Active Semi-Supervised Learning using Particle Competition and Cooperation in Networks

Fabricio Breve

Abstract— **Both Active Learning and Semi-Supervised Learning are important techniques when labeled data are scarce and unlabeled data are abundant. In this paper, these two machine learning techniques are combined into a new nature-inspired method, which employs particles walking in networks generated from the data. It uses combined competitive and cooperative behavior in order to possess nodes of the network, and thus labeling the corresponding data items. Particles represent labeled nodes, and new particles can be added on the fly to the network as the result of queries (new labels). This built-in mechanism saves a lot of execution time comparing to active learning frameworks, since only nodes affected by the new particles are updated, i.e., the algorithm does not have to be executed again for each new query (or new set of queries). The algorithm naturally adapts itself to new scenarios, i.e., more particles and more labeled nodes. Experimental results on some real-world data sets are presented and the proposed active semi-supervised learning method shows better classification accuracy than its only semi-supervised learning counterpart when the same amount of labeled data is used. Some criteria for selecting the rule to be used to choose data items to be queried are also identified.**

I. Introduction

ACTIVE LEARNING is a form of machine learning in which the learning algorithm is able to interactively query an human specialist (or some other information source) to obtain the labels of selected data points. Its key idea is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. Active learning is specially useful in many modern machine learning problems, where unlabeled data is easy to obtain, but labeling them is an expensive and/or time consuming task [1], [2].

Active Learning algorithms are usually arranged in categories according to how they choose which data points should have their labels queried. Some of the most common approaches includes: *uncertainty sampling*, in which the algorithms query for the labels of data points in which they have less confidence [3]–[6]; *querying by committee*, in which a variety of models are trained on the current labeled data, and then query for the labels of the data points in which they disagree the most [7]–[9]; *expected model change*, in which algorithms query the labels of the instances that would cause the greatest change in the current model [10]; *expected error reduction*, in which algorithms query data points which would most reduce expected error [11], [12]; *expected output variance reduction*, which consists of labeling those points that would minimize the output variance [13]; and *density-weighted methods*, in which the data density information is used in order to choose data points to query [10], [14].

Semi-Supervised Learning is also a subfield of machine learning. It focus on problems where there are lots of easily acquired unlabeled data, but the labeling process is expensive, time consuming, and often requiring the work of human specialists [15]–[17]. Therefore, both active learning and semi-supervised learning try to make the most of unlabeled data. However, semi-supervised learning exploits what the learner thinks it knows about the unlabeled data, sometimes by propagating labels through similar data points or by labeling the data which it has more confidence, and then retraining the algorithm. On the other hand, active learning techniques attempt to explore unknown aspects [2]. Actually, some common semi-supervised learning approaches have an active learning counterpart. For instance, the semi-supervised method self-training uses most confident labels to retrain the algorithm while active learning *uncertainty sampling* approaches query for the less confident labels; co-training semi-supervised methods relies on committees agreements, while active learning *query by committee* methods rely on committees disagreements [1].

Recently, a semi-supervised learning method, based on competition and cooperation among particles walking in a network, was developed [18]. A network is first built from the data using some similarity measure. Then, the particles walk in the network trying to possess its nodes, i.e., marking their territory. Particles carrying the same label, i.e. representing the same problem class, belong to the same team, which means they cooperate with each other to dominate nodes and put their label in them. At the same time, particles carrying different labels belong to different teams and compete against each other. This method highly differs from other graph-based learning semi-supervised methods, since most of them are equivalent to regularization frameworks in which the choice of loss function and regularization terms are their main difference [15].

In the particles method, labels are spread locally as the particles walks from node to node. Labeled nodes are used by particles as references. Each particle has both an exploratory behavior and a defensive behavior. They randomly alternates between them. The exploratory behavior makes the particles propagate their labels to unlabeled nodes, while the defensive behavior keeps the particles around: **a)** labeled nodes which have the same label; and **b)** nodes that their respective team already dominated. Labeled nodes are defined before the

Fabricio Breve is with the Department of Statistics, Applied Mathematics and Computation (DEMAC), Institute of Geosciences and Exact Sciences (IGCE), São Paulo State University (UNESP), Rio Claro, São Paulo, Brazil (email: fabricio@rc.unesp.br).

This work was supported by FAPESP and Fundunesp.

algorithm starts, according to the pre-labeled data points, so there is no retraining. Teams of particles usually have no problem dominating nodes closer to labeled nodes, unless these labeled nodes are outliers. But labeling nodes in the frontier regions or dense regions without labeled nodes can be challenging. The strongest competition usually concentrates on those region nodes. Therefore, the classification accuracy could improve if the particles algorithm was able to select nodes from those specific regions and query for their labels. Moreover, the same classification accuracy could be obtained with less labeled nodes, if some of these labels could be obtained for nodes dynamically chosen as the algorithm runs.

In this paper, the particle competition and cooperation method [18] is extended to realize active learning, so it unites the advantages of both semi-supervised and active learning techniques. The new algorithm needs only one labeled node per class to start. A particle is created for each labeled node. Then, new particles can be added on the fly for data points (nodes) dynamically chosen by the algorithm to have their labels queried. Each query may contain a single or multiple data points. Unlike many other methods, there is no need to restart the algorithm after new queries. The new particles algorithm is able to dynamically adapt itself in order to accommodate new particles and labeled nodes. This saves a lot of execution time, since only nodes affected by the new particles will have their domination levels - and possibly their labels - changed. Therefore, after a new query the stability is quickly reached again.

Two different versions of the proposed method are presented in this paper. In the first one, the algorithm always queries for the most dubious unlabeled network node, i.e., the algorithm queries only by uncertainty. The main advantage here is that, unlike many other base algorithms used in *querying by uncertainty* approaches, the particles algorithm runs only once, i.e., the labeled nodes and corresponding particles are added on the fly. In the second version, the algorithm alternates between querying the most dubious unlabeled network node, and querying the unlabeled network node which is more far away from any labeled node, according to the distances dynamically measured by the particles, while they walk. This second version address the problem of querying outliers, which may happen with *querying by uncertainty* approaches [12], [19]. At the same time, it helps to avoid situations in which a large region of the data set has no labeled nodes, which may lead to wrong label propagation. Each version may work better than the other depending on the data set.

The rest of this paper is organized as follows. The proposed model is described in section II. In Section III, computer simulation results are presented. Finally, Section IV concludes the paper.

II. Model Description

In this section, the proposed active semi-supervised learning method is introduced. It relies on particle competition and cooperation in networks. First, the data set is converted into an unweighted network. For each labeled data item, a particle is created and positioned in the node corresponding to that data item. A subset of particles representing nodes with the same label is called a *team*. Teams compete against each other to possess nodes of the network. Each node has a vector to represent the domination level of each team on it. While teammate particles act cooperatively to possess the nodes of the network, particles in different teams compete against each other trying to avoid rivals from entering their territory. At each iteration of the algorithm, each particle chooses a neighbor node to visit. The chosen neighbor is called *target node*. The particle increases its team domination level and decreases other teams' domination levels on the target node it has chosen. Each particle also has a strength level and a distance table. The strength level lowers or rises according to the dominance of the particle team in the node the particle is visiting. The distance table is used to prevent particles from leaving their neighborhood unprotected. It is calculated dynamically as the algorithm runs.

The network is built from any given data set $\chi = \{x_1, x_2, \ldots, x_l, x_{l+1}, \ldots, x_n\} \subset \mathbb{R}^m$, with its corresponding label set $L = \{1, 2, \ldots, c\}$. The first l points $x_i (i \leq l)$ are labeled as $y_i \in L$. The remaining points $x_u (l < u \leq n)$ are left unlabeled, i.e, $y_u = \emptyset$. An undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is created, in which $\mathbf{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, and $\mathbf{E}$ is the set of edges (v_i, v_j). Each node v_i corresponds to a data point x_i. Two nodes v_i and v_j are connected if v_j is among the k-nearest neighbors of v_i, or vice-versa, using the Euclidean distance. Otherwise, v_i and v_j are disconnected. For each network node $v_i \in \{v_1, v_2, \ldots, v_l\}$, corresponding to a labeled data point $x_i \in \{x_1, x_2, \ldots, x_l\}$, there is a corresponding particle $\rho_i \in \{\rho_1, \rho_2, \ldots, \rho_l\}$ which initial position is at v_i. The algorithm may start with as few as only one labeled node per class. As the algorithm queries for node labels, unlabeled nodes become labeled nodes. Therefore, new particles are created corresponding to these new labeled nodes.

Each particle ρ_j has two variables. The first one is $\rho_j^{\omega}(t) \in [0, 1]$, it holds the particle strength, which defines how much the particle can change node levels at time t. The second variable is a distance table $\rho_{\mathbf{j}}^{\mathbf{d}}(\mathbf{t}) = \{\rho_i^{d_1}(t), \rho_i^{d_2}(t), \ldots, \rho_i^{d_n}(t)\}$. Each element $\rho_j^{d_i}(t) \in [0 \quad n-1]$ holds the distance measured between node v_i and the particle corresponding node (initial position).

Each node v_i has a domination vector $\mathbf{v_i^{\omega}(t)} = \{v_i^{\omega_1}(t), v_i^{\omega_2}(t), \ldots, v_i^{\omega_c}(t)\}$, where each element $v_i^{\omega_\ell}(t) \in [0, 1]$ corresponds to the domination level from team ℓ over node v_i. For each node, the sum of the domination levels is always constant, $\sum_{\ell=1}^{c} v_i^{\omega_\ell} = 1$.

The domination levels are set differently for nodes corresponding to labeled and unlabeled samples. Those nodes corresponding to labeled data points are constant, always fully dominated by the corresponding team. But nodes corresponding to unlabeled data points are variable. They start with all teams' domination levels set equally and they change as particles target them. Therefore, for each node v_i, the

domination table $\mathbf{v_i^\omega}$ is set as follows:

$$v_i^{\omega_\ell}(0) = \begin{cases} 1 & \text{if } y_i = \ell \\ 0 & \text{if } y_i \neq \ell \text{ and } y_i \in L \\ \frac{1}{c} & \text{if } y_i = \emptyset \end{cases} . \quad (1)$$

Notice that (1) is called again each time a node becomes labeled as the result of a query, i.e., new labeled nodes domination levels become constant and fully set to their corresponding team.

Each particle ρ_j is created with its initial position set to its corresponding labeled node, and its initial strength set to the maximum, $\rho_j^\omega(0) = 1$. Particles start knowing only the distance to their corresponding labeled nodes, which is set to zero ($\rho_j^{d_i} = 0$). Other distances are set to the largest possible value ($\rho_j^{d_i} = n - 1$).

At each iteration t, each particle p_j selects a target neighbor node to visit. Each unlabeled node v_i selected as a target has its domination table updated as follows:

$$v_i^{\omega_\ell}(t+1) = \begin{cases} \max\{0, v_i^{\omega_\ell}(t) - \frac{0.1\rho_j^\omega(t)}{c-1}\} \\ \quad \text{if } \ell \neq \rho_j^f \\ v_i^{\omega_\ell}(t) + \sum_{r \neq \ell} v_i^{\omega_r}(t) - v_i^{\omega_r}(t+1) \\ \quad \text{if } \ell = \rho_j^f \end{cases} , \quad (2)$$

where ρ_j^f represents the class label of particle ρ_j. Each particle ρ_j will change its target node v_i by increasing the domination level of its team ($v_i^{\omega_\ell}$, $\ell = \rho_j^f$) while decreasing the domination levels of other teams ($v_i^{\omega_\ell}$, $\ell \neq \rho_j^f$)). Remember that labeled nodes domination tables are fixed, so (2) does not apply to them.

Particles get weaker or stronger according to the domination level of their team on their target nodes. At each iteration, a particle strength is updated as follows: $\rho_j^\omega(t) = v_i^{\omega_\ell}(t)$, where v_i is the target node, and $\ell = \rho_j^f$, i.e., ℓ is the class label of particle ρ_j.

Each particle ρ_j updates its distance table $\rho_j^{d_k}(t)$ at each iteration t as follows:

$$\rho_j^{d_k}(t+1) = \begin{cases} \rho_j^{d_i}(t) + 1 & \text{if } \rho_j^{d_i}(t) + 1 < \rho_j^{d_k}(t) \\ \rho_j^{d_k}(t) & \text{otherwise} \end{cases} , \quad (3)$$

where $\rho_j^{d_i}(t)$ is the distance from the current node to the particle corresponding (initial) node, and $\rho_j^{d_k}(t)$ is the distance from the target node to the particle corresponding node.

Notice that distance calculation is a dynamical process: particles have limited knowledge of the network; they do not know the nodes connection patterns. Unknown distances are calculated as the particles walk. Distances are updated as particles naturally find shorter paths.

Each particle ρ_j chooses its target node v_i among the neighbors of its current node. The probability of choosing each neighbor v_i is defined according to: a) the particle team domination on each neighbor node, $\rho_j^{\omega_\ell}$, and b) the inverse of neighbor node distance, $\rho_j^{d_i}$, to the particle initial position, v_j, as follows:

$$p(v_i|\rho_j) = \frac{W_{qi}}{2\sum_{\mu=1}^n W_{q\mu}} + \frac{W_{qi} v_i^{\omega_\ell}(1+\rho_j^{d_i})^{-2}}{2\sum_{\mu=1}^n W_{q\mu} v_\mu^{\omega_\ell}(1+\rho_j^{d_\mu})^{-2}}, \quad (4)$$

where q is the index of the node being visited by particle ρ_j and ℓ is the class label of particle ρ_j. A particle actually visits the target node only if, after applying (2), its team domination level on that node is higher than those from all other teams; otherwise, a shock happens and the particle stays at the current node until the next iteration.

Finally, after the last iteration of the algorithm, each unlabeled node is labeled after the team which has the highest domination level on it, i.e., $y_i = \arg\max_\ell v_i^{\omega_\ell}(t)$.

The average maximum domination levels of the nodes ($\langle v_i^{\omega_\ell} \rangle$, $\ell = \arg\max_q v_i^{\omega_q}$) can be used to identify when the algorithm reaches a fair level of stability. This value does not converge, as there is always some dispute on the nodes in classes' frontiers. But one may set a stop criterion for when there is no increase in this measure for a considerable amount of iterations. As a rule of thumb, $\frac{100.n}{l}$ iterations is usually enough, where n is the network size and l is the current amount of particles in the network. This criterion may be also used to define when it is time to query more labels and create new particles, as follows.

Every time the system reaches a fair level of stability, it queries for another node label and creates another particle corresponding to this new labeled node. This procedure is repeated until the target amount of labeled nodes is reached. There are two version of the algorithm. They differ in the rule used to choose the node which label will be queried. The first version, which will be called AL-PCC v1 from now, selects the unlabeled node that the algorithm is most uncertain about which label it should have, i.e., the node in which the algorithm has less confidence on the label currently assigned to it. The second version, AL-PCC v2, alternates between querying the most uncertain unlabeled network node and querying the unlabeled node which is more far away from any labeled node, according to the distances in the particles distance tables, dynamically built while they walk.

In order to select the network node with the most uncertain label at any given time, first the degree of uncertainty has to be calculated in each node using the following equation:

$$u_i(t) = \frac{v_i^{\ell**}(t)}{v_i^{\ell*}(t)}, \quad (5)$$

where

$$v_i^{\ell*}(t) = \arg\max_\ell v_i^\ell(t), \quad (6)$$

$$v_i^{\ell**}(t) = \arg \max_{\ell, \ell \neq v_i^{\ell*}(t)} v_i^\ell(t), \quad (7)$$

and $u_i \in [0 \quad 1]$, where $u_i = 0$ means completely confidence in the label given to the node, while $u_i = 1$ means the node label is completely undefined among two or more classes.

Then, the node with the most uncertain label, which is going to be queried, is defined using:

$$q(t) = \arg\max_i u_i(t). \quad (8)$$

As for the node which is more far away from any labeled node, first the distance from each node to its closest labeled node is defined according to the measures taken by all the particles:

$$s_i(t) = \min_j \rho_j^{d_i}(t). \tag{9}$$

Then, the node which is more far away from any labeled node is defined using:

$$q(t) = \arg\max_i s_i(t). \tag{10}$$

In summary, each time $\frac{100.n}{l}$ iterations pass without increase in $\langle v_i^{\omega_\ell} \rangle$, $\ell = \arg\max_q v_i^{\omega_q}$, AL-PCC v1 queries for a new label using (8), while AL-PCC v2 queries for a new label alternating between (8) and (10). For each new labeled node, (1) is applied and a new particle is created, corresponding to that node.

III. Computer Simulations

In this section, some computer simulation using some real-world data sets are presented in order to show the effectiveness of the proposed method. In each simulation, the correct classification rate obtained by the original particle competition and cooperation method (PCC) is compared with those achieved by the two versions of the proposed method (AL-PCC v1 and AL-PCC v2). The following parameters are fixed for the PCC method: $p_{grd} = 0.5$ and $\Delta_v = 0.1$. For the three methods, $k = 5$ is fixed. These are optimal or near optimal parameter values for most data sets. They were obtained by empirical optimization using the grid method.

In each experiment, the PCC method is executed using 1% to 10% data items randomly chosen to be pre-labeled (from the algorithm start), as it requires. On the other hand, for both versions of AL-PCC, we randomly choose only one data item per class to be pre-labeled. Then each time the algorithms reaches stability, they query the label of another node, chosen according to their specific rules. That is repeated until the defined amount of labeled items (1% to 10%) is reached. Notice that for the Iris and Wine data sets (Figures 1 and 2), simulations with only 1% and 2% labeled nodes are skipped, as a single labeled node per class would already reach 2% labeled nodes, leaving no room for queries. Each point in the graphics from Figures 1 to 7 is the average of 100 executions with different pre-labeled nodes.

Figure 1 shows the classification performance comparison when the methods are applied to the Iris data set [20]. Both AL-PCC v1 and AL-PCC v2 work better than the original PCC in this data set. AL-PCC v1 is the best. This is probably because Iris data set classes are fairly well separated. One of them is completely separated from the others, so there is no point in labeling new nodes in that class, and creating new particles there. The other two classes have some overlapping. AL-PCC v1 works better because it is probably querying most of those overlap nodes.

In Figure 1, the correct classification rate when the methods are applied to the Wine data set [20] is show. Again, AL-PCC v1 and AL-PCC v2 work better than the original

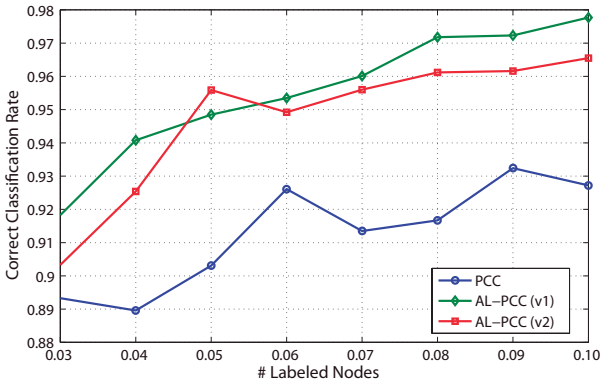

Fig. 1. Correct classification rate comparison when the methods are applied to the Iris data set [20] with different amounts of labeled nodes.

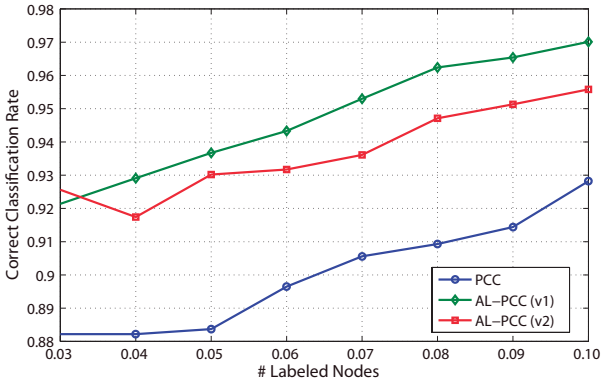

Fig. 2. Correct classification rate comparison when the methods are applied to the Wine data set [20] with different amounts of labeled nodes.

PCC. AL-PCC v1 is the best again. The explanation is that Wine data set also has well-separated classes, even more than Iris data set. Letting the algorithm concentrate on querying the frontier nodes probably sets the edge in the classification rate here.

Figure 3 shows the correct classification rate comparison when the three methods are applied to the Digit1 data set [16]. Once more, both AL-PCC v1 and AL-PCC v2 work better than the original PCC. AL-PCC v1 is the best again. Using 10% labeled nodes, AL-PCC v1 reaches almost perfect classification rate, with 99.98% accuracy.

In Figure 4, the classification performance comparison when the methods are applied to the USPS data set [16] is show. Here AL-PCC v1 is actually worse than the original PCC method. But AL-PCC v2 works better than both. This is one of those cases where querying only the most uncertain nodes leads to the labeling of outliers, which actually decreases the classification performance. AL-PCC v2 works better in this case because it also queries dense regions even if there is no uncertainty. This helps to define the frontiers in dense regions, minimizing the effects of labeling outliers.

Figure 5 shows the correct classification rate when the methods are applied to the COIL_2 data set [16]. These

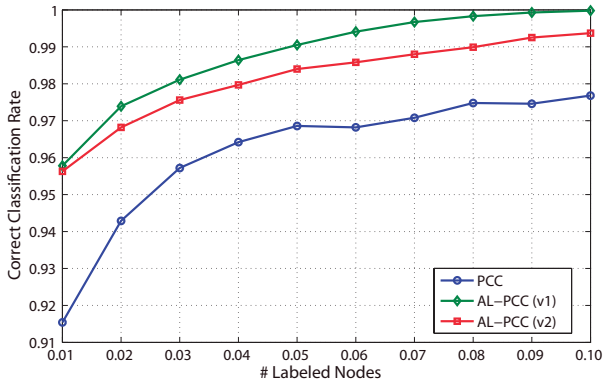

Fig. 3. Correct classification rate comparison when the methods are applied to the Digit1 data set [16] with different amounts of labeled nodes.

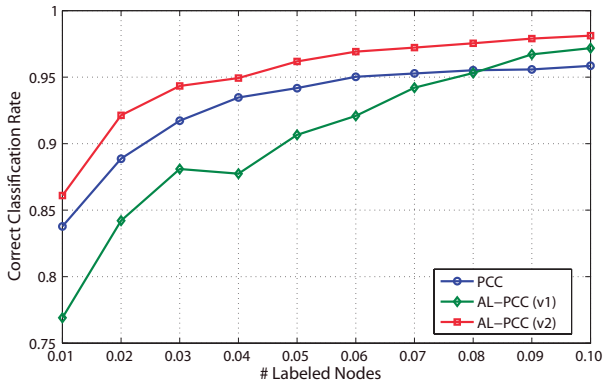

Fig. 4. Correct classification rate comparison when the methods are applied to the USPS data set [16] with different amounts of labeled nodes.

results are similar to the last case, AL-PCC v2 is the best, followed by the original PCC method. AL-PCC v1 has poor performance, not comparable to the others. Querying nodes sparsely distributed in the feature space is probably the key for good performance here as well.

Figure 6 shows the performance comparison when the methods are applied to the g241c data set [16]. This case is different than all the others. Here AL-PCC v1 is the best, followed by the original PCC method. AL-PCC v2 is the worst for the first time. This behavior indicates that classes may be severely mixed in a large portion of the network (and the features space). Therefore, choosing a single representative for a large dense region, as AL-PCC v2 does, may be a bad choice.

Finally, Figure 7 shows the correct classification rate when the methods are applied to the COIL data set [16]. As expected, AL-PCC v2 is the best. The same happened in the binary version of COIL (Figure 5). However, in this case AL-PCC v1 is not so bad. Actually, it nearly matches the performance of the original PCC method. This behavior may have an explanation on the presence of more frontiers, as the COIL data set has 6 classes, while COIL_2 is divided into only 2 classes.

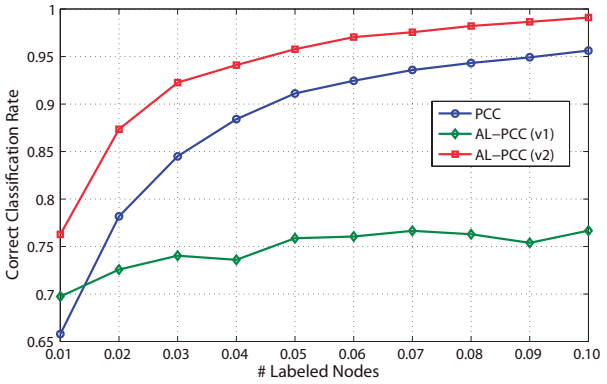

Fig. 5. Correct classification rate comparison when the methods are applied to the COIL_2 data set [16] with different amounts of labeled nodes.

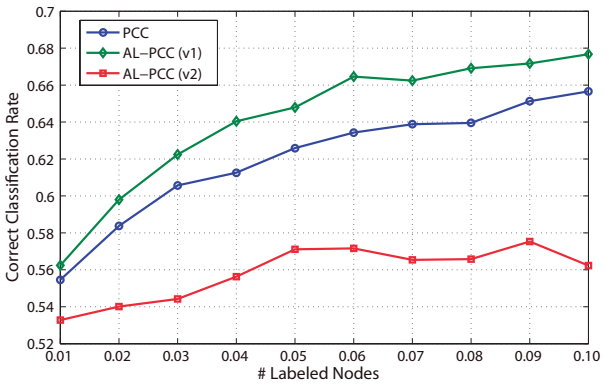

Fig. 6. Correct classification rate comparison when the methods are applied to the g241c data set [16] with different amounts of labeled nodes.

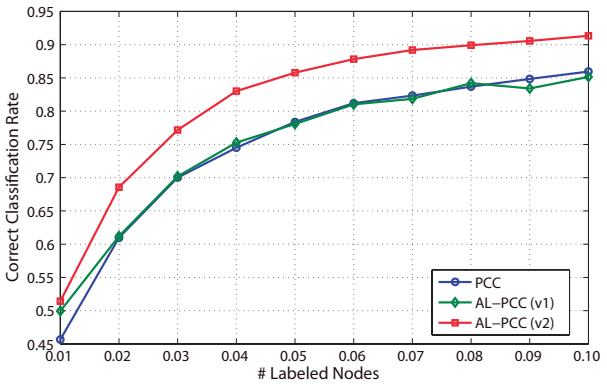

Fig. 7. Correct classification rate comparison when the methods are applied to the COIL data set [16] with different amounts of labeled nodes.

IV. Conclusions

This paper proposes a new nature-inspired method for active semi-supervised classification using teams of walking particles. These teams compete against each other for the possession of network nodes. Each team corresponds to a class of the problem. The method combines both active semi-supervised learning and active learning in a novel way. Each particle in the network corresponds to a labeled node. New particles are created on the fly as the algorithm queries for the labels of dynamically selected nodes. This built-in mechanism saves a lot of execution time comparing to most active learning frameworks, which re-execute the algorithms from start for each new query. In the proposed method, the algorithm continues its execution and naturally adapts itself as new labels are added. Only nodes affected by the new particles need to be updated, so the algorithm quickly reaches a new equilibrium after new particles are inserted, as the result of new queries.

Some computer simulations using real-world data sets were performed. The proposed method shows better classification accuracy than its only semi-supervised learning counterpart when the same amount of labeled data is used. The first version of the algorithm queries the labels of nodes in which there is more uncertainty. It works better in situations where the classes' frontiers do not have many outliers. The second version alternately queries most uncertainty nodes and nodes which are more distant than any labeled node, according to dynamical measures taken by the particles. This version works better in situations where classes' frontiers are not so well defined and there are many outliers. In the current stage, it is hard to tell beforehand which version will work better in a specific data set.

As future work, we intend to use some heuristics to automatically choose nodes to query according to characteristics the algorithm may detect from the data sets. Therefore, the user would not have to select a version of the algorithm a priori. The algorithm would be able to query nodes that are more dubious, more far away, or any other rule. Maybe it could even switch the rule its using on the fly, or decide to use different proportions (other than $50/50$) for each rule, instead of simply alternating between them. Moreover, it would be interesting to use historical information to define uncertainty, i.e., not only current domination levels, but also the history of changes in domination levels. The intensity of the dispute for a node may also indicate its uncertainty level, and this may be more accurate than considering only the information available in a given instant of time.

References

[1] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, 2012.

[2] F. Olsson, "A literature survey of active machine learning in the context of natural language processing," Swedish Institute of Computer Science, Box 1263, SE-164 29 Kista, Sweden, Tech. Rep. T2009:06, April 2009.

[3] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '94. New York, NY, USA: Springer-Verlag New York, Inc., 1994, pp. 3–12.

[4] D. Cohn, R. Ladner, and A. Waibel, "Improving generalization with active learning," in *Machine Learning*, 1994, pp. 201–221.

[5] G. Schohn and D. Cohn, "Less is more: Active learning with support vector machines," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 839–846.

[6] T. Scheffer, C. Decomain, and S. Wrobel, "Active hidden markov models for information extraction," in *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, ser. IDA '01. London, UK, UK: Springer-Verlag, 2001, pp. 309–318.

[7] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of the fifth annual workshop on Computational learning theory*, ser. COLT '92. New York, NY, USA: ACM, 1992, pp. 287–294.

[8] R. Liere and P. Tadepalli, "Active learning with committees for text categorization," in *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence*, ser. AAAI'97/IAAI'97. AAAI Press, 1997, pp. 591–596.

[9] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997.

[10] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *In Advances in Neural Information Processing Systems (NIPS*. MIT Press, 2008, pp. 1289–1296.

[11] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *J. Artif. Int. Res.*, vol. 4, no. 1, pp. 129–145, Mar. 1996.

[12] N. Roy and A. McCallum, "Toward optimal active learning through sampling estimation of error reduction," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 441–448.

[13] A. I. Schein and L. H. Ungar, "Active learning for logistic regression: an evaluation," *Mach. Learn.*, vol. 68, no. 3, pp. 235–265, Oct. 2007.

[14] Z. Xu, R. Akella, and Y. Zhang, "Incorporating diversity and density in active learning for relevance feedback," in *Proceedings of the 29th European conference on IR research*, ser. ECIR'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 246–257.

[15] X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep. 1530, 2005.

[16] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, MA: The MIT Press, 2006.

[17] S. Abney, *Semisupervised Learning for Computational Linguistics*. CRC Press, 2008.

[18] F. Breve, L. Zhao, M. Quiles, W. Pedrycz, and J. Liu, "Particle competition and cooperation in networks for semi-supervised learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 9, pp. 1686 –1698, sept. 2012.

[19] X. Zhu, J. Lafferty, and Z. Ghahramani, "Combining active learning and semi-supervised learning using gaussian fields and harmonic functions," in *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003, pp. 58–65.

[20] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml