Particle Competition in Complex Networks for Semi-Supervised Classification

Fabricio Breve, Liang Zhao, and Marcos Quiles

Institute of Mathematics and Computer Science, University of São Paulo, São Carlos SP 13560-970, Brazil,
{fabricio,zhao,quiles}@icmc.usp.br,

Abstract. Semi-supervised learning is an important topic in machine learning. In this paper, a network-based semi-supervised classification method is proposed. Class labels are propagated by combined random-deterministic walking of particles and competition among them. Different from other graph-based methods, our model does not rely on loss function or regularizer. Computer simulations were performed with synthetic and real data, which show that the proposed method can classify arbitrarily distributed data, including linear non-separable data. Moreover, it is much faster due to lower order of complexity and it can achieve better results with few pre-labeled data than other graph based methods.

Key words: semi-supervised learning, particle competition, complex networks, community detection

1 Introduction

Complex networks is a recent and active area of scientific research, which studies large scale networks with non-trivial topological structures, such as computer networks, telecommunication networks, transportation networks, social networks and biological networks [1, 2, 3]. Many of these networks are found to be divided naturally into communities or modules, thus discovering of these communities structure became one of the main issues in complex network study [4, 5, 6, 7, 8]. Recently, a particle competition approach was successfully applied to detect communities modeled in non-weighted networks [9].

The problem of community detection is also related to the machine learning field, which is concerned with the design and development of algorithms and techniques that allow computers to "learn", or improve their performance through experience [10]. Machine learning algorithms usually falls in one of these two categories: *supervised learning* and *unsupervised learning*. In supervised learning, the algorithm learns a function from the training data, which consists of pairs of samples and their respective labels, so after having seen a number of training examples the algorithm can predict the labels of unseen data. On the other hand, in unsupervised learning the samples are unlabeled and the objective is to determine how the samples are organized. One form of unsupervised learning is clustering, which is the partitioning of a data set into subsets (clusters), so

that the data in each cluster share some characteristics. The algorithm proposed in [9] to detect communities belongs to the unsupervised learning category.

With the emergence of the complex networks field and the study of larger networks, it is common to have large data sets in which only a small subset of samples are labeled. That happens because unlabeled data is relatively easy to collect, but labeling samples is often an expensive, difficult or time consuming task, since it often requires the work of humans specialists. Supervised learning techniques cannot handle this kind of problem because they require all samples labeled before the training process. Unsupervised learning techniques cannot be applied to solve this kind of problem either because they ignore label information of samples. In order to solve these problems a new class of machine learning algorithms arose, the semi-supervised class. *Semi-supervised* learning is halfway between supervised and unsupervised learning, it address these problems by combining a few labeled samples with a lot of unlabeled samples to produce better classifiers while requiring less human effort [11, 12]. For example, consider Fig. 2a, it is a toy data set with 2000 samples, but only 20 of them are labeled (red circles and blue squares), a supervised algorithm would learn from only these 20 samples and it would probably misclassify a lot of unseen samples, while an unsupervised algorithm would consider all the 2000 samples without any distinction, thus not taking advantage of the labeled ones. On the other hand, a semi-supervised algorithm can learn from both labeled and unlabeled samples, probably producing a better classifier. Semi-supervised methods include generative models [13, 14], cluster-and-label techniques [15, 16], co-training techniques [17, 18], low-density separation models, like Transductive Support Vector Machines (TSVM) [19], and graph-based methods, like Mincut [20] and Local and Global Consistency [21].

Traditional semi-supervised techniques, such as Transductive Support Vector Machine (TSVM) [19], can identify data classes of well defined form, but usually fail to identify classes of irregular form. Thus, assumptions on class distribution have to be made and unfortunately it is usually unknown a priori. On the other hand, since most graph based methods have high order of computational complexity ($O(n^3)$), it makes their use limited to small data sets [11]. This is considered as a serious shortage because semi-supervised learning techniques are usually applied to data sets with large amount of unlabeled data. Also, many graph based methods can be viewed as regularization frameworks, they are similar to each other, basically differing only in the particular choice of the loss function and the regularizer [22, 20, 21, 23, 24, 25].

In this paper we present a new kind of network-based semi-supervised classification technique, by using particle walking and competition. We extend the model proposed in [9] to handle weighted networks and to take advantage of pre-labeled data. The main contributions of this new method are:

– unlike most other graph-based models, it does not rely on loss functions or regularizers;
– it can classify arbitrarily distributed data, including linear non-separable data;

– it is much faster than other graph-based methods due to its low order of complexity and thus it can be used to classify large data sets;
– it can achieve better results than other graph-based methods when a small number of data samples is labeled.

This paper is organized as follows: Section 2 describes the model in details. Section 3 shows some experimental results from computer simulations, and in Section 4 we draw some conclusions.

2 Model Description

Our model is an extension of the particle competition approach proposed by [9]. Their model is used to detect communities in networks, represented by non-weighted networks. There are several particles walking in a network, competing with each other for the possession of network nodes, and rejecting intruder particles. In this way, after a number of iterations, each particle will be confined within a community of the network, so the communities can be divided by examining the nodes ownership.

In this paper, we have changed the nodes and particles dynamics, and some other details that will follow, so the new model is not only suitable to conduct semi-supervised learning, but also can represent weighted networks (weights represent pair-wise similarity between data samples). The model is described as follows:

Given a data set $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^m$ and a label set $L = \{1, 2, \ldots, c\}$, some samples x_i are labeled as $y_i \in L$ and some are unlabeled as $y_i = \emptyset$. The goal is to provide a label to these unlabeled samples.

First, we define a graph $G = (V, E)$, with $V = \{v_1, v_2, \ldots, v_n\}$, and each node v_i corresponds to a sample x_i. An affinity matrix W [21, 26] defines the weight between the edges in E as follows:

$$W_{ij} = \exp -||x_i - x_j||^2/2\sigma^2 \quad \text{if} \quad i \neq j, \tag{1}$$

$$W_{ii} = 0, \tag{2}$$

where W_{ij} defines the pair-wise relationship between x_i and x_j, with the diagonal being zero, and σ is a scaling parameter which controls how quickly the affinity W_{ij} falls off with the distance between x_i and x_j.

Then, we create a set of particles $P = (\rho_1, \rho_2, \ldots, \rho_c)$, in which each particle corresponds to a label in L. Each particle ρ_j has three variables $\rho_j^v(t)$, $\rho_j^\omega(t)$ and $\rho_j^\tau(t)$. The first variable, $\rho_j^v(t) \in V$, is used to represent the node v_i being visited by particle ρ_j at time t. The second variable, $\rho_j^\omega \in [\omega_{\min} \quad \omega_{\max}]$ is the particle potential characterizing how much the particle can affect a node at time t, in this paper we set the constants $\omega_{\min} = 0$ and $\omega_{\max} = 1$. The third variable, $\rho_j^\tau(t) \in V$, represents the target node by particle ρ_j at time t, sometimes the particle will be accepted by the node, so $\rho_j^\tau(t) = \rho_j^v(t)$, and some times it will be reject, thus $\rho_j^\tau(t) \neq \rho_j^v(t)$, as we will explain later.

Each node v_i have two variables: $v_i^{\rho}(t)$ and $v_i^{\omega}(t)$. The first, $v_i^{\rho}(t) \in P$ register the particle that owns node v_i at time t. The second variable is a vector $v_i^{\omega}(t) = \{v_i^{\omega_1}(t), v_i^{\omega_2}(t), \ldots, v_i^{\omega_c}(t)\}$ of the same size of L, where each element $v_i^{\omega_j}(t) \in [\omega_{\min} \quad \omega_{\max}]$ corresponds to the level of ownership by particle ρ_j over node v_i. So, at any given time t the particle ρ_j that owns v_i is defined as:

$$v_i^{\rho}(t) = \arg\max_j v_i^{\omega_j}(t). \tag{3}$$

Also, the following equations always holds:

$$\sum_{j=1}^{c} v_i^{\omega_j} = \omega_{max} + \omega_{\min}(c-1). \tag{4}$$

We begin the algorithm by setting the initial level of ownership vector v_i^{ω} by each particle ρ_j as follows:

$$v_i^{\omega_j}(0) = \begin{cases} \omega_{\max} & \text{if } y_i = j \\ \omega_{\min} & \text{if } y_i \neq j \text{ and } y_i \neq \emptyset \\ \omega_{\min} + (\frac{\omega_{\max} - \omega_{\min}}{c}) & \text{if } y_i = \emptyset \end{cases}, \tag{5}$$

which means the nodes corresponding to labeled samples already starts with their ownership set to the corresponding particle with maximum strength, while the other nodes starts with all particles ownership levels equally set.

The initial position of each particle $\rho_j^v(0)$ is set to one of the nodes that they already owns (corresponding to pre-labeled samples) as set in Eq. 5, so the following holds:

$$\rho_j^v(0) = \{v_i | y_i = j\}. \tag{6}$$

The initial potential of each particle is set as:

$$\rho_j^{\omega}(0) = \omega_{\max}. \tag{7}$$

We kept the concept of *random moving* and *deterministic moving* from the original model, where *random moving* means the particle will try to move to any neighbor randomly chosen, and *deterministic moving* means the particle will visit a node that it already owns. Here we extended these rules to handle weighted networks as follows: in *random moving* the particle ρ_j will try to move to any neighbor v_i randomly chosen with probability defined by:

$$p(v_i | \rho_j^v) = \frac{W_{ki}}{\sum_{q=1}^{n} W_{qi}}, \tag{8}$$

where k is the index of the node stored in ρ_j^v, so W_{ki} represents the weight of the edge connecting nodes ρ_j^v and v_i. In *deterministic moving* the particle ρ_j will try to move to any neighbor v_i randomly chosen with probability defined by:

$$p(v_i | \rho_j^v) = \frac{W_{ki} v_i^{\omega_j}}{\sum_{q=1}^{n} W_{qi} v_i^{\omega_j}}, \tag{9}$$

and again, k is the index of the node stored in ρ_j^v. At each iteration, each particle has probability p_{det} of taking deterministic moving and probability $1 - p_{\text{det}}$ of taking random moving, and $0 \leq p_{\text{det}} \leq 1$. Once the deterministic moving or random moving is chosen, the target neighbor $\rho_j^\tau(t)$ will be randomly chosen with probabilities defined by Eq. 8 or Eq. 9 respectively. Figure 1 shows an example situation where a particle is going to choose among three neighbors with different edges weights and ownership levels, the graphics show the probabilities of each node being chosen when using either deterministic or random moving.

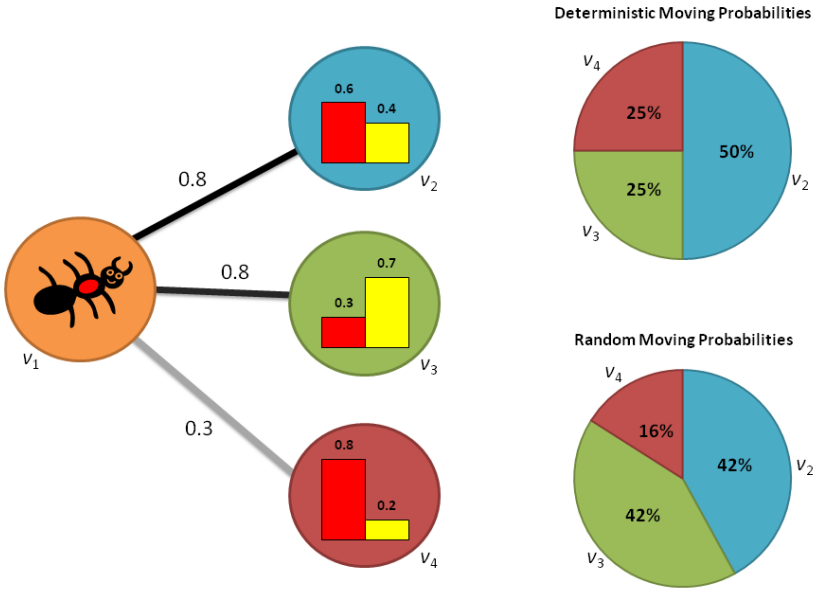


Fig. 1: Deterministic Moving and Random Moving Example Illustration. On the left, the particle ρ_1 is going to choose its target ρ_1^τ among v_2, v_3 and v_4. The graphics inside each node denotes their respective ownership levels $v_i^{\omega_1}$ (red) and $v_i^{\omega_2}$ (yellow), which corresponds to ρ_1 and ρ_2 respectively. On the right, the graphs show the probabilities of choosing each node for either deterministic or random moving.

Regarding the node dynamics, at time t, each ownership level $v_i^{\omega_k}(t)$ of each node v_i, which was chosen by a particle ρ_j as its target $\rho_j^\tau(t)$, is defined as follows:

$$v_i^{\omega_k}(t+1) = \begin{cases} v_i^{\omega_k}(t) & \text{if } y_i \neq \emptyset \\ \max\{\omega_{\min}, v_i^{\omega_k}(t) - \frac{\Delta_v \rho_j^\omega(t)}{c-1}\} & \text{if } y_i = \emptyset \text{ and } k \neq j \\ v_i^{\omega_k}(t) + \sum_{q \neq k} v_i^{\omega_q}(t) - v_i^{\omega_q}(t+1) & \text{if } y_i = \emptyset \text{ and } k = j \end{cases}, \quad (10)$$

where $0 < \Delta_v \leq 1$ is a parameter to control the ownership levels changing speed. If Δ_v takes a low value, the node ownership levels change slowly, while if it takes a high value, the node ownership levels change quickly. Each particle ρ_j will increase their corresponding ownership level $v_i^{\omega_j}$ of the node v_i they are targeting while decreasing the ownership levels (of this same node) that corresponds to the other particles, always respecting Eq. 4. So if the particle already owns the node its targeting, it will reinforce it, else it will increase its own ownership level, possibly becoming the new owner. However, v_i^{ω} is fixed when $y_i \neq \emptyset$ (particle visiting a pre-labeled sample), so their ownership levels never change.

Regarding the particle dynamics, at time t, each particle potential $\rho_j^{\omega}(t)$ is set as:

$$\rho_j^{\omega}(t+1) = v_i^{\omega_j}(t+1) \quad \text{with} \quad v_i(t+1) = \rho_j^{\tau}(t+1), \tag{11}$$

which means every particle ρ_j have their potential ρ_j^{ω} set to the value of its ownership level $v_i^{\omega_j}$ from the node it is currently targeting. This way, a particle will be strong as it is walking in its own neighborhood, but it will become weak if it try to invade another neighborhood. Notice that to handle semi-supervised learning we have made v_i^{ω} fixed when $y_i \neq \emptyset$, so a particle ρ_j can always "recharge" their potential to the maximum ($\rho_j^{\omega} = \omega_{\max}$) when visiting nodes v_i corresponding to pre-labeled samples of its own class ($y_i = j$). Meanwhile, particles ρ_j cannot visit or change ownership levels of nodes v_i corresponding to pre-labeled samples of other class ($y_i \neq j$ and $y_i \neq \emptyset$) no matter how hard they try, and they will become weak ($\rho_j^{\omega} = \omega_{\min}$) every time they try.

Finally, the particle position at time t is defined as follows:

$$\rho_j^{v}(t+1) = \begin{cases} \rho_j^{\tau}(t+1) & \text{if } v_i^{\rho}(t+1) = \rho_j \\ \rho_j^{v}(t) & \text{if } v_i^{\rho}(t+1) \neq \rho_j \end{cases}, \tag{12}$$

with $v_i = \rho_j^{\tau}(t+1)$, which means that after raising its own ownership level on the target node ρ_j^{τ}, the particle ρ_j will move to it if it already owned it or if it became the new owner after that ownership level increase, else, it will stay where it was. Notice that the node owner v_i^{ρ} at any time is defined by Eq. 3.

We have also introduced a "reset" mechanism to take the particles back to one of the nodes corresponding to pre-labeled samples after a pre-defined amount of steps, so the particles could walk around all the pre-labeled nodes. This way, after each r steps, the particles are reset using Eq. 6, and their respective potentials are set to the maximum by Eq. 7.

So, in summary, our algorithm works as follows:

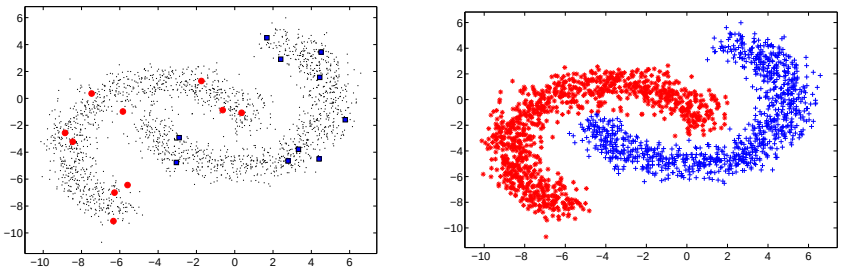
1. Build the affinity matrix W by using Eq. 1,
2. Set nodes ownership levels by using Eq. 5,
3. Set particles initial positions and potentials by using Eq. 6 and Eq. 7 respectively,
4. Repeat steps 5 to 11 until convergence or for a pre-defined number of steps,
5. Select between deterministic moving or random moving,
6. Select the target node for each particle by using Eq. 8 or Eq. 9 for deterministic moving or random moving respectively,

7. Update nodes ownership levels by using Eq. 10,
8. Update nodes ownership flags by using Eq. 3,
9. Update particles potentials by using Eq. 11,
10. Update the particles positions by using Eq. 12,
11. If r steps are reached, reset particles positions and potentials using Eq. 6 and Eq. 7 respectively.

3 Computer Simulations

In this section, we present the simulation results of some semi-supervised classification tasks by using the proposed model with synthetic and real data sets. We also compare our method with the Global and Local Consistency Method [21] for both results accuracy and execution time. For our method, the following parameters were held constant: $p_{\text{det}} = 0.6$ and $\Delta_v = 0.1$. The other parameters (σ and r) were set to their optimal values for each experiment. For the Consistency Method, the parameter $\alpha = 0.99$ was held constant, as the authors did in their original article [21], and σ was set to its optimal value for each experiment.

The first experiment was carried by using the artificial image shown in Fig. 2a, a toy data set with 2000 samples divided in two linearly non-separable classes (1000 samples per class), 20 of these samples (1%) are pre-labeled (10 from each class). Our algorithm was able to accurately classify all the unlabeled data as shown in Fig. 2b.



(a) Toy Data (Banana-Shaped) (b) Classification results

Fig. 2: Classification of the banana-shaped patterns. (a) toy data set with 2000 samples divided in two classes, 20 samples are pre-labeled (red circles and blue squares). (b) classification achieved by the proposed method.

In the second experiment, we have used the Iris data set from the UCI Machine Learning Repository [27], which contains 4 attributes and 3 classes of 50 instances each, where each class refers to a type of iris plant. Both our Particle Method and the Consistency Method were used to perform semi-supervised

classification in this data set. At each set of experiments, some samples (10% to 2%) were randomly chosen as the pre-labeled samples, and the remaining ones were presented unlabeled to both algorithms for classification. The classification results from the algorithms are shown in Figure 3. As our algorithm is non-deterministic there is small differences in the results obtained from different runs, so all the results presented here are the average of 100 runs with the same parameters and the same pre-labeled samples.

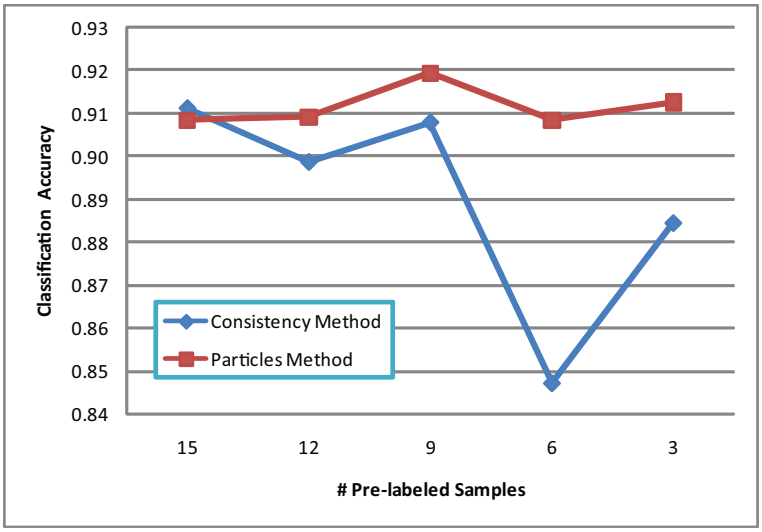


Fig. 3: Classification Accuracy in the Iris data set with different number of pre-labeled samples.

Our third experiment was performed using another real database, the Wine data set, also from the UCI Machine Learning Repository [27]. This data set results from a chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars. The analysis determined the quantities of 13 constituents (attributes) found in these three types of wines, there are 178 samples in total. Again, for each set of experiments, some samples were randomly chosen as the pre-labeled samples while the others were presented to the algorithms unlabeled. The classification results from both algorithms are shown in Figure 4. The presented results, once more, are the average of 100 runs with the same parameters and pre-labeled samples.

By observing Figures 3 and 4 we can notice that the Particle Method outperformed the Consistency Method in most cases, specially with the Wine data set. The major gains are observed when there are fewer pre-labeled nodes, which is another advantage of our method.

We also expect our method to be faster than other graph-based methods, because most of them have order of complexity $O(n^3)$ [11]. For instance, con-

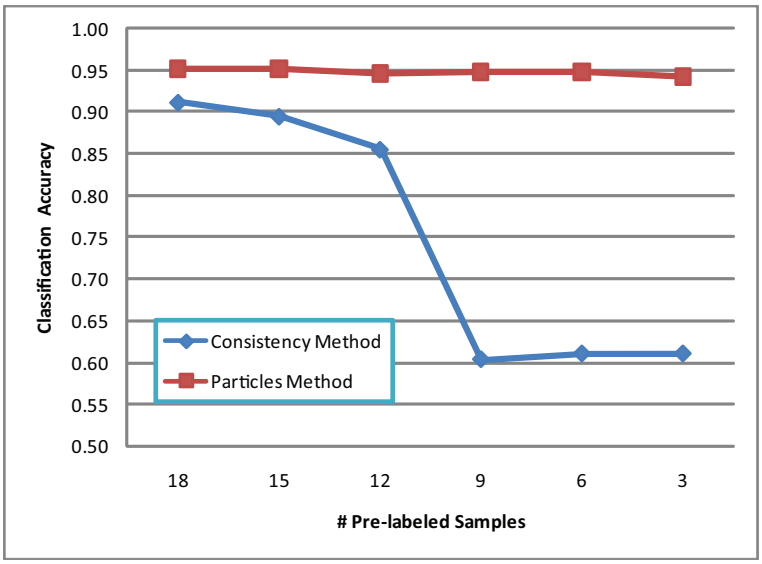


Fig. 4: Classification Accuracy in the Wine data set with different number of pre-labeled samples.

sider the Consistency Method [21], although it usually requires few iterations to converge, each iteration is $O(n^3)$ as they include $n \times n$ matrix multiplications. Also, there is a single step before the iterations, the laplacian normalization, which has high computational cost, for small data sets the cost of this step is even higher than all the iterations together. On the other hand, although our method usually requires thousands of iterations even for small data sets, each iteration is only $O(n \times c)$, so we expect our method to escalate well and to be quite fast for larger data sets.

In order to verify our expectations we have generated some banana-shaped data sets with increasing number of samples, using PRTools [28] function `gendatb` with the variance parameter fixed to 0.8. Then, we let both our method and the Consistency Method classify each of these data sets, running with their optimal parameters. For each data set we randomly chosen 10% of the samples (half from each class) to be the pre-labeled samples input for both algorithms, and finally we have measured the time each algorithm takes to achieve at least 95% correct classification of the remaining samples. All these tests were ran in a regular desktop computer with an Intel Core 2 Quad Processor model Q9450 and 4GB of RAM. Both algorithms were implemented using MATLAB [29]. The results are shown in Fig. 5. Notice that each experiment was repeated 20 times and the values in this graphic are the average time in these 20 runs. By observing the results, it is clear that our algorithm becomes much faster as we increase the data set size, confirming our expectations.

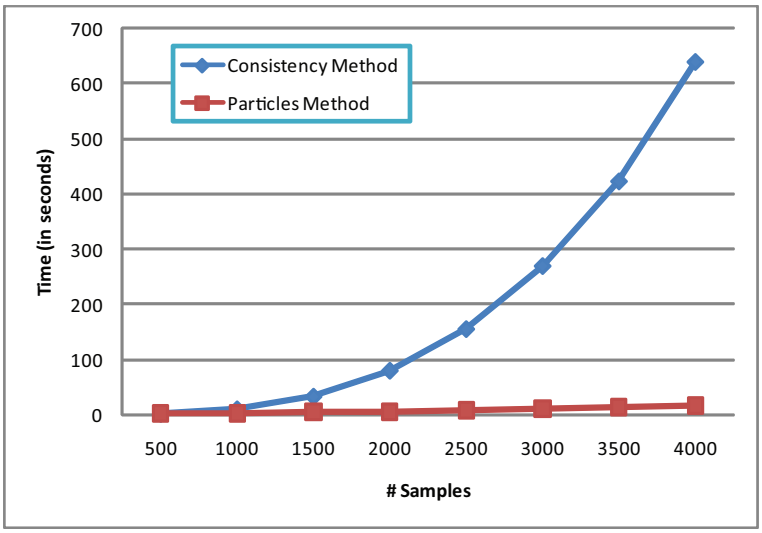


Fig. 5: Time elapsed to reach 95% correct classification with data sets of different sizes.

4 Conclusions

This paper presents a new network-based method for semi-supervised classification using combined random-deterministic walking and competition among particles, where each particle corresponds to a class of the problem. Starting from a small territory corresponding to a few pre-labeled samples, these particles can expand their domain by walking in their neighborhood and preventing other particles from entering in their territory.

Computer simulations were performed in order to check the model viability, and the results shows that our model is a promising mechanism for semi-supervised classification, achieving good classification accuracy in both synthetic and real data. It is important to notice that as the data set size grows our technique becomes much faster than other graph-based methods. This is a desirable feature since semi-supervised classification techniques are usually applied to data sets with large number of unlabeled samples. Also, our method seems to be less affected by the size of pre-labeled set, the ability to learn from less pre-labeled samples is another desirable feature in semi-supervised classification, as pre-labeling samples is the expensive or time consuming task to be avoided.

Another advantage of our method is that it can incorporate unseen data without any modifications in the algorithm, just insert the new nodes in the graph, calculate their connection weights and after some iterations each of these new node will belong to a particle, probably the particle that owns its neighbors. The new nodes can even affect the classification of older nodes, as they can become a new strong link between different neighborhoods.

5 Acknowledgements

This work is supported by the State of São Paulo Research Foundation (FAPESP) and the Brazilian National Council of Technological and Scientific Development (CNPq).

References

1. Newman, M.: The structure and function of complex networks. SIAM Review 45, pp. 167–256. Society for Industrial and Applied Mathematics (2003)
2. Dorogovtsev, S., Mendes, F.: Evolution of Networks: From Biological Nets to the Internet and WWW. Oxford University Press (2003)
3. Bornholdt, S., Schuster, H.: Handbook of Graphs and Networks: From the Genome to the Internet. Wiley-VCH (2006)
4. Newman, M., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E 69, 026113 (2004)
5. Newman, M.: Modularity and community structure in networks. In: Proceedings of the National Academy of Science of the United States of America 103, pp. 8577-8582 (2006)
6. Duch, J., Arenas, A.: Community detection in complex networks using extremal optimization. Physical Review E 72, 027104 (2005)
7. Reichardt, J., Bornholdt, S.: Detecting Fuzzy Community Structures in Complex Networks with a Potts Model. Physical Review Letters 93(21), 218701 (2004)
8. Danon. L, Díaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment 9, P09008 (2005)
9. Quiles, M., Zhao, L., Alonso, R., Romero, R.: Particle competition for complex network community detection. Chaos 18, 033107 (2008).
10. Mitchell, T.: Machine Learning. McGraw Hill (1997)
11. Zhu, X.: Semi-Supervised Learning Literature Survey. Technical Report, Computer Sciences, University of Wisconsin-Madison (2005)
12. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge (2006)
13. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning 39, pp. 103-134 (2000)
14. Fujino, A., Ueda, N., Saito, K.: A hybrid generative/discriminative approach to semi-supervised classifier design. In: AAAI-05, Proceedings of the Twentieth National Conference on Artificial Intelligence, pp. 764–769 (2005)
15. Demiriz, A., Bennet, K., Embrechts, M.: Semi-supervised clustering using genetic algorithms. In: Proceedings of Artificial Neural Networks in Engineering, pp. 809–814. ASME Press (1999)
16. Dara, R., Kremer, S., Stacey, D.: Clustering unlabeled data with SOMs improves classification of labeled real-world data. In: Proceedings of the World Congress on Computational Intelligence (WCCI), pp. 2237-2242 (2002)
17. Blum, A. Mitchell, T.: Combining labeled and unlabeled data with co-training. In: COLT: Proceedings of the Workshop on Computational Learning Theory, pp. 92–100 (1998)
18. Mitchell, T.: The role of unlabeled data in supervised learning. In: Proceedings of the Sixth International Colloquium on Cognitive Science (1999)

19. Vapnik, V.: Statistical Learning Theory. Wiley, New York (2008)
20. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using Gaussian fields and harmonic functions. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 912–919. Morgan Kaufmann, San Francisco (2003)
21. Zhou, D., Bousquet, O., Lal, T., Weston, J., Schölkopf, B.: Learning with local and global consistency. Advances in Neural Information Processing Systems 16, pp. 321–328. MIT Press, Cambdridge (2004)
22. Blum, A., Chawla, S.: Learning from labeled and unlabeled data using graph mincuts. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 19–26. Morgan Kaufmann, San Francisco (2001)
23. Belkin, M., Matveeva, I., Niyogi, P.: Regularization and semisupervised learning on large graphs. In: Conference on Learning Theory, pp. 624–638. Springer (2004)
24. Belkin, M., Niyogi, P., Sindhwani, V.: On manifold regularization. In: Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005), pp. 17–24. Society for Artificial Intelligence and Statistics, New Jersey (2005)
25. Joachims, T.: Transductive learning via spectral graph partitioning. In: Proceedings of International Conference on Machine Learning 2003, pp. 290-297. AAAI Press (2003)
26. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: analysis and an algorithm. Advances in Neural Information Processing Systems 14, pp. 849–856. MIT Press (2001)
27. Asuncion, A., Newman, D.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences, http://www.ics.uci.edu/∼mlearn/MLRepository.html (2007)
28. PRTools: The Matlab Toolbox for Pattern Recognition, http://prtools.org
29. MATLAB, http://www.mathworks.com/products/matlab/