups	232	225	308
Total	756	793	1045
Majority class (%)	69.3	71.2	70.5
Number of features	480	319	576

- The four *WebKB* datasets (**DB1-DB4**) [73] consist of web pages gathered from computer science departments from four universities (there are four datasets, one for each university), with each page manually labeled into one of four categories: course, faculty, student and project [20]. The pages are linked by citations (if x links to y then it means that y is cited by x, not to be confused with the four co-citation datasets). Each web page in the dataset is also characterized by a binary word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1703 unique words (words appearing less than 10 times were ignored). Originally, a fifth category, Staff, was present but since it contained only very few instances, it was merged with the Project class. Details on these datasets are shown in Table 1.

- The three *Ego Facebook* datasets (**DB5-DB7**) [74] consist of "circles" (or friends communities) from Facebook. Facebook data were collected from survey participants using a Facebook application. The original dataset includes node features (profiles), circles, and ego networks for 10 networks. Those data are anonymized and the exact signification of the circles is unknown [74]. We use only the three first networks and the classification task is to predict the affiliation to a circle. Details on these datasets are shown in Table 2. Each dataset has two classes.

- The CiteSeer dataset (**DB8**) [73] consists of 3312 scientific publications classified into six classes. The pages are linked by citation. Each publication in the dataset is described by a binary word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3703 unique words (words appearing less than 10 times were ignored). The target variable contains the topic

Table 3: Class distribution of the *Citeseer, Cora* and *Wikipedia* datasets.

Class	**Citeseer** (DB8)	**Cora** (DB9)	**Wikipedia** (DB10)
Class 1	269	285	248
Class 2	455	406	509
Class 3	300	726	194
Class 4	75	379	99
Class 5	78	214	152
Class 6	188	131	409
Class 7		344	181
Class 8			128
Class 9			364
Class 10			351
Class 11			194
Class 12			81
Class 13			233
Class 14			111
Total	1392	2708	3271
Majority class (%)	32.7	26.8	15.6
Number of features	3703	1434	4973

of the publications (six topics). Details on this dataset are shown in Table 3.

- The Cora dataset (**DB9**) [73] consists of 2708 scientific publications classified into one of seven classes denoting topics as for previous dataset. Pages are linked by citations. Each publication is also described by a binary word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1434 unique words or features (words appearing less than 10 times were ignored). The target variable represents the topic of the publications. Details on this dataset are shown in Table 3.

- The Wikipedia dataset (**DB10**) [73] consists of 3271 Wikipedia articles that appeared in the featured list in the period Oct. 7-21, 2009. Each document belongs to one of 14 distinct broad categories, which were obtained by using the category under which each article is listed. After stemming and stop-word removal, the content of each document is represented by a tf/idf-weighted feature vector, for a total of 4973 words. Pages are linked by citation. The target variable represents the articles field (14 different topics). Details on this dataset are shown in

Table 3.

Moreover, in order to study the impact of the relative information provided by the *graph structure* and the *features on nodes*, we created new derived datasets by *weakening gradually* the information provided by the node features. More precisely, for each dataset, the features available on the nodes have been ranked by decreasing association (using a chi-square statistics) with the target classes to be predicted. Then, datasets with *subsets* of the features containing respectively the 5 (**5F**), 10 (**10F**), 25 (**25F**), 50 (**50F**) and 100 (**100F**) most informative features were created (5 sets of features). These datasets are weakened versions of the original datasets, allowing to investigate the respective impact of features on nodes and graph structure. We also investigated sets with more features (200 and 400), but conclusions were the same, so that they are not reported here for conciseness.

5.2. Compared classification models

In this work, a transductive scheme is used, as we need to know the whole graph structure to label unlabeled nodes. The 16 different algorithms described before will be compared and can be sorted in three categories, according to the information they use. Some algorithms use only features to build the model (denoted as **X** – *data matrix with features only*), others use only the graph structure (denoted as **A** – *adjacency matrix of the graph only*), and the third category uses both the structure of the graph and the features of the nodes (denoted as **AX** – *combined information*).

5.2.1. Using features on nodes only

This reduces to a standard classification problem and we use a linear Support Vector Machine (SVM) based on the features of the nodes to label these nodes (**SVM-X**). Here, we consider SVMs in the binary classification setting (i.e. $y_i \in \{-1, +1\}$). For multiclass problems, we used a one-vs-one strategy [75]. This classifier is considered as a baseline. In practical terms, we use the well-known Liblinear library [76]. Notice that SVM follows an inductive scheme, unlike all other methods. Transductive SVMs [77] were also considered, but their available Matlab implementation was too slow to be included in the present analysis.

5.2.2. Using graph structure only

Three different families of methods using graph structure only are investigated.

For the **bag of path classifier** based on the bag-of-paths group betweenness (**BoP-A**), the betweenness is computed for each class in turn. Then, each unlabeled node is assigned to the class showing the largest value (see Section 4.3.1 for more details).

Then, for the **sum-of-similarities method** based on the **regularized commute time kernel** (**CTK-A**), the classification procedure is the same as BoP-A: the class similarity is computed for each class in turn and each unlabeled node is assigned to the class showing the largest similarity (see Section 4.3.2).

The four **graph embedding techniques** discussed in Section 4.1 are used together with a SVM, without considering any node feature, are also considered. The SVM is trained using a given number of extracted dominant eigenvectors derived from each measure (this number is a parameter to tune). The SVM model is then used to classify the unlabeled nodes. SVMs using Moran's I, Geary's c, local principal component analysis and the bag-of-paths modularity (see Section 4.1) are denoted as **SVM-M-A**, **SVM-G-A**, **SVM-L-A** and **SVM-BoPM-A**, respectively.

5.2.3. Using both information (features on nodes and graph structure)

Here, we investigate the following models.

In the **double kernel SVM** (**DK-SVM-AX**), two kernels are computed, one defined on the graph and the second from the node features $\mathbf{X}_{\text{new}} = [\mathbf{K}_{\text{A}}, \mathbf{K}_{\text{X}}]$ (see Section 4.2.2). A SVM is then used to classify the unlabeled nodes.

Similarly, the **support vector machine using autocovariates** (**ASVM-AX**), autocovariates are added to the node features $\mathbf{X}_{\text{new}} = [\mathbf{X}_{\text{features}}, \mathbf{Ac}]$ (see Section 4.2.1).

On the other hand, the **spatial autoregressive model** (**SAR-AX**) is a spatial extension of the standard regression model (see Section 4.2.3), used to classify the unlabeled nodes.

Moreover, the dominant eigenvectors (this number is a parameter to tune) provided by the four **graph embedding techniques** (Section 4.1) are combined with the node features and then injected into a linear SVM classifier. The new set of feature is therefore $\mathbf{X}_{\text{new}} = [\mathbf{X}_{\text{features}}, \mathbf{X}_{\text{embedding}}]$, where $\mathbf{X}_{\text{embedding}}$ can be obtained using Moran's I, Geary's c, local principal component analysis and the bag-of-paths modularity (see Section 4.1). Those four variants are named **SVM-M-AX**, **SVM-G-AX**, **SVM-L-AX** and **SVM-BoPM-AX**, respectively.

Furthermore, **co-training based on two SVMs** (**SVM-COT-AX**) uses two SVM classifiers, one based on a kernel computed from the features

X and the second based on a graph kernel computed from **A** (see Section 4.4.1). A two-steps procedure is then used to classify the unlabeled nodes.

Finally, the **SVM based on kernel canonical correlation analysis** (**SVM-KCA-AX**) first aligns two kernels, one based on **X** and one based on **A**. Then the two aligned kernels are used together as for DK-SVM-AX.

The considered classifiers, together with their parameters to be tuned, are listed in Table 4.

Table 4: The 16 classifiers, the value range tested for tuning their parameters and the most frequently selected values: Mode is the most selected value across all datasets. Note that p, the number of extracted eigenvector, is given in %: this is the relative number of kept features with respect to the number of node of the graph (different for each dataset).

Classification model	**Use A**	**Use X**	**Acronym**	**Param.**	**Tested values**	**Mode**
Bag of paths betweenness (4.3.1)	yes	no	BoP-A	$\theta > 0$	$10^{[-9,-6,-3,0]}$	10^{-6}(40.2%)
Sum of similarities with the RCT kernel (4.3.2)	yes	no	CTK-A	$\lambda > 0$	$0.2, 0.4, 0.6, 0.8, 1$	0.8(39.4%)
SVM on Moran's extracted features only (4.1.1)	yes	no	SVM-M-A	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	10^{-2}(63.0%)
				$p > 0$	$[5, 10, 20, 35, 50\%]$	5%(74.0%)
SVM on Geary's extracted features only (4.1.2)	yes	no	SVM-G-A	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	10^{-2}(34.8%)
				$p > 0$	$[5, 10, 20, 35, 50\%]$	5%(39.6%)
SVM on LPCA's extracted features only (4.1.3)	yes	no	SVM-L-A	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	10^{2}(47.3%)
				$p > 0$	$[5, 10, 20, 35, 50\%]$	5%(69.5%)
SVM on BoP modularity extracted features (4.1.4)	yes	no	SVM-BoPM-A	$\theta > 0$	$10^{[-9,-6,-3,0]}$	10^{0}(35.2%)
				$C > 0$	$10^{[-6,-3,0,3,6]}$	10^{3}(44.4%)
				$p > 0$	$[5, 10, 20, 35, 50\%]$	5%(72.0%)
SVM on node features only (**baseline**)	no	yes	SVM-X	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	10^{-2}(27.2%)
Spatial autoregressive model (4.2.3)	yes	yes	SAR-AX	*none*	–	-
SVM on Moran and nodes features (4.1.1)	yes	yes	SVM-M-AX	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	10^{2}(26.9%)
				$p > 0$	$[5, 10, 20, 35, 50\%]$	5%(33.3%)
SVM on Geary and nodes features (4.1.2)	yes	yes	SVM-G-AX	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	10^{2}(21.2%)
				$p > 0$	$[5, 10, 20, 35, 50\%]$	5%(31.8%)
SVM on LPCA and nodes features (4.1.3)	yes	yes	SVM-L-AX	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	10^{2}(28.4%)
				$p > 0$	$[5, 10, 20, 35, 50\%]$	5%(41.8%)
SVM on BoP modularity and nodes features (4.1.4)	yes	yes	SVM-BoPM-AX	$\theta > 0$	$10^{[-9,-6,-3,0]}$	10^{0}(27.4%)
				$C > 0$	$10^{[-6,-3,0,3,6]}$	10^{3}(41.0%)
				$p > 0$	$[5, 10, 20, 35, 50\%]$	5%(48.9%)
SVM on autocovatiates and nodes features (4.2.1)	yes	yes	ASVM-AX	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	10^{0}(28.1%)
SVM on a double kernel (4.2.2)	yes	yes	SVM-DK-AX	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	10^{-4}(31.7%)
Co-training based on two SVMs (4.4.1)	yes	yes	SVM-COT-AX	$C > 0$	$10^{[-6,-3,0,3,6]}$	10^{-6}(34.8%)
SVM based on kernel canonical correlation (4.4.2)	yes	yes	SVM-KCA-AX	$C > 0$	$10^{[-6,-3,0,3,6]}$	10^{-6}(44.2%)

5.3. Experimental methodology

The classification accuracy will be reported for a 20% labeling rate, i.e. proportion of nodes for which labels are known. Labels of remaining nodes are deleted during model fitting phase and are used as test data during the assessment phase, where the various classification models predict the most suitable category of each unlabeled node in the test set.

A standard 5-fold nested cross-validation is used for assessing the investigated methods. For each dataset and for each considered feature set, samples (nodes) are randomly assigned into 5 external folds, which defines

Table 5: Time analysis of the 16 classifiers. **Time 251** is the time in seconds required to label DB4, with two classes and 251 nodes (Student vs other, cfr Table 1). **Time 756** is the time in seconds required to label DB5, with two classes and 756 nodes. **Ratio** is computed as **Time 756** divided by **Time 251** (from DB4 to DB5, the number of node is multiplied by 3.012). Computation times are averaged on 10 runs. Param is a reminder about the parameters to be tuned. The quickest methods are indicated in bold.

Acronym	**Param**	**Implementation**	**Time 251**	**Time 756**	**Ratio**
BoP-A	θ	Matlab (not sparse)	0.69s	25.99s	37.47
CTK-A	α	Matlab (sparse)	0.15s	0.59s	**4.06**
SVM-M-A	C,p	Matlab with MEX(C)	2.03s	28.44s	14.01
SVM-G-A	C,p	Matlab with MEX(C)	0.44s	4.66s	10.57
SVM-L-A	C,p	Matlab with MEX(C)	0.53s	6.13s	11.55
SVM-BoPM-A	θ,C,p	Matlab with MEX(C)	2.16s	58.46s	27.03
SVM-X	none	Matlab with MEX(C)	0.10s	0.22s	**2.08**
SAR-AX	none	Lesage's Matlab toolbox	2.96s	43.80s	14.79
SVM-M-AX	C,p	Matlab with MEX(C)	2.00s	29.76s	14.89
SVM-G-AX	C,p	Matlab with MEX(C)	0.47s	5.19s	11.00
SVM-L-AX	C,p	Matlab with MEX(C)	0.53s	6.88s	13.00
SVM-BoPM-AX	θ,C,p	Matlab with MEX(C)	2.18s	60.41s	27.73
ASVM-AX	C	Matlab with MEX(C)	1.50s	7.78s	**5.18**
SVM-DK-AX	C	Matlab with MEX(C)	1.26s	17.00s	13.48
SVM-COT-AX	C	Matlab with MEX(C)	2.43s	34.92s	14.34
SVM-KCA-AX	C	Matlab	4.37s	82.76s	18.95

one *run* of the experimental comparison. Moreover, for each external fold, a 5-fold internal, nested, cross-validation is performed to tune the parameters of the models (see Table 4). The results for one specific run are then computed by taking the average over the 5 external folds. The whole procedure is repeated 5 times to mitigate the effect of lucky/unlucky samples-to-fold assignation, so that 5 runs of the experimental comparison for each dataset and feature set are performed, with different fold assignments.

5.4. Results and discussion

First of all, most frequently selected parameter values are reported on Table 4. We observe that the most selected value for p (the number of eigenvectors extracted for representing the graph structure; see Section 4.1) is actually low. This is good news since efficient eigensystem solvers can be used to compute sequentially the first eigenvectors corresponding to the largest (or smallest) eigenvalues.

The classification accuracy and standard deviation, averaged on the 5 runs, are reported on Tables 6 (for the methods based on both features and the graph structure) and 7 (for the methods based on the graph structure

Table 6: Classification accuracy in percent ± standard deviation, obtained on the 5 runs, the "AX" combined methods (as well as the baseline) and the 10 datasets. Results are reported for the five different feature sets (100F stands for the set of 100 features, and so on). The standard deviation is computed on the 5 folds of the external cross-validation and the 5 independent runs. Best results for each dataset and feature set are highlighted in bold.

		SAR	SVM-G	SVM-M	ASVM	SVM-DK	SVM-BoPM	SVM-L	SVM-COT	SVM-KCA	SVM
		AX	AX	AX	AX	AX	AX	AX	AX	AX	X
DB1	100F	53.2±7.5	84.0±1.4	83.7±1.4	79.4±0.8	**84.9±1.1**	84.1±1.0	83.7±1.3	66.7±1.5	84.1±0.7	83.7±1.5
	50F	66.9±0.9	79.5±1.8	79.5±2.0	79.2±1.9	**80.9±1.6**	79.0±3.2	79.5±2.0	64.4±2.0	78.3±0.9	80.6±0.7
	25F	65.4±3.2	69.8±4.0	68.1±4.4	**74.6±1.3**	73.6±1.5	73.9±2.0	68.6±4.4	57.5±2.0	69.3±1.6	74.6±1.6
	10F	64.4±3.3	63.0±3.8	62.2±4.8	**71.4±3.5**	69.7±3.2	69.8±2.6	62.7±3.9	56.5±1.5	70.3±3.1	70.9±2.1
	5F	62.0±4.2	57.3±5.4	58.1±3.3	**65.8±4.1**	65.2±1.7	60.4±0.9	57.9±2.6	51.7±3.1	65.0±1.2	65.5±0.8
DB2	100F	62.9±1.6	**81.0±0.6**	80.6±0.4	75.1±1.5	79.9±1.4	80.4±1.1	80.4±1.0	60.1±1.2	78.8±1.4	80.8±0.4
	50F	66.5±4.2	76.6±1.4	76.4±2.0	74.4±2.2	76.8±0.9	76.6±1.5	76.7±1.4	58.6±2.5	75.9±0.8	**76.9±0.8**
	25F	66.9±3.6	70.6±2.3	71.0±2.3	73.3±1.1	74.2±2.0	73.4±1.9	71.0±2.5	57.1±1.8	73.0±1.2	**75.5±1.5**
	10F	64.8±5.7	58.9±10.6	57.3±8.2	72.3±2.3	72.4±1.3	72.1±2.3	58.6±9.2	56.6±2.3	72.9±1.5	**74.3±2.1**
	5F	55.7±8.6	56.8±8.1	56.4±7.9	**70.9±1.5**	68.1±2.7	65.7±1.7	57.3±6.8	52.6±1.4	67.6±2.2	66.2±2.0
DB3	100F	64.2±4.4	80.8±1.3	80.7±1.5	80.1±0.7	**81.3±0.7**	81.2±0.9	80.8±1.7	59.0±1.0	80.9±0.6	80.9±1.7
	50F	65.9±3.2	77.2±0.7	77.3±0.8	77.7±1.6	77.7±2.0	78.3±1.1	77.3±0.7	55.5±2.3	77.7±1.0	**78.3±1.3**
	25F	69.1±2.1	73.6±3.2	73.1±2.9	75.7±0.9	76.9±1.2	**78.2±0.7**	73.5±3.2	55.1±1.6	76.8±0.8	77.5±1.0
	10F	63.6±4.4	64.0±8.0	63.7±6.9	**75.7±1.3**	74.9±0.7	74.9±0.8	64.6±7.9	53.8±2.2	74.8±2.1	75.6±1.9
	5F	60.5±6.4	64.3±8.2	65.3±7.4	69.4±1.9	**71.1±0.6**	68.5±3.3	64.1±8.5	51.4±0.4	68.9±0.9	71.0±1.6
DB4	100F	70.5±3.8	83.2±0.7	83.2±0.6	81.1±1.8	**84.3±0.9**	83.4±0.6	83.2±0.9	59.7±1.7	83.3±1.5	83.1±0.7
	50F	72.0±3.8	78.5±1.6	78.5±1.7	79.4±1.0	80.0±2.0	**81.6±1.6**	78.8±1.3	60.7±1.7	79.5±1.2	81.1±2.3
	25F	68.1±3.4	74.3±4.2	74.4±4.1	**79.8±1.9**	78.6±0.9	79.2±0.6	74.1±4.7	60.7±2.1	78.6±1.1	79.4±1.2
	10F	67.7±4.8	64.8±7.7	64.2±7.5	**76.2±1.8**	72.2±0.8	70.2±1.8	65.7±7.0	57.7±0.9	71.4±1.2	71.7±0.7
	5F	61.6±8.8	59.6±7.4	59.4±7.8	**75.5±1.7**	74.4±1.7	73.7±1.7	58.8±7.7	58.2±1.2	73.8±1.1	75.0±0.6
DB5	100F	50.7±12.3	87.8±0.8	87.7±0.7	**92.1±0.4**	88.8±1.0	88.1±0.5	87.9±0.8	91.2±1.2	88.4±0.8	88.5±0.5
	50F	66.2±17.2	87.6±2.2	89.9±1.9	**92.3±0.3**	88.9±0.5	88.9±0.6	88.8±2.1	91.5±1.2	88.8±0.3	89.1±0.7
	25F	80.3±16.6	90.4±1.4	93.7±1.6	**95.3±0.2**	89.1±1.4	91.1±0.9	90.8±1.5	92.1±0.7	89.6±0.6	89.5±0.5
	10F	76.1±9.8	93.1±1.2	94.9±0.7	**95.5±0.4**	89.4±1.2	92.9±0.9	93.7±1.3	91.9±0.7	89.4±1.3	89.5±0.5
	5F	74.7±8.3	93.2±1.1	94.1±1.4	**95.6±1.8**	87.2±0.1	90.1±4.4	94.2±0.8	89.9±1.0	87.3±0.0	87.2±0.4
DB6	100F	72.7±8.1	91.6± 0.4	**93.2±1.1**	92.6±0.4	92.1±0.4	92.5±0.5	91.8±0.9	93.0±0.5	92.3±0.3	91.4±0.2
	50F	78.9±14.4	89.6±2.1	**95.0±1.0**	94.8±1.1	91.5±0.3	92.9±0.8	93.1±0.8	92.2±0.5	91.4±0.5	91.2±0.3
	25F	86.3±8.8	91.7± 1.2	**96.6±1.0**	96.0±1.0	92.0±0.2	94.7±1.0	95.2±2.2	92.5±0.4	91.8±0.4	91.8±0.5
	10F	78.3±9.9	93.5± 0.7	**97.6±0.4**	96.6±1.2	89.7±3.1	95.6±1.4	96.6±0.6	94.0±0.5	90.9±2.5	92.0±0.1
	5F	78.1±10.0	93.3±0.8	**97.6±0.4**	96.6±1.3	88.5±7.6	95.0±1.4	96.5±1.1	92.3±0.5	88.7±7.5	92.2±0.1
DB7	100F	56.4±11.2	74.9±0.8	74.9±1.2	**79.8±0.8**	76.1±0.7	74.1±1.2	75.3±1.1	79.8±1.0	75.4±0.7	74.7±1.1
	50F	61.5±11.3	77.5±1.5	76.0±1.7	**80.5±0.9**	79.6±0.6	77.7±1.0	78.4±1.2	81.1±0.5	78.9±1.1	78.0±1.1
	25F	69.7±8.0	79.3±0.5	79.9±0.9	**81.6±0.4**	80.3±0.4	79.1±0.4	80.0±0.6	81.3±0.4	80.8±0.3	78.9±2.4
	10F	66.0±10.6	78.7±2.1	80.7±1.2	**81.2±0.1**	80.2±0.8	79.3±1.3	**81.2±0.9**	80.2±0.9	80.6±0.3	80.5±0.3
	5F	66.2±10.3	79.9±0.8	79.8±1.1	**80.9±0.6**	80.5±0.4	77.1±2.9	**80.9±0.8**	77.9±0.4	80.4±0.2	78.5±3.9
DB8	100F	61.2±4.9	70.5±0.6	70.5±0.6	66.1±0.6	70.4±0.3	**70.6±0.4**	70.5±0.6	68.1±0.3	70.4±0.3	70.5±0.6
	50F	64.5±1.6	62.9±4.2	64.1±3.1	66.1±0.7	68.5±0.3	68.7±0.5	62.8±4.3	68.6±0.8	68.6±0.2	**68.8±0.5**
	25F	56.0±7.9	59.9±3.1	65.4±0.8	66.0±1.2	**70.2±0.3**	66.0±0.4	62.3±2.0	70.0±0.8	68.1±0.6	66.7±0.4
	10F	44.2±11.2	57.1±3.1	66.3±1.9	62.9±0.8	**72.7±0.4**	63.6±1.0	61.4±2.7	65.6±1.5	65.7±1.3	59.4±0.4
	5F	42.7±12.0	57.1±2.5	67.5±0.6	61.9±0.9	**72.0±1.4**	63.0±2.1	61.9±1.2	65.8±4.4	68.0±0.9	53.9±0.6
DB9	100F	**77.5±1.0**	71.4±0.4	71.3±0.4	70.6±0.9	71.3±0.7	71.7±0.7	71.2±0.5	75.7±0.6	67.5±0.9	71.4±0.4
	50F	64.3±5.7	66.0±2.5	72.1±1.2	**76.3±0.3**	69.4±0.2	71.9±1.5	73.0±1.8	76.1±0.6	68.4±0.5	68.6±0.1
	25F	53.6±10.1	67.5±3.9	73.8±2.5	**77.0±0.3**	74.0±0.3	76.2±0.3	73.9±2.8	74.5±0.4	70.1±0.1	64.2±0.1
	10F	42.9±9.8	72.1±3.1	76.4±1.8	74.9±0.5	76.6±0.3	**77.3±0.9**	76.8±1.9	69.7±0.6	67.1±0.1	56.3±0.2
	5F	37.1±6.8	73.2±2.5	76.1±1.3	71.8±0.9	78.0±0.3	**80.3±1.2**	76.3±1.6	65.6±1.1	67.0±0.3	42.3±1.0
DB10	100F	32.1±5.2	54.6±0.5	54.4±0.5	44.5±1.2	54.2±0.6	**56.2±0.5**	54.9±0.2	49.6±0.2	52.1±0.3	54.6±0.5
	50F	35.6±7.7	45.9±1.7	46.7±0.9	37.1±0.4	41.9±0.3	45.0±0.6	**48.6±0.7**	40.4±0.5	40.8±0.5	40.2±0.4
	25F	35.4±6.1	43.4±2.0	45.5±2.4	32.7±0.9	36.7±0.6	41.6±1.4	**46.2±2.4**	34.7±0.5	35.5±0.2	34.2±0.2
	10F	29.0±2.5	39.2±2.1	41.9±1.9	28.2±0.9	31.8±0.3	42.2±0.4	**42.3±2.0**	32.9±0.4	31.1±0.1	30.8±0.1
	5F	21.2±1.8	35.9±2.0	38.7±2.6	25.1±0.8	25.8±0.2	**42.0±0.8**	39.6±2.5	31.6±0.9	25.8±0.2	25.2±0.3

Table 7: Classification accuracy in percent ± standard deviation, obtained on the 5 runs, the 6 "A" methods (and baseline) and the 10 datasets. Baseline results are reported for 100F feature sets. The standard deviation is computed on the 5 folds of the external cross-validation and the 5 independent runs. Best results for each dataset and feature set are highlighted in bold.

	BoP	CTK	SVM-M	SVM-G	SVM-L	SVM-BoPM	SVM (100F)
	A	A	A	A	A	A	X
DB1	54.5±1.7	54.2±0.9	46.3±5.1	43.4±2.7	40.8±2.3	43.5±0.8	**83.7±1.5**
DB2	41.8±4.2	42.4±1.1	33.1±1.7	33.3±2.2	33.1±3.5	32.3±3.4	**80.8±0.4**
DB3	48.4±0.6	46.7±1.3	47.0±4.9	44.1±2.8	40.4±1.5	39.3±2.3	**80.9±1.7**
DB4	45.7±1.5	42.9±3.2	40.0±1.5	40.0±2.3	42.0±1.6	42.6±3.3	**83.1±0.7**
DB5	**96.8±0.1**	96.3±0.1	95.4±0.5	89.8±4.1	90.9±0.8	91.7±0.6	88.5±0.5
DB6	**98.6±0.1**	98.4±0.1	95.1±0.3	93.8±0.4	95.9±1.0	94.4±1.1	91.4±0.2
DB7	82.5±0.2	**82.9±0.5**	81.6±0.8	78.5±0.5	79.1±1.4	79.1±0.9	74.7±1.1
DB8	69.9±0.6	**70.5±0.4**	55.9±0.4	68.1±0.3	62.4±0.9	67.5±1.2	**70.5±0.6**
DB9	78.1±0.2	**81.7±0.2**	74.5±0.6	75.6±0.7	76.6±0.4	80.3±0.3	71.4±0.4
DB10	35.3±0.3	36.4±0.2	30.8±0.6	35.0±0.2	34.4±0.7	14.9±0.3	**54.6±0.5**

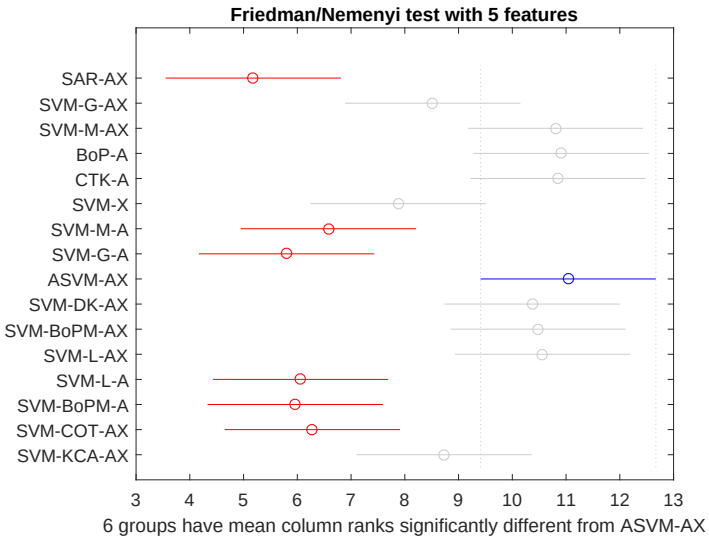

Figure 2: Mean rank (circles) and critical difference (plain line) of the Friedman/Nemenyi test, over 5 runs and all datasets, obtained on partially labeled graphs. The blue method has the best mean rank and is statistically better than red methods. Labeling rate is 20% and the critical difference is 3.26. This figure shows the results when only 5 node features are considered (5F datasets).

only), for the 10 different datasets and the 5 sets of features. Bold values indicate the best performance on each row. Recall that the BoP-A, CTK-A, SVM-M-A, SVM-G-A, and SVM-L-A methods do not depend on the node

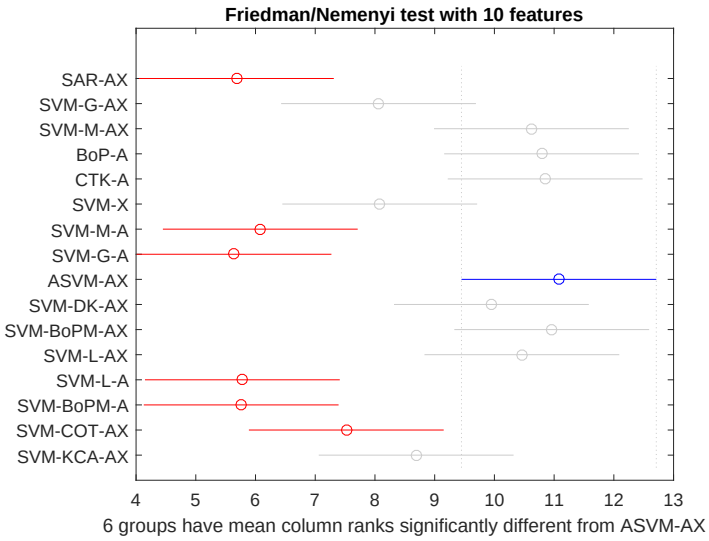

Figure 3: Friedman/Nemenyi test considering 10 node features (10F datasets); see Figure 2 for details. The critical difference is 3.26.

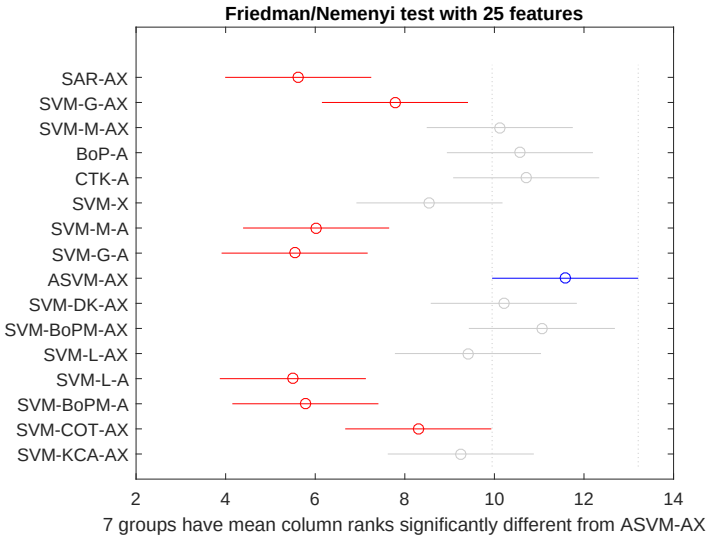

Figure 4: Friedman/Nemenyi test considering 25 node features (25F datasets); see Figure 2 for details. The critical difference is 3.26.

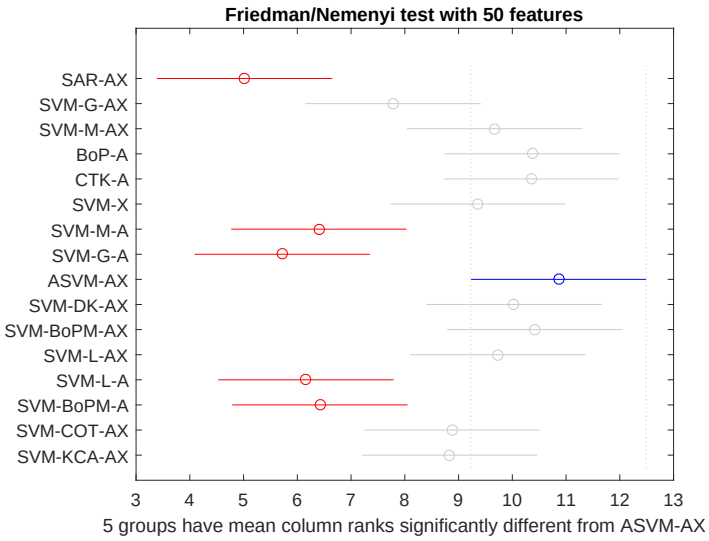

Figure 5: Friedman/Nemenyi test considering 50 node features (50F datasets); see Figure 2 for details. The critical difference is 3.26.

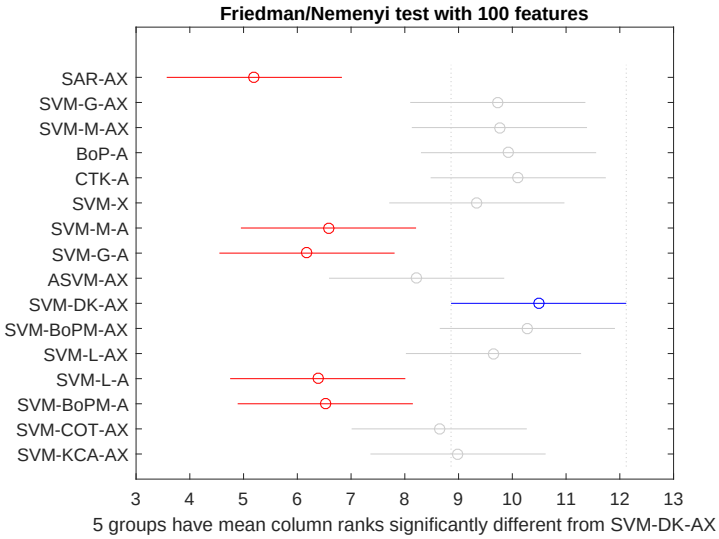

Figure 6: Friedman/Nemenyi test considering 100 node features (100F datasets); see Figure 2 for details. The critical difference is 3.26.

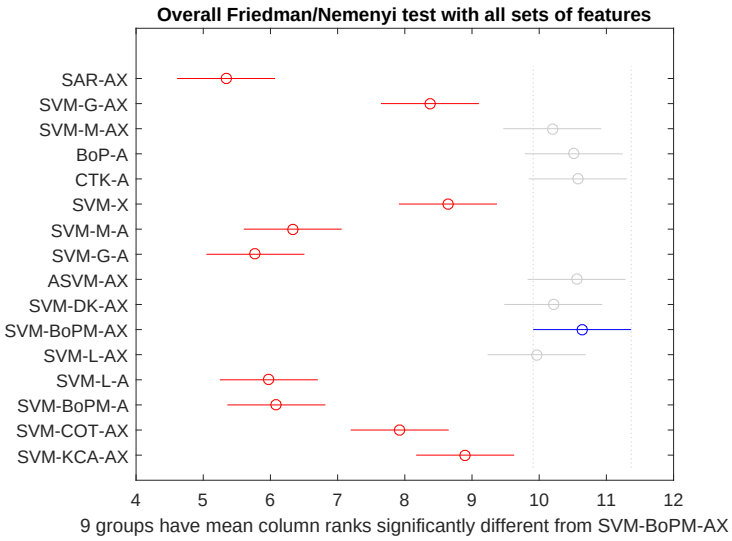

Figure 7: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F , 50F, 100F). The critical difference is 1.46; see Figure 2 for details.

features as they are based on the graph structure only.

Moreover, the different classifiers are compared across datasets through a Friedman test and a Nemenyi post-hoc test [78]. The Friedman test is a non-parametric equivalent of the repeated-measures ANOVA. It ranks the methods for each dataset separately, the best algorithm getting the rank 1, the second best rank 2, etc. Once the null hypothesis (the mean ranking of all methods is equal, meaning all classifiers are equivalent) is rejected with p-value < 0.05, the (non parametric) post-hoc Nemenyi test is then computed. Notice that all Friedman tests were found to be positive in our experiments. The Nemenyi test determines whether or not each method is significantly better (p-value less than 0.05 based on the 5 runs, the considered datasets and feature sets) than another.

This is reported, for each feature set in turn (5F, 10F, ..., 100F), and thus increasing information available on the nodes, in Figures 2 to 6, while the result of an overall test based on all the features sets and datasets is shown in Figure 7.

5.4.1. Overall performances on all datasets and all node feature sets

From Tables 6 to 7 and Figure 7, overall best performances on all dataset and all node features sets are often obtained either by a SVM based on node

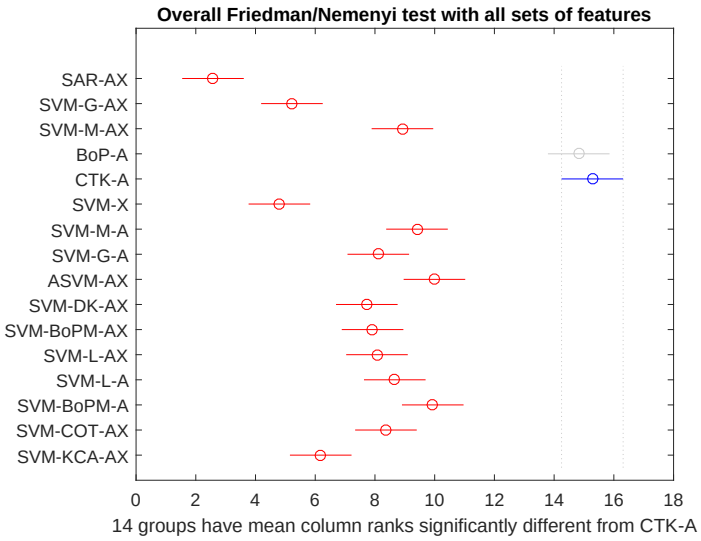

Figure 8: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F, 50F, 100F), but computed only on datasets DB5 to DB9 (driven by graph structure, A); see Figure 2 for details. The critical difference is 2.06.

features combined with new features derived from the graph structure (Subsection 4.1), or, unexpectedly, by the CTK-A sum-of-similarities method (using graph structure only; see Subsection 4.3.2), which performs quite well on datasets five to nine. The BoP-A node betweenness (using graph structure only, see Subsection 4.3.1) is also competitive and achieves results similar to the sum-of-similarities CTK-A method (as already observed in [30]).

However, the best method among the graph structure plus node features SVM is not straightforward to determine (see Figure 7). From Figures 2 to 6, the main trend is that the performance decreases when the number of features decreases, which seems normal.

However, this trend is not always observed; for example, with the SVM-M-AX method (SVM with features extracted from Moran's index and features on nodes, see Subsection 4.1.1) and dataset DB5, the performances rise when the number of features decreases. This can be explained by observing that each dataset labeling can be better explained in terms of its graph structure (graph-driven datasets, DB5 to DB9), or by its node features (features-driven datasets, DB1 to DB4, plus DB10).

To confirm this fact, the network structure autocorrelation was computed

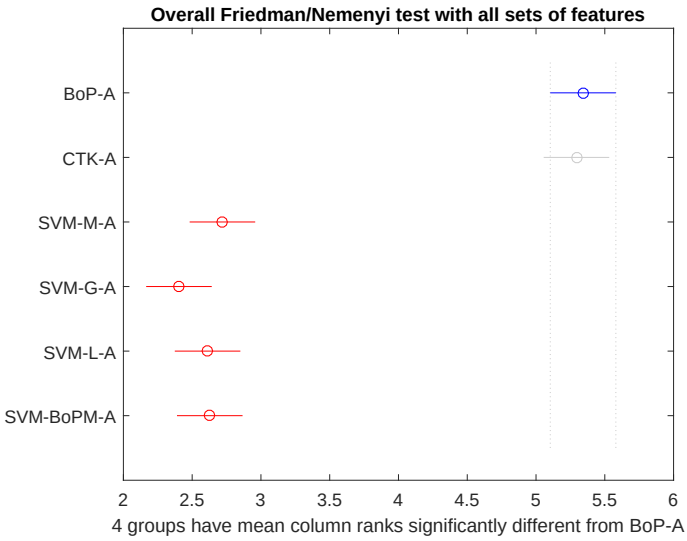

Figure 9: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F, 50F, 100F), for methods based on graph information alone (A); see Figure 2 for details. The critical difference is 0.48.

for each class (i.e., for each $\mathbf{y}^c$) and the average is reported for each dataset. This measure quantifies to which extent the target variable is correlated with its neighboring nodes. The values are reported on Table 8 for Moran's I, Geary's c and the LPCA contiguity ratio (see Subsection 4.1.1). For Moran's I, high values (large autocorrelation) indicate that the graph structure is highly informative. This is the opposite for Geary and LPCA, as small values correspond to a large autocorrelation. It can be observed that our hypothesis is clearly confirmed.

Nevertheless, from Tables 6 and 7 and Figure 7, the best overall performing methods combining node features and graph structure are (excluding the methods based on the graph alone, BoP-A and CTK-A), SVM-BoPM-AX (SVM with bag-of-paths modularity, see Subsection 4.1.4) and ASVM-AX (SVM based on autocovariates, see Subsection 4.2.1). While performing better on our datasets (their mean rank is slightly higher), they are however not statistically different from SVM-M-AX, SVM-L-AX and SVM-DK-AX.

Notice also that, from Figure 7, if we look at the performances obtained by a baseline linear SVM based on node features only (SVM-X), we clearly observe that integrating the information extracted from the graph structure improves the results. Therefore, it seems to be a good idea to consider

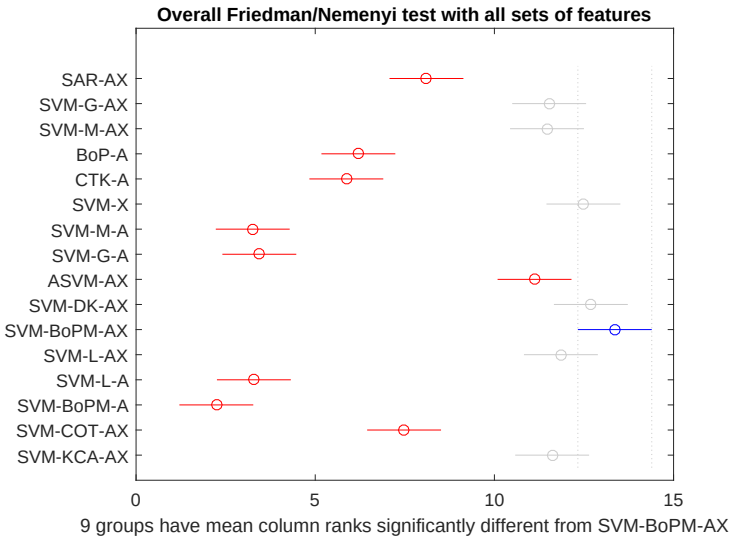

Figure 10: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F, 50F, 100F), but computed only on datasets DB1 to DB4 and DB10 (driven by node features, X); see Figure 2 for details. The critical difference is 2.06.

collecting link information, which could improve the classification results.

5.4.2. Exploiting either the structure of the graph or the node features alone

Obviously, as already mentioned, datasets DB5 to DB9 are graph-driven, which explains the good performances of the sum-of-similarities CTK-A and BoP-A on these data. For these datasets, the features on the nodes do not help much for predicting the class label, as observed when looking to Figure 8 where results are displayed only on these datasets. It also explains the behavior of method SVM-M-AX on dataset DB5, among others.

In this case, the best performing methods are the sum-of-similarities CTK-A and the bag-of-paths betweenness BoP-A (see Subsection 4.3). This is clearly confirmed by displaying the results of the methods based on the graph structure only in Figure 9 and the results obtained on the graph-driven datasets in Figure 8. Interestingly, in this setting, these two methods ignoring the node features (CTK-A and BoP-A) are outperforming the SVM-based methods.

Conversely, on the node features-driven datasets (DB1 to DB4 and DB10; results displayed in Figure 10 and Tables 6 and 7), all SVM methods based on node features (and graph structure) perform well while methods based

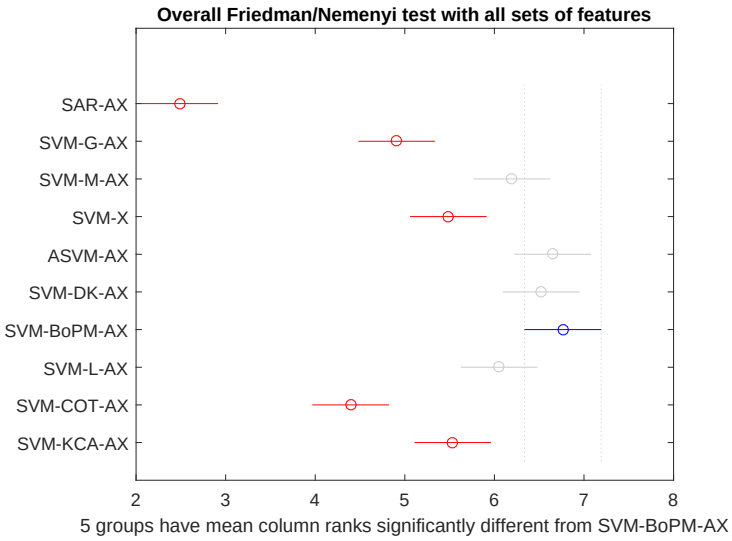

Figure 11: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F, 50F, 100F), performed only on methods combining graph structure and node features information (AX, plus simple SVM-X as baseline). See Figure 2 for details. The critical difference is 0.86.

on the graph structure only obtain much worse results, as expected. In this setting, the situation is rather similar to the overall results case (see Figure 7). The two best techniques are SVM-BoPM-AX (SVM with bag-of-paths modularity, see Subsection 4.1.4) and SVM-DK-AX (SVM based on a double kernel, see Subsection 4.2.2). However, this time, these two first ranked methods are not significantly better than the simple linear SVM based on features only (SVM-X), as shown in Figure 10. Thus, for the node features-driven datasets, the graph structure does not bring much additional information.

From another point of view, Figure 11 takes into account all datasets and compares only the methods combining node features and graph structure. In this setting, the best ranked methods are again SVM-BoPM-AX, ASVM-AX and SVM-DK-AX which are now significantly better than the baseline SVM-X (less methods are compared on more datasets).

Notice that SVM-KCA-AX performs better than SVM-COT-AX, but is not significantly different from SVM-DK-AX, although SVM-DK-AX's mean rank is higher than SVM-KCA-AX's mean rank.

Finally, the worst performing method is always SAR-AX, except if only

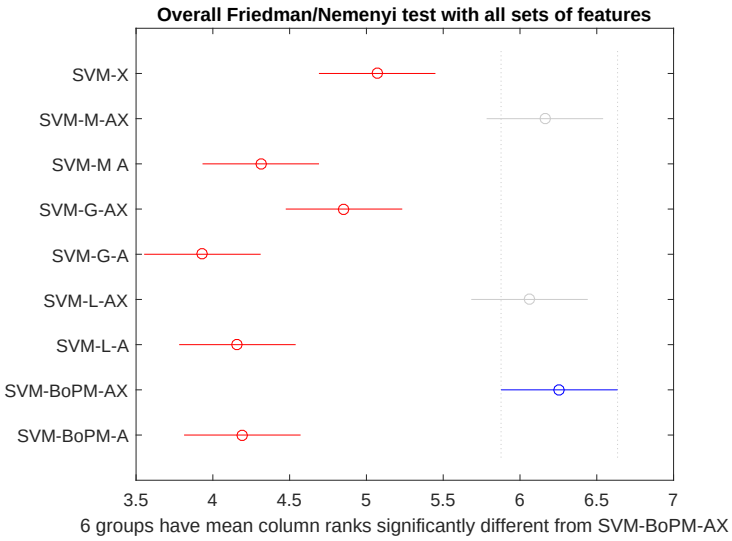

Figure 12: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F, 50F, 100F), only considering methods based on a graph embedding (plus regular linear SVM for comparison). See Figure 2 for details. The critical difference is 0.76.

features-driven datasets are considered. Even in this case, SAR-AX outperforms only graph-based methods (see Figure 10).

5.4.3. Comparison of graph embedding methods

Concerning the embedding methods described in Subsection 4.1, we can conclude that Geary's index (SVM-G-A and SVM-G-AX) should be avoided by preferring the bag-of-paths modularity (SVM-BoP-AX), Moran's index (SVM-M-AX) or Local Principal Component Analysis (SVM-L-AX). This is clearly observable when displaying only the results of the methods combining node features and graph structure in Figure 11. This result is further confirmed when comparing only the methods based on a graph embedding in Figure 12.

5.4.4. Discussion about time complexity

Table 5 shows a comparison of computation time for the 16 classifiers. Notice that, from columns **Time 251** to **Time 756**, the number of nodes is multiplied by 3.012. The fastest methods are CTK-A (which has an efficient sparse implementation, quasi-linear in number of samples/modes) for graph-based classifiers and ASVM-AX for classifiers combining features and

Table 8: Mean autocorrelation of class membership computed on all the investigated datasets. For Moran's I, a high value corresponds to a large autocorrelation. Conversely, for Geary's c and LPCA, a small value implies a large autocorrelation. For each autocorrelation measure, $-$ indicates the presence of negative autocorrelation, a value close to **0** indicates a lack of structural association and $+$ indicates presence of positive autocorrelation. See Subsection 4.1 and [45] for details. Datasets can be divided into two groups (more driven by graph structure (A) or by features on nodes (X)), according to these measures. I_0 is equal to $-1/(n-1) \approx 0$, where n is the number of nodes.

A-driven	$+$	**0**	$-$	DB 5	DB 6	DB 7	DB 8	DB 9
Moran's I	$> I_0$	$= I_0$	$< I_0$	1.27	1.09	0.66	0.53	0.79
Geary's c	< 1	1	> 1	0.09	0.09	0.33	0.19	0.12
LPCA c. ratio [2]	cr_0	$> cr_0$		0.20	0.13	0.58	0.67	0.26
X-driven	$+$	**0**	$-$	DB 1	DB 2	DB 3	DB 4	DB 10
Moran's I	$> I_0$	$= I_0$	$< I_0$	-0.22	-0.12	-0.15	-0.06	0.15
Geary's c	< 1	1	> 1	0.78	0.59	0.63	0.57	0.43
LPCA c. ratio [2]	cr_0	$> cr_0$		2.54	2.10	1.86	1.90	0.82

structural information. Interestingly, those two classifiers actually achieved good overall results previously. Notice that BoP-based methods are the slowest due to a full matrix inversion ($O(n^3)$). Most other methods exhibit a nearly quadratic behavior. Finally, notice that most SVM methods use the Liblinear library [76], allowing Matlab to execute C code through a MEX file.

5.4.5. Summary of main findings

To summarize, the experiments lead to the following conclusions: The best performing methods are highly dependent on the dataset. We observed (see Table 8) that, quite naturally, some datasets are more graph-driven in the sense that the network structure conveys important information for predicting the class labels, while other datasets are more node features-driven and, in this case, the graph structure does not help much. However, it is probably a good idea to take into consideration information about the graph structure, because this additional information can improve significantly the results, depending on the dataset (see Figures 11 and 12).

If we consider the graph structure alone, the two best investigated methods are the sum-of-similarities (CTK-A) and the bag-of-paths betweenness

[2]LPCA contiguity ratio is positive and lower-bounded by $cr_0 = 1 - \sqrt{\lambda_{\max}}$ (which tends to be close to zero) where $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{A}$. The upper bound is unknown [51].

(BoP-A, see Subsection 4.3). They clearly outperform the graph embedding methods, but also the SVMs on some datasets. This is confirmed by a paired signed Wilcoxon test: BoP-A and CTK-A outperform SVM-X at $p < 10e^{-5}$.

When, in addition, informative features on nodes are available, it is worth considering combining the information, and, in this context, we found that the best performing methods are SVM-BoPM-AX (SVM with bag-of-paths modularity, see Subsection 4.1.4), ASVM-AX (SVM based on autocovariates, see Subsection 4.2.1) and SVM-DK-AX (SVM based on a double kernel, see Subsection 4.2.2) (see Figure 11). Taking the graph structure into account improves the results over a baseline SVM considering node features only. This is confirmed (but only at $p < 0.05$) for the two first methods by a paired signed Wilcoxon test: SVM-BoPM-AX outperforms SVM-X with $p = 0.042$ and ASVM-AX outperforms SVM-X with $p = 0.029$. On the contrary, the p-value for SVM-DK-AX against SVM-X is only 0.182.

6. Conclusion

This work considered a data structure made of a graph and plain features on nodes of the graph. In this context, 16 semi-supervised classification methods were investigated to compare the feature-based approach, the graph structure-based approach, and the dual approach combining both information sources. It appears that the best results are often obtained either by a SVM method (the considered baseline classifier) based on plain node features combined to a given number of new features derived from the graph structure (namely from the BoP modularity or autocovariates), or by the sum-of-similarities and the bag-of-paths modularity method, based on the graph structure only, which perform well on some datasets for which the graph structure carries important class information.

Indeed, we observed empirically that some datasets can be better explained by their graph structure (graph-driven datasets), or by their node features (features-driven datasets). Consequently, neither the graph-derived features alone or the plain features alone is sufficient to obtain optimal performances. In other words, in some situations, standard feature-based classification results can be improved significantly by integrating information from the graph structure. In particular, the most effective methods were based on bag-of-paths modularity (SVM-BoPM-AX), autocovariates (ASVM-AX) or a double kernel (SVM-DK-AX).

The take-away message can be summarize as follows: if the dataset is graph-driven, a simple sum-of-similarities or a bag-of-paths betweenness are sufficient, but this is not the case if the features on the nodes are (more)

informative. In both cases, SVM-BoPM-AX, ASVM-AX, SVM-DK-AX still ensured good overall performances, as shown on the investigated datasets.

A key point is therefore to determine *a priori* if a given dataset is graph-driven or features-driven. In this paper we proposed to use some well-known spatial autocorrelation indexes to tackle this issue. Further investigations will be carried in that direction. In particular, how can we automatically infer properties of a new dataset (graph-driven or features-driven) if all class labels are not known? Can we rely on measuring autocorrelation based on features?

Finally, the present work does not analyze the scalability of the methods, this is also left for further work.

Acknowledgement

Acknowledgement: This work was partially supported by the Elis-IT project funded by the "Région wallonne" and the Brufence project supported by INNOVIRIS ("Région bruxelloise"), Belgium. We thank this institution for giving us the opportunity to conduct both fundamental and applied research.

References

[1] S. Sun, A survey of multi-view machine learning, Neural Computing & Applications 23 (2013) 2031–2038.

[2] J. Zhao, X. Xie, X. Xu, S. Sun, Multi-view learning overview: Recent progress and new challenges, Information Fusion 38 (C) (2017) 43–54.

[3] F. Fouss, M. Saerens, Yet another method for combining classifiers outputs: A maximum entropy approach, in: Proceedings of the 5th International Workshop on Multiple Classifier Systems (MCS 2004), Lecture Notes in Computer Science, Vol. 3077, Springer-Verlag, 2004, pp. 82–91.

[4] L. Kuncheva, Combining pattern classifiers: methods and algorithms, Wiley, 2004.

[5] R. M. Cooke, Experts in uncertainty, Oxford University Press, 1991.

[6] R. A. Jacobs, Methods for combining experts' probability assessments, Neural Computation 7 (1995) 867–888.

[7] D. Chen, X. Cheng, An asymptotic analysis of some expert fusion methods, Pattern Recognition Letters 22 (2001) 901–904.

[8] J. Kittler, F. M. Alkoot, Sum versus vote fusion in multiple classifier systems, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (1) (2003) 110–115.

[9] F. Lad, Operational subjective statistical methods, John Wiley & Sons, 1996.

[10] G. J. Klir, T. A. Folger, Fuzzy sets, uncertainty, and information, Prentice-Hall, 1988.

[11] D. Dubois, M. Grabisch, H. Prade, P. Smets, Assessing the value of a candidate: Comparing belief function and possibility theories, in: Proceedings of the Fifteenth international conference on Uncertainty in Artificial Intelligence, 1999, pp. 170–177.

[12] C. Merz, Using correspondence analysis to combine classifiers, Machine Learning 36 (1999) 226–239.

[13] W. B. Levy, H. Delic, Maximum entropy aggregation of individual opinions, IEEE Transactions on Systems, Man and Cybernetics 24 (4) (1994) 606–613.

[14] I. J. Myung, S. Ramamoorti, J. Andrew D. Bailey, Maximum entropy aggregation of expert predictions, Management Science 42 (10) (1996) 1420–1436.

[15] S. Abney, Semisupervised learning for computational linguistics, Chapman and Hall/CRC, 2008.

[16] O. Chapelle, B. Scholkopf, A. Zien (editors), Semi-supervised learning, MIT Press, 2006.

[17] F. Fouss, M. Saerens, M. Shimbo, Algorithms and models for network data and link analysis, Cambridge University Press, 2016.

[18] T. Hofmann, B. Schölkopf, A. J. Smola, Kernel methods in machine learning, The Annals of Statistics 36 (3) (2008) 1171–1220.

[19] E. D. Kolaczyk, Statistical analysis of network data: methods and models, Springer, 2009.

[20] S. A. Macskassy, F. Provost, Classification in networked data: a toolkit and a univariate case study, Journal of Machine Learning Research 8 (2007) 935–983.

[21] T. Silva, L. Zhao, Machine learning in complex networks, Springer, 2016.

[22] A. Subramanya, P. Pratim Talukdar, Graph-based semi-supervised learning, Morgan & Claypool Publishers, 2014.

[23] X. Zhu, Semi-supervised learning literature survey, unpublished manuscript (available at http://pages.cs.wisc.edu/ jerryzhu/research/ssl/semireview.html) (2008).

[24] X. Zhu, A. Goldberg, Introduction to semi-supervised learning, Morgan & Claypool Publishers, 2009.

[25] S. Hill, F. Provost, C. Volinsky, Network-based marketing: Identifying likely adopters via consumer networks, Statistical Science 21 (2) (2006) 256–276.

[26] F. R. Chung, Spectral graph theory, American Mathematical Society, 1997.

[27] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from examples, Journal of Machine Learning Research 7 (2006) 2399–2434.

[28] X. He, Laplacian regularized d-optimal design for active learning and its application to image retrieval, IEEE Transactions on Image Processing 19 (1) (2010) 254–263.

[29] S. Chakrabarti, B. Dom, P. Indyk, Enhanced hypertext categorization using hyperlinks, in: Proceedings of the ACM International Conference on Management of Data (SIGMOD 1998), 1998, pp. 307–318.

[30] B. Lebichot, I. Kivimaki, K. Françoisse, M. Saerens, Semi-supervised classification through the bag-of-paths group betweenness, IEEE Transactions on Neural Networks and Learning Systems 25 (2014) 1173–1186.

[31] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT' 98, ACM, New York, NY, USA, 1998, pp. 92–100.

[32] D. Borcard, P. Legendre, All-scale spatial analysis of ecological data by means of principal coordinates of neighbour matrices, Ecological Modelling 153 (1-2) (2002) 51–68.

[33] S. Dray, P. Legendre, P. Peres-Neto, Spatial modelling: a comprehensive framework for principal coordinate analysis of neighbour matrices, Ecological Modelling 196 (3-4) (2006) 483–493.

[34] A. Meot, D. Chessel, R. Sabatier, Operateurs de voisinage et analyse des donnees spatio-temporelles (in french), in: D. Lebreton, B. Asselain (Eds.), Biometrie et environnement, Masson, 1993, pp. 45–72.

[35] L. Tang, H. Liu, Relational learning via latent social dimensions, in: Proceedings of the ACM conference on Knowledge Discovery and Data Mining (KDD 2009), 2009, pp. 817–826.

[36] L. Tang, H. Liu, Scalable learning of collective behavior based on sparse social dimensions, in: Proceedings of the ACM conference on Information and Knowledge Management (CIKM 2009), 2009, pp. 1107–1116.

[37] L. Tang, H. Liu, Toward predicting collective behavior via social dimension extraction, IEEE Intelligent Systems 25 (4) (2010) 19–25.

[38] D. Zhang, R. Mao, Classifying networked entities with modularity kernels, in: Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008), ACM, 2008, pp. 113–122.

[39] R. Haining, Spatial data analysis, Cambridge University Press, 2003.

[40] D. Pfeiffer, T. Robinson, M. Stevenson, K. Stevens, D. Rogers, A. Clements, Spatial analysis in epidemiology, Oxford University Press, 2008.

[41] T. Waldhor, Moran's spatial autocorrelation coefficient, in: Encyclopedia of Statistical Sciences, 2nd ed. (S. Kotz, N. Balakrishnana, C. Read, B. Vidakovic and N. Johnson, editors), Vol. 12, Wiley, 2006, pp. 7875–7878.

[42] L. Waller, C. Gotway, Applied spatial statistics for public health data, Wiley, 2004.

[43] P. Moran, Random associations on a lattice, in: Proceedings of the Cambridge Philosophical Society, 1947, pp. 321–328.

[44] P. Moran, The interpretation of statistical maps, Journal of the Royal Statistical Society, Series B, 10 (1948) 243–251.

[45] P. de Jong, C. Sprenger, F. van Veen, On extreme values of moran's i and geary's c, Geographical Analysis 16 (1) (1984) 17–24.

[46] K. V. Mardia, J. T. Kent, J. M. Bibby, Multivariate analysis, Academic Press, 1979.

[47] R. C. Geary, The contiguity ratio and statistical mapping, The Incorporated Statistician 5 (3) (1954) 115–146.

[48] U. von Luxburg, A tutorial on spectral clustering, Statistics and Computing 17 (4) (2007) 395–416.

[49] M. Newman, Networks: an introduction, Oxford University Press, 2010.

[50] H. Benali, B. Escofier, Analyse factorielle lissee et analyse des differences locales, Revue de Statistique Appliquee 38 (2) (1990) 55–76.

[51] L. Lebart, Contiguity analysis and classification, in: W. Gaul, O. Opitz, M. Schader (Eds.), Data Analysis, Studies in classification, data analysis, and knowledge organization, Springer, 2000, pp. 233–243.

[52] M. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical Review E 69 (2004) 026113.

[53] M. Newman, Modularity and community structure in networks, in: Proceedings of the National Academy of Sciences (USA), Vol. 103, 2006, pp. 8577–8582.

[54] R. Devooght, A. Mantrach, I. Kivimaki, H. Bersini, A. Jaimes, M. Saerens, Random walks based modularity: Application to semi-supervised learning, in: Proceedings of the 23rd International Conference on World Wide Web, WWW '14, 2014, pp. 213–224.

[55] J. E. Besag, Nearest-neighbour systems and the auto-logistic model for binary data, Journal of the Royal Statistical Society. Series B (Methodological) 34 (1) (1972) 75–83.

[56] N. H. Augustin, M. A. Mugglestone, S. T. Buckland, An autologistic model for the spatial distribution of wildlife, Journal of Applied Ecology 33 (2) (1996) 339–347.

[57] N. H. Augustin, M. A. Mugglestone, S. T. Buckland, The role of simulation in modelling spatially correlated data, Environmetrics 9 (2) (1998) 175–196.

[58] Q. Lu, L. Getoor, Link-based classification, in: Proceedings of the 20th International Conference on Machine Learning (ICML 2003), 2001, pp. 496–503.

[59] Y. Pawitan, In all likelihood: statistical modelling and inference using likelihood, Oxford University Press, 2001.

[60] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the em algorithm (with discussion), Journal of the Royal Statistical Society B 39 (1) (1977) 1–38.

[61] G. McLachlan, T. Krishnan, The EM algorithm and extensions, 2nd ed, Wiley, 2008.

[62] V. Roth, Probabilistic discriminative kernel classifiers for multi-class problems, in: B. Radig, S. Florczyk (Eds.), Pattern Recognition: Proceedings of the 23rd DAGM Symposium, Vol. 2191 of Lecture Notes in Computer Science, Springer, 2001, pp. 246–253.

[63] B. Scholkopf, A. Smola, Learning with kernels, The MIT Press, 2002.

[64] J. Shawe-Taylor, N. Cristianini, Kernel methods for pattern analysis, Cambridge University Press, 2004.

[65] F. Fouss, K. Francoisse, L. Yen, A. Pirotte, M. Saerens, An experimental investigation of kernels on a graph on collaborative recommendation and semisupervised classification, Neural Networks 31 (2012) 53–72.

[66] T. Gartner, Kernels for structured data, World Scientific Publishing, 2008.

[67] J. LeSage, R. K. Pace, Introduction to spatial econometrics, Chapman & Hall, 2009.

[68] A. Mantrach, N. van Zeebroeck, P. Francq, M. Shimbo, H. Bersini, M. Saerens, Semi-supervised classification and betweenness computation on large, sparse, directed graphs, Pattern Recognition 44 (6) (2011) 1212–1224.

[69] K. Francoisse, I. Kivimaki, A. Mantrach, F. Rossi, M. Saerens, A bag-of-paths framework for network data analysis, Neural Networks 90 (2017) 90–111.

[70] M. Saerens, Y. Achbany, F. Fouss, L. Yen, Randomized shortest-path problems: Two related models, Neural Computation 21 (8) (2009) 2363–2404.

[71] D. Zhou, O. Bousquet, T. Lal, J. Weston, B. Scholkopf, Learning with local and global consistency, in: Proceedings of the Neural Information Processing Systems Conference (NIPS 2003), 2003, pp. 237–244.

[72] D. R. Hardoon, S. R. Szedmak, J. R. Shawe-taylor, Canonical correlation analysis: An overview with application to learning methods, Neural Comput. 16 (12) (2004) 2639–2664.

[73] S. Prithviraj, G. Galileo, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad, Collective classification in network data, AI Magazine 29 (3) (2008) 93–106.

[74] J. McAuley, J. Leskovec, Learning to discover social circles in ego networks, Advances in Neural Information Processing Systems (NIPS).

[75] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, Transaction on Neural Network 13 (2) (2002) 415–425.

[76] R. Fan, K. Chang, C. Hsieh, X. Wang, C. Lin, LIBLINEAR: A library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.

[77] A. Gammerman, V. Vapnik, V. Vowk, Learning by tranduction, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Wisconsin, 1998, pp. 273–297.

[78] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.