Bias-Guided Random Walk for Network-Based Data Classification

Thiago Henrique Cupertino and Liang Zhao

Institute of Mathematics and Computer Science
University of São Paulo, São Carlos, Brazil
{thiagohc,zhao}@icmc.usp.br

Abstract. This paper presents a new network-based classification technique using limiting probabilities from random walk theory. Instead of using a traditional heuristic to classify data relying on physical features such as similarity or density distribution, it uses a concept called ease of access. By means of an underlying network, in which nodes represent states for the random walk process, unlabeled instances are classified with the label of the most easily reached class. The limiting probabilities are used as a measure for the ease of access by taking into account the biases provided by an unlabeled instance in a specific adjacency matrix weight composition. In this way, the technique allows data classification from a different viewpoint. Simulation results suggest that the proposed scheme is competitive with current and well-known classification algorithms.

Keywords: network-based learning, data classification, supervised learning, random walk, limiting probabilities.

1 Introduction

Supervised machine learning comprises the construction of a model by using information extracted from a training data set. The constructed model defines decision borders that are used to classify unlabeled data [1]. An unlabeled instance is classified depending on its relative position to the decision borders. Due to its importance in various real applications, many classification techniques have been developed, such as Neural Networks, k-Nearest Neighbors (kNN), Linear Discriminate Analysis (LDA), Naive-Bayes Method, Support Vector Machines (SVM) and Decision Tree [1, 2, 3, 4, 5]. These traditional classification techniques divide the data space according to physical features (similarity, distance, or distribution) of the training data. In this way, many intrinsic and semantic relations among data items are ignored as, for example, topological structure and pattern formation.

On the other hand, the usage of an underlying network can take into account these previously mentioned relationships among data. In the machine learning domain, many recent works have applied random walk processes to perform semi-supervised learning. In this learning paradigm, just a few data compose

C. Guo, Z.-G. Hou, and Z. Zeng (Eds.): ISNN 2013, Part II, LNCS 7952, pp. 375–384, 2013.

the training data set, and so the classification processes makes use of the information provided by the unlabeled data, which is most commonly represented by nodes in a network. Many of these works share the regularization framework, differing only in the particular choice of the loss function and the regularizer [6, 7, 8, 9, 10, 11]. In these works, the concept of relationship among data is the measurement of how easy labels propagate or how easy a random walker reaches target nodes on a network structure. In these techniques, the underlying link structure is responsible for giving the probabilities or weights between two neighboring nodes to support the label propagation or the walker transition between two linked nodes. Similarly, random walks have also been extensively applied to unsupervised learning, in which there is no labeled data, such as community detection and data clustering [12, 13, 14, 15, 16]. However, very few efforts have been done for network-based supervised learning [17, 18, 19, 20], and thus this work is also a contribution to the use of network-based techniques in the supervised learning field.

Here, we propose a new network-based classification heuristic which consider the *ease of access* of unlabeled instances to each class. Differently from previous works, the proposed technique uses the dynamical process measure called random walk limiting probabilities. Limiting probabilities are applied to random walk processes to measure the limiting state transitions through an underlying network [21]. In the proposed scheme, the training data set is used to construct the network, in which instances (nodes) represent the states a random walker visits during the process. An unlabeled instance is considered belonging to the class that is most easily reached, that is, the limiting transition probability for a random walker to that class, after the insertion of the unlabeled instance bias in the underlying adjacency weight matrix, is large. As a consequence of the dynamical processes, both local and global relationships among nodes are taken into account.

This paper is organized as follows: section 2 describes the model for the supervised classification technique. In section 3, simulation results and comparisons are presented. Finally, section 4 concludes the paper.

2 Supervised Inductive Classification Model

In this section, the technique is derived for a supervised inductive classification model. To be classified, an unlabeled instance is first inserted into the network of training data as a virtual state. The concept of virtual state means that the probability of belonging to this state is not considered, that is, the random walker can visit this virtual state, but only the information extracted directly from the training data is used for classification. The insertion of an unlabeled instance as a virtual state is carried out by a specific weight composition and aims to provide a bias to the classification process by enhancing the probabilities of the unlabeled instance's network neighborhood. Therefore, the bias prioritizes near classes in the state space (classes with the large transition probability) by adopting the assumption that close instances belong to the same class. The mathematical formulation is given next.

Training Phase. We consider it is given a labeled data set $\mathcal{X}^{(l)} = \{\mathbf{x}_i^{(l)},\ i = 1, \ldots, n\}$ containing only labeled instances, where each instance is described by q attributes $\mathbf{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{iq}\}$. A weighted undirected network $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$ without self-loops is constructed, in which data instances are represented by nodes, $\mathcal{V} = \mathcal{X}^{(l)}$, and similarities among instances are represented by weights of the edges, $\mathcal{E} = [\mathcal{W}_{ij}],\ i, j = 1, \ldots, n$. The network similarity matrix $\mathcal{W} = \{w_{ij}\}$ can be calculated by using any distance function. w_{ij} is the similarity between the pair of instances $\mathbf{x}_i^{(l)}$ and $\mathbf{x}_j^{(l)}$.

Classification Phase. To perform the classification of an unlabeled instance $\mathbf{x}^{(u)}$, the set of nodes $\mathcal{V}$ is considered as a state space set, meaning that each node is a possible state for a random walker. The transition probabilities among states are given by a normalized transition matrix P, whose construction is further explained.

First, the unlabeled instance $\mathbf{x}^{(u)}$ is inserted into the network $\mathcal{N}$. To do so, a similarity vector $\mathcal{S} = [\mathcal{S}_i],\ i = 1, \ldots, n$, between $\mathbf{x}^{(u)}$ and all other nodes $\mathbf{x}_i^{(l)}$ is calculated by using a distance function. Next, a new asymmetric and $n \times n$ modified similarity matrix $\hat{\mathcal{W}}$ is constructed by composing the matrix of weight biases $\hat{\mathcal{S}}$:

$$\hat{\mathcal{W}} = \mathcal{W} + \epsilon\hat{\mathcal{S}}, \tag{1}$$

where ϵ is a non-negative parameter and $\hat{\mathcal{S}}$ is an $n \times n$ matrix composition:

$$\hat{\mathcal{S}} = \begin{bmatrix} \mathcal{S}^T \\ \mathcal{S}^T \\ \vdots \\ \mathcal{S}^T \end{bmatrix},$$

where $\mathcal{S}^T$ is the transpose of vector $\mathcal{S}$.

In Eq. 1, it can be observed that the weight biases of the virtual state $\mathbf{x}^{(u)}$, encoded in matrix $\hat{\mathcal{S}}$, are applied over all edges $\mathcal{W}$ of network $\mathcal{N}$, that is, the weight of each edge is linearly added up with the corresponding weight bias. The idea behind this operation is that the distance between any pair of nodes is reduced because of the new route introduced by the insertion of the unlabeled data instance. The higher the proximity between the unlabeled instance and a node, say node j, the more strengthened the connections from all other nodes to node j are. The parameter ϵ controls the influence of weight bias provided by matrix $\hat{\mathcal{S}}$ on the original network. The larger is the value of parameter ϵ, the greater will be the influence of the bias weights provided by the virtual state.

After the bias composition, the virtual state $\mathbf{x}^{(u)}$ is effectively inserted into network $\mathcal{N}$ as a virtual state. To do so, an $(n+1) \times (n+1)$ adjacency matrix $\mathcal{A} = \{a_{ij}\}$ is constructed:

$$\mathcal{A} = \begin{bmatrix} \hat{\mathcal{W}} & \mathcal{S} \\ \mathcal{S}^{\mathcal{T}} & 0 \end{bmatrix}. \tag{2}$$

In this formulation, without loss of generality, the virtual state is inserted as the last entry $(n+1)$ of matrix $\mathcal{A}$.

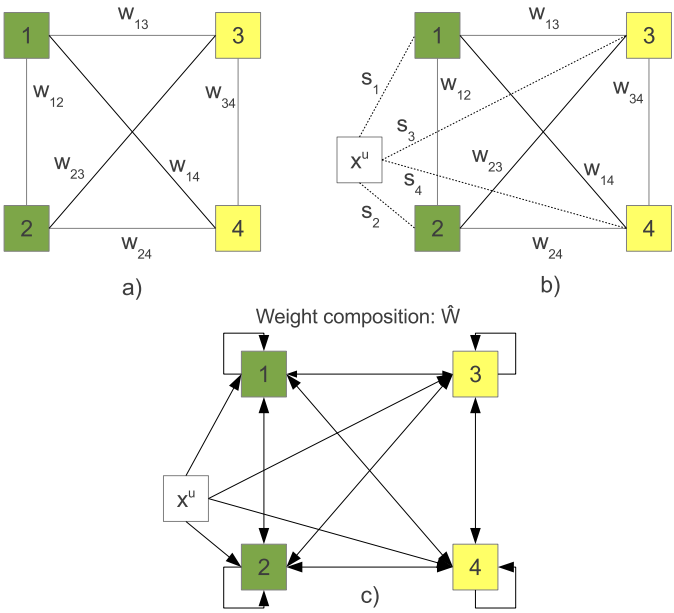

Fig. 1. Illustration for supervised inductive classification of unlabeled instance $\mathbf{x}^u$. a) An undirected and complete network $\mathcal{N}$ is formed by using 4 training instances of 2 classes: green and yellow; b) the similarities $\mathcal{S}$ are calculated for the unlabeled instance $\mathbf{x}^u$; c) modified network $\mathcal{N}$, directed with self-loops, after bias composition (Eq. 1) and insertion of the unlabeled virtual state $\mathbf{x}^u$ (Eq. 2).

The two steps described above can be easily understood by using the toy example depicted in Fig. 1. In this example, a network is formed by 4 labeled instances belonging to 2 distinct classes, *blue* and *red*, each one containing 2 representative instances (Fig. 1a). In the initial network, the links are undirected and the similarity matrix is symmetric. Next, giving an unlabeled instance $\mathbf{x^u}$ to be classified, the similarity vector $\mathcal{S}$ between $\mathbf{x}$ and all other nodes is computed (Fig. 1b). After this computation, the weight biases of $\mathbf{x^u}$ are added up to the original similarity matrix to form a biased similarity matrix for the same network (Eq. 1). After that, the unlabeled instance $\mathbf{x^u}$ is effectively inserted into the network (Eq. 2). It can be seen from Fig. 1c that the network becomes directed and with self-loops (except the virtual state, that has no self-loop) and the biased weight matrix is no more symmetric.

After the virtual state insertion, we are able to compute the entries of the transition matrix $\mathcal{P} = [\mathcal{P}_{ij}]$ by scaling the entries of matrix $\mathcal{A}$:

$$p_{ij} = a_{ij} / \sum_{j=1}^{n+1} a_{ij}.$$

Algorithm 1. Classification Algorithm

Input:
$\mathcal{X}^{(l)}$: training data set
$\mathbf{x}^{(u)}$: unlabeled instance
Parameters:
t : number of largest probabilities to be selected
ϵ : bias weighting
Output:
c : estimated class ($c \in \{1, \ldots, C\}$) for $\mathbf{x}^{(u)}$
Training:
1. $\mathcal{N}$ = Create a network from $\mathcal{X}^{(l)}$
Classification:
2. $\hat{\mathcal{W}}$ = Compose bias weights into $\mathcal{N}$ (Eq. 1)
3. $\mathcal{A}$ = Insert the virtual state $\mathbf{x}^{(u)}$ into $\mathcal{N}$ (Eq. 2)
4. $\mathbf{p}^{\infty}$ = Compute limiting probabilities (Eq. 3)
5. $\mathcal{T}$ = Select the t largest limiting probabilities from $\mathbf{p}^{\infty}$
6. c = Assign $\mathbf{x}^{(u)}$ the most representative class in $\mathcal{T}$

With the above matrix $\mathcal{P}$ at hand, the limiting probabilities can be calculated. This calculation is performed by two ways: by finding the eigenvector corresponding to the unit eigenvalue of matrix $\mathcal{P}$ or, as a faster manner, iterating the system

$$\mathbf{p}_{i+1} = \mathcal{P}\mathbf{p}_i, \tag{3}$$

to the stationary state, where $\mathbf{p}$ is an $(n+1) \times (n+1)$ normalized vector. It results in the following vector:

$$\mathbf{p}^{\infty} = [p_1\, p_2 \ldots p_{n+1}]$$

where each column represents a state, and each entry p_i represents the probability of $\mathbf{x}^{(u)}$ belonging to the class of state i.

As the final step, the classification of $\mathbf{x}^{(u)}$ is accomplished by assigning it the most representative label from the set of states. To achieve that, a set $\mathcal{T}$ containing the t states with the largest limiting probabilities are selected in a descend order and the most representative class in $\mathcal{T}$ is associated to $\mathbf{x}^{(u)}$.

In a concise form, the proposed supervised inductive classification process can be summarized by Algorithm 1.

3 Results for Real Data Sets and Comparisons

We present classification results using the proposed supervised technique, as well as a comparative study against some current classifiers. In the experiments, 15 data sets were selected from the UCI machine learning repository [22]. Table 1 shows the metadata for all data sets. As can be seen in this table, the selection was made to encompass diversity on data domains as well as to consider different number of classes, attributes and class sizes. They vary from 3 to 15, 4 to 91

Table 1. Information of all data sets used in simulations

Domain	Instances	Attributes	Classes
Zoo	101	16	7
Hayes-Hoth	132	5	3
Iris	150	4	3
Teaching	151	5	3
Wine	178	13	3
Image	210	19	7
Glass	214	9	6
E. Coli	336	8	8
Libras	360	91	15
Balance	625	4	3
Vehicle	846	18	4
Vowel	990	13	11
Yeast	1484	8	10
Wine Q. (Red)	1599	12	6
Segment	2310	19	7

and 101 to 1599, respectively. The Euclidean distance was used in all simulations as a distance measurement. Eventual categorical attributes, in data sets such as Balance and Zoo, were treated as numerical. As a data preparation, each instance vector was normalized to have a magnitude of 1. Individual cases were normalized by dividing each attribute of the instance by the square root of the sum of the squares of the individual attributes. Thus, an instance $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ was normalized by dividing each attribute x_{ij} by $\sum_{j=1}^{p} {x_{ij}}^2$.

The parameter optimization for the proposed technique was done as follows. It was created a complete network $\mathcal{N}$ (Step 1 of Alg. 1) by using the Euclidean measurement as a distance function. Parameter t (Step 5 of Alg. 1) ranged from 1 to the number of instances in the largest class of the training set. Parameter ϵ (Eq. 1) was evaluated by the grid method with the values $\{0, 0.1, 0.2, \ldots, 10\}$.

The influence of parameter ϵ in the classification accuracy was evaluated. As stated before in Sec. 2, ϵ is responsible for weighting the biases provided by the virtual state. Figure 2 depicts the accuracy in function of parameter ϵ for eight selected data sets. The results were averaged over 50 simulations. Each simulation was performed by using a 10-fold stratified cross-validation process [23]. In this process, the data set is split in 10 disjoint sets and, in each run, 9 sets are used as training data and 1 set is used as the test data, resulting in a total of 10 runs. Therefore, 50×10 runs were executed. It can be seen on Fig. 2 that next to value 0 - where the biases influence are reduced because of a small weight - the classification accuracies are poor. On the other hand, as ϵ becomes larger, the accuracies increase and stabilize before it approaches 10. In this later case, the biases play a main role due to the large weight applied to them (Eq. 1). These scenarios configure a convergent behavior for parameter ϵ and can help in the simulations by restricting the search space.

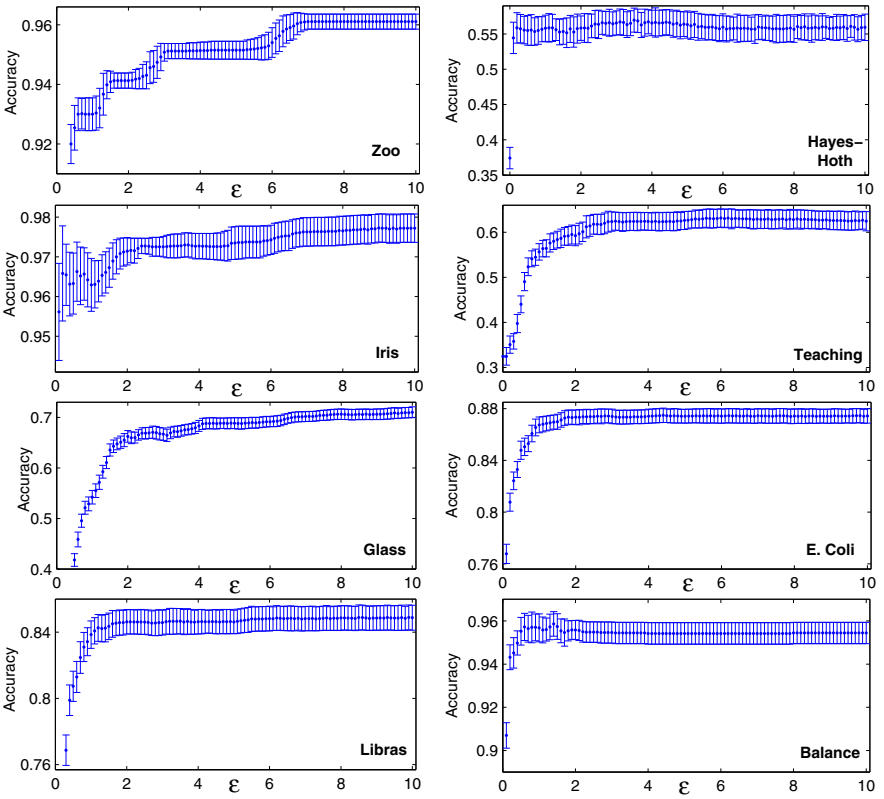

Fig. 2. Classification accuracy in function of parameter ϵ. The curves show standard deviations for each point in error bars. 50 simulations were averaged.

The proposed technique was compared to other 4 well-known multi-class classification algorithms: Weighted kNN (WkNN), Decision Tree C4.5 [24], Multi-Class SVM (MSVM) [25] and the network-based k-Associated Optimal Graph (kAOG) [17]. For the parametric algorithms (all of the algorithms except kAOG), a repeated cross-validation [23] was done in order to optimize their respective parameters. For the MSVM algorithm, we used the *one-against-one* multi-class version, in which $C(C-1)/2$ binary classifiers distinguishes between every pair of classes by using a voting scheme. To avoid ties, the output of each MSVM corresponded to the real valued decision functions. For reducing the parameter search space in MSVM model selection, the only kernel in consideration was the radial basis function, $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2}$, and the stopping criterion for the optimization method was defined as the Karush-Kuhn-Tucker violation to be less than 10^{-3}, the same condition used in [26]. In the WkNN, the classification process was performed by using the sum of the weights between the instance to be labeled and its k-nearest neighbors. Specifically, the weight between two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined by $1/\mu(\mathbf{x}_i, \mathbf{x}_j)$, where μ is a

Table 2. Classification accuracy (%) followed by standard deviation and rank for each algorithm. Each result shows the adjusted parameters for model selection. Best result for each data set is in bold face.

Data set	**Proposed** (ϵ, t)	**kAOG**	**WkNN** (k)	**C4.5** (cf, m)	**MSVM** (cp, γ)
Zoo	96.2 ± 0.3 (7.8, 1)	**97.0 ± 5.2**	96.2 ± 5.8 (1)	95.8 ± 5.3 (0.1, 0)	96.3 ± 6.4 ($2^1, 2^1$)
Hayes-Hoth	**57.0 ± 1.7** (3.5, 1)	55.7 ± 12.6	56.8 ± 13.2 (3)	46.7 ± 10.5 (1, 0)	45.4 ± 13.1 ($2^{12}, 2^{13}$)
Iris	**98.1 ± 0.2** (3.1, 29)	97.4 ± 3.2	97.9 ± 3.3 (19)	95.0 ± 5.8 (0.25, 2)	97.0 ± 4.6 ($2^{-2}, 2^3$)
Teaching	**63.1 ± 2.0** (6.0, 1)	62.5 ± 11.6	63.0 ± 12.3 (9)	58.2 ± 14.9 (1, 0)	52.5 ± 7.9 ($2^6, 2^3$)
Wine	84.6 ± 1.8 (9.4, 1)	83.5 ± 8.5	84.1 ± 8.2 (1)	91.7 ± 6.7 (0.5, 1)	**94.4 ± 5.8** ($2^{11}, 2^2$)
Image	75.5 ± 1.0 (1.8, 1)	75.3 ± 7.5	75.4 ± 8.2 (3)	80.7 ± 7.6 (0.8, 3)	**86.7 ± 7.4** ($2^{10}, 2^{-3}$)
Glass	**72.5 ± 1.2** (10.0, 2)	**72.5 ± 8.1**	71.8 ± 9.0 (1)	66.9 ± 9.4 (0.1, 3)	69.5 ± 5.6 ($2^{10}, 2^4$)
E. Coli	**87.5 ± 0.6** (4.4, 9)	85.8 ± 6.4	87.4 ± 5.4 (9)	83.6 ± 6.1 (0.25, 3)	86.7 ± 8.2 ($2^{12}, 2^{-9}$)
Libras	84.9 ± 0.7 (9.0, 1)	85.4 ± 5.3	84.8 ± 5.4 (1)	71.6 ± 7.5 (0.8, 1)	**86.6 ± 5.0** ($2^7, 2^2$)
Balance	96.3 ± 0.5 (4.1, 9)	94.9 ± 2.5	96.7 ± 2.1 (11)	89.6 ± 3.7 (0.5, 1)	**98.2 ± 0.9** ($2^7, 2^0$)
Vehicle	67.6 ± 0.7 (10, 4)	67.9 ± 4.4	67.6 ± 4.1 (5)	70.7 ± 3.5 (0.5, 2)	**84.4 ± 3.4** ($2^10, 2^3$)
Vowel	97.5 ± 0.9 (10.0, 1)	**98.9 ± 0.7**	98.8 ± 0.9 (11)	78.6 ± 4.3 (0.5, 0)	97.5 ± 1.9 ($2^7, 2^0$)
Yeast	59.4 ± 0.4 (10.0, 12)	53.6 ± 3.8	**60.9 ± 3.6** (16)	55.8 ± 3.6 (0.1, 5)	58.9 ± 4.8 ($2^{11}, 2^0$)
Wine Q. Red	61.0 ± 0.3 (10.0, 1)	61.8 ± 3.6	**64.0 ± 3.8** (19)	59.8 ± 2.4 (1, 0)	60.4 ± 3.2 ($2^9, 2^1$)
Segment	93.1 ± 0.5 (10.0, 1)	93.7 ± 1.5	93.6 ± 1.4 (5)	95.4 ± 1.2 (1, 0)	**96.6 ± 1.2** ($2^11, 2^0$)
Avg. Rank	**2.67 ± 1.50**	3.33 ± 1.80	2.80 ± 1.26	5.00 ± 1.96	3.13 ± 2.20

distance measurement. The number of neighbors k ranged from 1 to the number of instances in the largest class of the training set. For the C4.5 algorithm, two parameters were adjusted, the confidence factor which assumes the values $cf \in \{0, 0.1, 0.25, 0.5, 0.8, 1\}$, where smaller values incur more pruning (1 is for no pruning), and the minimum number of instances that a set must have in order to be further partitioned is $m \in \{0, 1, 2, 3, 4, 5, 10, 15, 20, 50\}$.

Table 2 presents the classification accuracy on the test set followed by the standard deviation. Best results are in bold face. At the last line, the average rank for each algorithm is shown. The calculation procedure for the rank measurement is as follows: i) for each data set, the algorithms were ranked according to their average performance, that is, the best algorithm was ranked as first, the second best was ranked as second, and so on; and ii) for each algorithm, the average rank was based on the rank values on all the data sets. It can be seen that the proposed technique achieved the best average ranking amongst all simulated techniques. Moreover, it is worth emphasizing that our technique exhibited the smallest deviation values over all results.

4 Conclusions

This paper has presented a new network-based classification technique which applies limiting probabilities from random walk theory. In the supervised model, these probabilities represent the ease of access an unlabeled instance has to the

classes in the training set in an underlying network. Unlabeled nodes are classified with the label of the class most easily reached. Simulations have suggested that the proposed technique is competitive with the current well-known classification techniques. As future works, mathematical models could be studied to describe and shed more light to the proposed technique. We hope this research contributes to the network-based learning area, specially to development of new supervised classification heuristics.

Acknowledgments. The authors would like to acknowledge the São Paulo State Research Foundation (FAPESP) and the Brazilian National Council for Scientific and Technological Development (CNPq) for the financial support given to this research.

References

[1] Duda, R.O., Stork, D.G., Hart, P.E.: Pattern classification, 2nd edn. Wiley-Interscience (2000)
[2] Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning 29, 131–163 (1997)
[3] Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2(2), 121–167 (1998)
[4] Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice Hall, Englewood Cliffs (1998)
[5] Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining, 1st edn. Addison-Wesley (2005)
[6] Blum, A., Chawla, S.: Learning from labeled and unlabeled data using graph mincuts. In: Proceedings of the Eighteenth International Conference on Machine Learning, San Francisco, pp. 19–26. Morgan Kaufmann (2001)
[7] Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: Proc. 20th Int. Conf. Mach. Learn., pp. 912–919 (2003)
[8] Belkin, M., Matveeva, I., Niyogi, P.: Regularization and semi-supervised learning on large graphs. In: Shawe-Taylor, J., Singer, Y. (eds.) COLT 2004. LNCS (LNAI), vol. 3120, pp. 624–638. Springer, Heidelberg (2004)
[9] Belkin, M., Niyogi, P., Sindhwani, V.: On manifold regularization. In: Proc. 10th Int. Workshop Artif. Intell. Stat., pp. 17–24 (2005)
[10] Zhou, D., Schölkopf, B.: Adaptive computation and machine learning. In: Discrete Regularization, pp. 237–250. MIT Press, Cambridge (2006)
[11] Silva, T., Zhao, L.: Network-based stochastic semisupervised learning. IEEE Transactions on Neural Networks and Learning Systems 23(3), 451–466 (2012)
[12] Zheng, X., Lin, X.: Automatic determination of intrinsic cluster number family in spectral clustering using random walk on graph. In: 2004 International Conference on Image Processing, ICIP 2004, vol. 5, pp. 3471–3474 (October 2004)
[13] Alamgir, M., von Luxburg, U.: Multi-agent random walks for local clustering on graphs. In: 2010 IEEE 10th International Conference on Data Mining, ICDM, pp. 18–27 (December 2010)

[14] Cai, B., Wang, H., Zheng, H., Wang, H.: An improved random walk based clustering algorithm for community detection in complex networks. In: 2011 IEEE International Conference on Systems, Man, and Cybernetics, SMC, pp. 2162–2167 (October 2011)
[15] Breve, F., Zhao, L., Quiles, M., Pedrycz, W., Liu, J.: Particle competition and cooperation in networks for semi-supervised learning. IEEE Transactions on Knowledge and Data Engineering 24(9), 1686–1698 (2012)
[16] Silva, T., Zhao, L.: Stochastic competitive learning in complex networks. IEEE Transactions on Neural Networks and Learning Systems 23(3), 385–398 (2012)
[17] Bertini, J.R., Zhao, L., Motta, R., de Andrade Lopes, A.: A nonparametric classification method based on k-associated graphs. Information Sciences 181, 5435–5456 (2011)
[18] Cupertino, T.H., Silva, T., Zhao, L.: Classification of multiple observation sets via network modularity. Neural Computing and Applications (Print) (2012)
[19] Cupertino, T.H., Zhao, L.: Using katz centrality to classify multiple pattern transformations. In: Proceedings of the 2012 Brazilian Symposium on Neural Networks, pp. 1–6 (2012)
[20] Silva, T., Zhao, L.: Network-based high level data classification. IEEE Transactions on Neural Networks and Learning Systems 23(6), 954–970 (2012)
[21] Gallager, R.G.: Discrete Stochastic Processes, 1st edn. Springer (1996)
[22] Frank, A., Asuncion, A.: UCI machine learning repository (2010)
[23] Kim, J.H.: Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. Computational Statistics and Data Analysis 53, 3735–3745 (2009)
[24] Quinlan, J.R.: C4.5: Programs for Machine Learning, 1st edn. Morgan Kaufman Publishers (1993)
[25] Vapnik, V.: The Nature of Statistical Learning Theory, 1st edn. Springer (1999)
[26] Hsu, C.W., Lin, C.J.: A comparison of methods for multiclass support vector machines. IEEE Transactions on Neural Networks 13, 415–425 (2002)