Chapter 5
Network-Based Supervised Learning

Abstract In this chapter, we focus on supervised learning algorithms that act on networked environments. These methods utilize external information in the form of labels to induce or train their models. Generally, the learning process is composed of two serial steps denominated training and classification phases. While in the first the algorithm attempts to learn from the data according to some external aid, such as of a human expert, in the latter the algorithm is tested against unseen data to verify its generalization power. In network-based methods, both phases take place in a network by navigating through it or updating its structure according to new information originated from the human expert. In the test phase, normally the network structure remains static as new data items are classified. However, some algorithms attempt to update the learned network structure in a process classified as self-learning. In this chapter, we present some of the shortcomings and advantages of using the network-based approach to conduct supervised learning. Representative network-based methods are discussed.

5.1 Introduction

Network-based unsupervised and semi-supervised learning techniques have been extensively studied in the literature [4, 10]. There are still, however, few reported network-based supervised learning techniques [3]. In this regard, there is a big space for the proposal and discovery of new ways of supervised learning in networked environments. Presumably, several network-based semi-supervised inductive methods, such as those presented in [2, 5, 19], can be converted into a supervised learning scheme when a reasonable number of labeled instances is provided. However, these methods aim at considering unlabeled instances during the training phase and a network-based approach is employed to model the data into a manifold in order to first propagate the labels to all of the unlabeled instances. In this case, if the majority of instances in the data set is labeled, there is no space for label propagation in a data network. Thus, a regular supervised approach that uses only labeled instances in the learning process would be preferable [3].

Another type of network-based classification approach refers to relational classification. Such type of supervised classification deals with data that differs from the typical data because they violate the instance-independence assumption, which

T.C. Silva, L. Zhao, *Machine Learning in Complex Networks*,
DOI 10.1007/978-3-319-17290-3_5

means that the class label of an instance might not depend only on its own attributes, but also on the labels of its neighbors [14]. This kind of data is usually presented in a network form (also termed within-network data) with some of the vertices labeled and the rest unlabeled. The task is to infer the labels of the unlabeled vertices. Relational classification techniques can be applied to solve a wide range of problems, such as in the discovery of molecular pathways in gene expressions [17], classification of linked scientific research papers [13], link prediction [12], among others. For example, in link prediction on social networks [1, 7, 9, 12], the task is to predict the likelihood of a future association between two vertices, knowing that there is no association between the vertices in the current state of the network [9]. Such approach has a wide variety of further applications, among which we highlight in: recommendation systems, identification of probable professional or academic associations in e-commerce sites or scientific collaboration networks, and of structures of criminal networks and structural analysis in the field of microbiology or biomedicine. All of these applications demand for much more efficient and versatile approaches for link prediction, thereby making it an important and scientifically attractive research topic. Another application that is also related to relational classification is defined as the detection of small connected subgraphs that best capture the relationship between two vertices in a social network. In this respect, the research in [8] proposed an efficient algorithm based on electrical circuit laws to find the connected subgraph from large social networks. It has also been shown that a connected subgraph can be used to effectively compute several topological feature values for the supervised link prediction problem, especially when the network is very large [9].

State-of-the-art approaches that spread labels inside the network to infer labels of interrelated vertices in a jointly manner are known as collective inference models [16]. This kind of inference can significantly reduce classification error when compared to traditional inference techniques [11]. Collective inference methods may use both data attributes and data relational features to perform classification. Traditionally, vector-based methods have treated data items as independent ones, which makes it possible to infer class membership on an instance per instance basis. With networked data, the class membership of one data item (vertex) may have an influence on the class membership of a related vertex. Furthermore, vertices that are not directly linked may be related by chains of links, fact that suggests that it may be beneficial to infer the class memberships of all of the vertices simultaneously. Collective inference in relational data makes simultaneous statistical judgments regarding the values of an attribute or attributes of multiple entities in a network for which some attribute values are not known [16].

In the literature, some algorithms have been proposed that only employ collective inference on specific phases of the learning process. For example, one may employ a local classifier, such as Naïve Bayes or relational probability trees, to predict labels for each unlabeled vertex and further use a collective inference algorithm, such as

ICA [13] or Gibbs sampling [11], to restate the class labels of vertices that are employed in the next iteration. Such kinds of methods are called local classifiers. Another kind of approach, called global formulation-based methods, does not use a separate local classifier, but it uses the entire algorithm for the training and inference, also using relational and non-relational data. Such an approach conducts training with the objective of optimizing a global objective function. Examples of these algorithms include loopy belief propagation and relaxation labeling [18]. In search of a unification on relational data classification in networks, the research in [15] proposed a general supervised learning network-based framework. The framework builds a model considering three components: a local classifier, which makes use of a training set to estimate the probability distribution of the classes; a relational classifier, which also aims to estimate a probability distribution but now considering the neighboring relations in the network; and a collective inference component, which further refines the class predictions.

While collective inference presents some advantages, in some cases, inferring labels collectively causes uncertainties that may actually lead to lower classification accuracies when compared to non-relational approaches. For example, an incorrectly predicted label may influence the predictions of its neighbors in future iterations, possibly cascading this error through long chains of vertices [16]. On one hand, there is a tendency to represent data by networks; on the other hand, some approaches consider transforming networked data into raw, vector-based data in order to apply classical methods, such as SVM and neural networks. This kind of method requires extracting features from the networked data in order to construct a trainable vector-based set. The task of feature extraction from a given relational data can be divided according to the presence or absence of labels in the vertices, and named label-dependent and label-independent extraction, respectively. The former uses both network structure and label information throughout the neighboring vertices and the latter exclusively considers the network structure [16]. An approach to estimate the similarity between edges in a network or between two networks as a whole is graph kernel. Briefly, these kinds of methods use a kernel to establish a similarity measure on networks. In this approach, the main difficulty is in defining a kernel that is suitable for the network structure and reasonably efficient to be evaluated.

5.2 Representative Network-Based Supervised Learning Techniques

In the following, we present several representative network-based supervised learning techniques.

5.2.1 Classification Using k-Associated Graphs

This technique is introduced in [3]. As usual for network-based methods, the basis of the k-associated graphs technique lies on representing the training set as a network, more specifically a directed network referred to as k-associated graph. Such a network is built from a vector-based data set by abstracting data items to vertices and pairwise similarities to edges. After a k-associated graph is constructed for a given k, the purity measure for every component in the network is computed and is used to determine the optimal network for classification, both on the training and the test phases. The edges in a k-associated graph are established in accordance with a modified version of the k-nearest neighbor technique. In this peculiar network formation heuristic, only vertices that share the same label or class are permitted to interconnect. This simple rule generates class components in the overall network.

The purity is defined for each component (isolated subgraph) in the network as follows: given a parameter k, which is used to construct the networks using the modified version of the k-nearest neighbor technique, a vertex can have at most $2k$ connections. Since the resulting networks are digraphs or directed networks, each vertex will have degrees ranging from k to $2k$.[1] The purity measure explores this feasible range of degree values that each vertex can assume. In essence, it quantifies the proportion of edges that has effectively been created between vertices of the same class over the total number of possible connections per each vertex, $2k$. In mathematical terms, the purity ϕ of component α, $\phi^{(\alpha)}$, is then defined as:

$$\phi^{(\alpha)} = \frac{\bar{k}^{(\alpha)}}{2k}, \tag{5.1}$$

in which $\bar{k}^{(\alpha)}$ denotes the average degree of component $\alpha \in \mathscr{G}$. In general, a purity value close to 1 indicates that a large portion of edges are shared among vertices in the network component, resulting in a high-density component, while lower values reveal high levels of class mixture between components of different classes. It is for this reason $\phi^{(\alpha)}$ is called a purity measure for component α: large values indicate purer components in terms of connections. The purity measure can be conceived as the *a priori* probability of connections within a component. This property is explored by the classifier to decide the classes of each of the test instances.

In the referred technique, one can note that parameter k plays a key role in the learning process during the training phase, as its value has considerable implications on the resulting network topology. By virtue of this, a procedure for estimating the value of k for each of the components has been developed in [3]. It is intuitive that some networks may have better components than others according to the purity measure. Rarely, the network obtained using a uniform and unique value of k for

[1] In an undirected network, in contrast, each vertex will always have a degree of $2k$, because whenever vertex $j \in \mathscr{V}$ is one of the k-nearest neighbors of $i \in \mathscr{V}$, then the reciprocal is always true.

all of the network component produces the best configuration of vertices into class components. A single value of k produces components with nearly the same size, therefore structure and purity are restrained to only one possible value of k at a time. Consequently, it would be better to allow for multiples values of k to represent the same data space. In this way, each class component can then decide the best fit of k in accordance with the observed data distribution, producing therefore class-dependent component sizes and purity values. Bearing this in mind, a suggestive idea is to obtain a network with the best organization of the data into components with different and localized k, i.e., each component has its own optimal k. The optimality is obtained by choosing a class-dependent k such that the purity of each network component is the highest. The network component together with its optimal class-dependent k value is termed the k-associated optimal network.

To obtain the optimal k-associated graph, the rationale is to increase k while keeping the best components found so far starting from the 1-associated graph. For each k and network component, the purity measure is calculated and is used to compare between components of different k-associated graphs formed within different values of k. The component with the highest purity value is maintained, while the others are discarded.

Once the optimal k-associated graph has been properly obtained, then the classification phase begins. In this phase, the authors use a Bayes classifier in order to predict. Specifically, the *a priori* probabilities are calculated using a normalized purity value of each of the class components, rather than the traditional size proportions that we encounter in the literature.

A great potentiality of this technique is that no parameters are needed to be adjusted, which eliminates the step of external model selection. However, since the algorithm must create a network from the vector-based data set, then its time complexity is at least of the order of $\mathcal{O}(V^2)$.

5.2.2 Network Learning Toolkit (NetKit)

This work is introduced in [15]. This is a within-network inference technique, rather than an across-network inference method. In within-network inference methods, the training data items are connected directly to the test entities whose labels or classes are to be estimated. In contrast, in across-network inference, we often learn from one network and apply the learned models to a separate, presumably similar network. In essence, the toolkit is composed of three terms, each of which focusing on different perspectives or visions of the data items. The modules are:

1. *Non-relational ("local") module*: This component consists of a (learned) model that only uses local information of the data items, namely the information about their attributes, to estimate their labels or classes. The local models can be used to generate priors that comprise the initial state for the relational learning and collective inference components. They also can be used as one source of evidence

during collective inference. These models are typically produced by traditional machine learning methods.
2. *Relational module*: In contrast to the non-relational component, the relational model makes use of the relations in the network as well as of the attribute values of related entities, possibly through long chains of relations. Relational models also may use local attributes of the data items.
3. *Collective inference module*: The collective inference component determines how the unknown values are estimated together, possibly influencing each other in a collective manner.

Depending on the choices of each of the three aforementioned components, one can get new types of classifiers. Some of these choices result in well-known classifiers in the related community. For example, using a Naïve Bayes classifier as the local model, a Naïve Bayes Markov Random Field classifier for the relational model, and relaxation labeling for the collective inference module form the system used by Chakrabarti *et al.* [15].

It is worth registering that the collective inference component can explore relational autocorrelation, which is a widely observed characteristic of relational data. This phenomenon may reveal that a variable for one instance is highly correlated with the value of the same variable on another instance. By making inferences about multiple data instances simultaneously, collective inference can significantly reduce classification error in some cases.

The importance of NetKit is threefold: (i) it generalizes several existing methods for classification in networked data, thereby making comparison to existing methods possible; (ii) it enables the creation and use of many new algorithms by its modularity and extensibility; and (iii) it enables the analysis/comparison of individual components and configurations. These contributions are welcomed by the literature, because, since then, there has been no systematic study of machine learning methods for within-network classification that compares various algorithms on several data sets.

5.2.3 Classification Using Ease of Access Heuristic

This network-based supervised classification technique is proposed in [6]. The intuition of this method is to perform the classification task using a heuristic called *ease of access* in a networked environment. The measurement of ease of access is built upon the concept of limiting probabilities in the Markov chain theory. First, a set of labeled instances is mapped as vertices of a network. Recalling that a network can be conceived as a discrete Markov chain, each vertex then represents a state in the Markovian process. When classifying an unlabeled data, this network of only labeled vertices is modified by a specific link weight composition, which takes into account the bias information of that unlabeled instance. The bias information alters the network structure in a way that, after the computation of the modified limiting

probabilities, the most easily reached labeled instances represent the class label of that unlabeled instance.

The classification problem requires a given labeled data set, $\mathscr{L} = \{x_1, x_2, \ldots, x_L\}$, where each instance is described by P attributes $x_i = (x_{i1}, x_{i2}, \ldots, x_{iP})$. Each instance in this set has a single assigned label $y \in \mathscr{Y}$. It is also given an unlabeled data set, $\mathscr{U} = \{x_{L+1}, x_{L+2}, \ldots, x_{L+U}\}$, containing instances whose labels are to be estimated. There are L labeled instances and U unlabeled instances. The classification technique is divided into the two classical phases: training and classification.

5.2.3.1 Training Phase

In the training phase, a weighted and undirected network $\mathscr{G} = \langle \mathscr{V}, \mathscr{E} \rangle$ is constructed without self-loops. Vertices represent labeled data instances, $\mathscr{V} = \mathscr{L}$, and link weights are established using a similarity function (cf. Sect. 4.2) and a network formation strategy (cf. Sect. 4.3). At the end of this phase, we get a network $\mathscr{G}$ called training network.

5.2.3.2 Classification Phase

To classify an unlabeled instance $x \in \mathscr{U}$, a weight vector $s = [s_1, s_2, \ldots, s_L]$ is first calculated, in which each entry s_i contains the similarity of that unlabeled data to the labeled vertex i. That is, vertex x is inserted into the training network $\mathscr{G}$ by calculating the link weights to all of the other labeled vertices into this network, and is subsequently removed from $\mathscr{G}$. Then, the weighted and asymmetric adjacency matrix with L vertices is perturbed as follows:

$$\hat{\mathbf{A}} = \mathbf{A} + \epsilon \hat{\mathbf{S}}, \tag{5.2}$$

in which $\mathbf{A}$ and $\hat{\mathbf{A}}$ are the original and the perturbed adjacency matrices, respectively; ϵ is a non-negative parameter; and $\hat{\mathbf{S}}$ is the following $L \times L$ matrix:

$$\hat{\mathbf{S}} = \begin{bmatrix} \mathrm{S}_{(1)} \\ \mathrm{S}_{(2)} \\ \vdots \\ \mathrm{S}_{(L)} \end{bmatrix}, \tag{5.3}$$

in which $\mathrm{S}_{(i)}$ is a row-vector $L \times 1$ whose entries are all s_i. It can be observed in (5.2) that the weight biases of the unlabeled instance x, encoded in matrix $\hat{\mathbf{S}}$, are applied over all of the links in the original adjacency matrix $\mathbf{A}$ of the training network $\mathscr{G}$, that is, the weight of each link is linearly added up with its corresponding weight bias. The idea behind this operation is that the distance between any pair of vertices

is modified due to the new weights of network routes introduced by the insertion of the link biases from the unlabeled instance links. The higher the similarity between the unlabeled instance and a vertex, say vertex i, the more strengthened are the connections from all of the other vertices to vertex i after this operation. The parameter ϵ controls the influence of the weight biases in the training network. The larger the value of parameter ϵ is, the greater is the influence of the bias weights.

The perturbed adjacency matrix $\hat{\mathbf{A}}$ is termed as the classification network. By using this network, it is now possible to apply the random walk limiting probabilities over the states represented by the network vertices. The transition probabilities can be found by means of the matrix $\hat{\mathbf{A}}$. To compute the entries of the transition matrix $\mathbf{P}$, the entries of matrix $\hat{\mathbf{A}}$ are normalized:

$$\mathbf{P}_{ij} = \hat{\mathbf{A}}_{ij} \Big/ \sum_{j \in \mathscr{V}} \hat{\mathbf{A}}_{ij}. \tag{5.4}$$

With the above matrix $\mathbf{P}$ at hand, the limiting probabilities can be calculated by using one of two possible ways: finding the eigenvector corresponding to the unit eigenvalue of matrix $\mathbf{P}$ or iterating the system

$$p_{i+1} = p_i \mathbf{P} \tag{5.5}$$

to the stationary state, where p is the state distribution. Under the constraints discussed in Sect. 2.4.1, the limiting or stationary probability is unique and independent on the system's initial state and has the form:

$$\mathbf{p}^{\infty} = \pi = [\pi_1, \pi_2, \ldots, \pi_L], \tag{5.6}$$

in which each element represents a state, and each entry p_i can be interpreted as the probability of x to belong to the class of state i.

As the final step, the classification of x is accomplished by assigning it the most representative label from the set of states. To accomplish that, a set $\mathscr{T}$ containing the t states with the largest limiting probabilities are selected and the most representative class in $\mathscr{T}$ is associated to x.

5.3 Chapter Remarks

The literature in network-based supervised learning is still very scarce, as very few methods have been developed so far. This is mainly due to the fact that the large percentage of labeled data makes the data network almost fixed. Then, there is not enough space for label propagation. Up to now, the developed techniques are based on two main ideas: (1) collective inference with the data network, as done in k-associated graphs [3] and the network learning toolkit [15]; (2) classification using the pattern formation of the entire network, as performed in the classification that uses the ease of access heuristic [6]. At the conceptual level, the within-network

techniques do not differ much from traditional data classification ones. However, we still have a large space to explore the across-network approach. This is because we have many ways to characterize the global patterns of the data network. In Chap. 8, we walk through a pioneer across-network supervised learning technique.

References

1. Barabási, A.L., Jeong, H., Neda, Z., Ravasz, E., Schubert, A., Vicsek, T.: Evolution of the social network of scientific collaborations. Phys. A Stat. Mech. Appl. **311**(3–4), 590–614 (2002)
2. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J. Mach. Learn. Res. **7**, 2399–2434 (2006)
3. Bertini J.R. Jr., Zhao, L., Motta, R., Lopes, A.A.: A nonparametric classification method based on K-Associated graphs. Inf. Sci. **181**, 5435–5456 (2011)
4. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-supervised learning. Adaptive Computation and Machine Learning. MIT, Cambridge (2006)
5. Chen, J., Fang, H.R., Saad, Y.: Fast approximate kNN graph construction for high dimensional data via recursive lanczos bisection. J. Mach. Learn. Res. **10**, 1989–2012 (2009)
6. Cupertino, T.H., Zhao, L., Carneiro, M.G.: Network-based supervised data classification by using an heuristic of ease of access. Neurocomputing **149**(Part A), 86–92 (2015)
7. Dorogovtsev, S.N., Mendes, J.F.F.: Evolution of Networks: From Biological Nets to the Internet and WWW (Physics). Oxford University Press, Oxford (2003)
8. Faloutsos, C., Mccurley, K.S., Tomkins, A.: Fast discovery of connection subgraphs. In: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining (KDD), pp. 118–127. ACM, New York (2004)
9. Hasan, M.A., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. In: Proceedings of SDM 06 workshop on Link Analysis, Counterterrorism and Security (2006)
10. Jain, A.K.: Data clustering: 50 years beyond K-Means. Pattern Recogn. Lett. **31**, 651–666 (2010)
11. Jensen, D., Neville, J., Gallagher, B.: Why collective inference improves relational classification. In: In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 593–598 (2004)
12. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. **58**(7), 1019–1031 (2007)
13. 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. Washington (2003)
14. Macskassy, S.A., Provost, F.: A simple relational classifier. In: Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at the Knowledge Discovery and Data Mining Conference (KDD), pp. 64–76 (2003)
15. Macskassy, S.A., Provost, F.: Classification in networked data: a toolkit and a univariate case study. J. Mach. Learn. Res. **8**, 935–983 (2007)
16. McDowell, L., Gupta, K.M., Aha, D.W.: Cautious collective classification. J. Mach. Learn. Res. **10**, 2777–2836 (2009)
17. Segal, E., Wang, H., Koller, D.: Discovering molecular pathways from protein interaction and gene expression data. In: Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology, Brisbane, pp. 264–272 (2003)
18. Sen, P., Namata, G.M., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. Artif. Intell. Mag. **29**(3), 93–106 (2008)
19. Sindhwani, V., Niyogi, P., Belkin, M.: Beyond the point cloud: from transductive to semi-supervised learning. In: Proceedings of the 22nd international conference on Machine learning (ICML), pp. 824–831. ACM, New York (2005)