Particle Competition and Cooperation in Networks for Semi-Supervised Learning with Concept Drift

Fabricio Breve*
São Paulo State University (UNESP)
Rio Claro, São Paulo, Brazil
Email: fabricio@rc.unesp.br

Liang Zhao
University of São Paulo (USP)
São Carlos, São Paulo, Brazil
Email: zhao@icmc.usp.br

***Abstract*—Concept drift is a problem of increasing importance in machine learning and data mining. Data sets under analysis are no longer only static databases, but also data streams in which concepts and data distributions may not be stable over time. However, most learning algorithms produced so far are based on the assumption that data comes from a fixed distribution, so they are not suitable to handle concept drifts. Moreover, some concept drifts applications requires fast response, which means an algorithm must always be (re)trained with the latest available data. But the process of labeling data is usually expensive and/or time consuming when compared to unlabeled data acquisition, thus only a small fraction of the incoming data may be effectively labeled. Semi-supervised learning methods may help in this scenario, as they use both labeled and unlabeled data in the training process. However, most of them are also based on the assumption that the data is static. Therefore, semi-supervised learning with concept drifts is still an open challenge in machine learning. Recently, a particle competition and cooperation approach was used to realize graph-based semi-supervised learning from static data. In this paper, we extend that approach to handle data streams and concept drift. The result is a passive algorithm using a single classifier, which naturally adapts to concept changes, without any explicit drift detection mechanism. Its built-in mechanisms provide a natural way of learning from new data, gradually "forgetting" older knowledge as older labeled data items became less influent on the classification of newer data items. Some computer simulation are presented, showing the effectiveness of the proposed method.**

***Index Terms*—Concept Drift, Semi-Supervised Learning, Particle Competition and Cooperation, Machine learning.**

I. Introduction

Concept drift is a problem of increasing importance to machine learning and data mining. Data sets under analysis are no longer only static databases, but also data streams in which concepts and data distributions may not be stable over time. Some examples include climate prediction, fraud detection, energy demand and many other real-world applications. However, most algorithms produced so far are based on the assumption that data comes from a fixed distribution, i.e., they are not suited to handle concept drift. Applying them to such problems would inevitable result in low performance. In order to address the problem of learning from data streams with concept drift, one has to implement some kind of incremental learning which address two conflicting objectives: retaining previously learned knowledge that is still relevant, and replacing any obsolete knowledge with current information. This problem is also refereed in the literature as *learning in nonstationary environments* [1]–[5].

The term "concept drift" was coined by [6], who formulated the problem of incremental learning from noisy data and presented STAGGER, an adaptive learning algorithm which was one of the earliest attempts to solve this problem. Concept drift refers to a nonstationary learning problem over time. When all data is sampled from the same data distribution, we say the concept is stable. If for any two time points the data distribution is different, we say that there is a concept drift [1]. Nowadays, concept drift algorithms are commonly classified in two major groups: active or passive algorithms. The active algorithms are also called trigger based algorithms, as they have some kind of detector which indicates a need for model change, i.e., the learner adaptivity is triggered [7]–[11]. On the other hand, passive algorithms, also called evolving algorithms, do not detect changes, the drift is simply assumed, i.e., the learner evolves independently of triggers or detectors. Most techniques in this category are classifier ensembles [2], [12]–[16] where adaptivity is achieved by assigning different weights to individual models output at each instant in time.

In some applications, the large amount of data generated makes it impossible to collect and store it for later analysis. Moreover, some applications, such as fraud detection in credit cards and natural disaster forecast, requires fast response. In order to proper detect and learn the concept drifts in these conditions, the incremental learning algorithms requires that the data stream is constantly fed to it, either online or in batches. Typically, to achieve good performance regardless of shifting in concepts, an algorithm must always be (re)trained with the latest available data. This may pose another difficult in applications where the process of labeling data is expensive and/or time consuming when compared to unlabeled data acquisition. In such scenarios, only a small fraction of the incoming data may be effectively labeled. Semi-supervised learning methods may be useful in these cases, as they are specifically designed to learn from data sets where there are lots of unlabeled data and only a few labeled data items [17]–[19].

Semi-supervised learning methods may be split in some

*F. Breve is also with the University of São Paulo (USP), São Carlos, São Paulo, Brazil, Email: fabricio@icmc.usp.br

categories, like generative models [20], [21], cluster-and-label techniques [22], [23], co-training and tri-training techniques [24]–[27], low-density separation models [28], and graph-based methods, which is the most active category in the recent years, and it includes methods like Mincut [29], Local and Global Consistency [30], label propagation techniques [31], [32], among others. Though many semi-supervised graph-based methods were developed, most of them are similar and they may be seen as regularization framework [17], differing only in the particular choice of the loss function and the regularizer [29], [30], [33]–[36]. Moreover, most of these methods are also based on the assumption that the data is static. Therefore, the semi-supervised learning with concept drift problem is still an open challenge in machine learning.

Recently, a semi-supervised learning method based on particle competition and cooperation in networks was developed [37]. In this method, particles walk in the network trying to possess its nodes, i.e., marking their territory. Particles representing the same problem class/label belong to the same team, and they cooperate with their teammates to dominate nodes for their respective team (class/label). At the same time, particles representing different classes/labels belong to different teams and compete against each other. Like most semi-supervised learning algorithms, the particle competition and cooperation method was created on the assumption that the data sets are always static. In this paper, we extended that approach to handle data streams and concept drift. These objectives were achieved by introducing rules to create new nodes, to eliminate older nodes and to rearrange the connections in the network, as new data examples becomes available in small batches. We also introduce rules to create particles for each new labeled data item, while eliminating old particles after they fulfill their purpose of spreading their classes labels, among other improvements.

The proposed algorithm is different from most other concept drift algorithms. It is a semi-supervised learning graph-based algorithm, which takes advantage of both labeled and unlabeled data. It receives the data items in small batches and it is specially suited for gradual or incremental changes in concept [1]. It falls in the category of passive algorithms, as it naturally adapts to concept changes, without any explicit drift detection mechanism. And differently from most of other methods in this category, it does not use ensembles of classifiers, it is rather a single classifier approach which also does not include any explicit retraining process. Its built-in mechanisms provide a natural way of learning from new data, gradually "forgetting" older knowledge as older labeled data items became less influent on the classification of newer data items.

The rest of this paper is organized as follows. The proposed model is described in section 2. In Section 3, computer simulation results are presented. Finally, Section 4 concludes the paper.

II. Model Description

In this section we describe the proposed method. The data items are transformed into nodes of an undirected and unweighed network as they arrive. For each labeled data item, a particle is generated and put in the corresponding node. A group of particles with the same label is called a *team*. Each node in the network has a vector of elements corresponding to the domination level of each team of particles over that node. As the system runs, particles use a random-greedy rule to choose a neighbor to visit. They increase the domination level of their respective team in the chosen node, at the same time that they decrease the domination levels of other teams. Teams of particles will act cooperatively trying to dominate as many nodes as possible while preventing intrusion of other teams. After some iterations, unlabeled nodes will be labeled after the team of particles which have dominated them.

The first step is converting the data set into a network. Each data item is converted into a network node and connected to its k-nearest neighbors according to the Euclidean distance. Since the algorithm receives new data items in batches, it will reconfigure the network to reflect these changes each time a new batch arrives. Basically, the algorithm will always keep a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of current nodes and $\mathbf{E}$ is the set of current edges (v_i, v_j), which can also be represented by an adjacency matrix $\mathbf{W}$:

$$W_{ij} = \begin{cases} 1 & \text{if } x_j \text{ is among the } k\text{-nearest neighbors} \\ & \quad \text{of } x_i \text{ or vice-versa} \\ 0 & \text{otherwise} \end{cases}, \quad (1)$$

so W_{ij} specifies whether there is an edge between the pair of nodes v_i and v_j. The algorithm keeps a vector $Y = \{y_1, y_2, \ldots, y_n\}$, where y_i takes the label of the node v_i if it is known, or 0, otherwise. The label set is defined as $L = \{1, 2, \ldots, c\}$, so a number is assigned to each of the c classes and 0 is reserved for nodes which label is unknown.

The network also has a maximum size v_{max} that it is allowed to reach. It is important that the network does not grow indefinitely, because it would not only slower the algorithm, but the older nodes would affect the classification of newer nodes, which is not desirable when classifying nonstationary data streams. Therefore, as the maximum size is reached, the oldest nodes receive their definitive label and they are removed from the network as the new nodes are created. The labeling process will be explained later.

For each labeled node v_i (i.e., $y_i \neq 0$) which arrives in the network, a particle ρ_j is generated and its initial position is set at the node v_i. Particles generated for nodes with the same class label form a *team* and cooperate among themselves to compete with other teams. So, each team represents a class of the network. There is also a limit ρ_{max} in the amount of the particles a network can have. Therefore, when the limit is reached, the oldest particles will be removed as new particles are created. Each particle ρ_j has a variable $\rho_j^{\omega}(t) \in [0, 1]$, called *particle strength*, which indicates how much the particle can affect nodes domination levels at time t.

Each node v_i has a vector variable $\mathbf{v_i^{\omega}(t)} = \{v_i^{\omega_1}(t), v_i^{\omega_2}(t), \ldots, v_i^{\omega_c}(t)\}$ called *domination levels*, and each element $v_i^{\omega_\ell}(t) \in [0, 1]$ corresponds to the level of domination of team ℓ over node v_i. At each node, the sum of the domination levels

is always constant, as follows:

$$\sum_{\ell=1}^{c} v_i^{\omega_\ell} = 1, \tag{2}$$

because particles increase the node domination level of their own team and, at the same time, decreases the other teams' domination levels.

Each node v_i has an initial value of its domination vector $\mathbf{v_i^\omega}$ set as follows:

$$v_i^{\omega_\ell}(t) = \begin{cases} \frac{1}{c} & \text{if } y_i = 0 \\ 1 & \text{if } y_i = \ell \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

i.e., for each unlabeled node ($y_i = 0$), the domination levels of all particle teams are set to the same value $\frac{1}{c}$, where c is the amount of teams (classes); and for each labeled node ($y_i \neq 0$), the domination level of the dominating team is set to the highest value 1, while the domination levels of other teams are set to the lowest value 0.

Each particle has its initial position set to its corresponding labeled node, and their initial strength is set as follows:

$$\rho_j^\omega(t) = 1, \tag{4}$$

i.e., each particle starts with maximum strength.

At each iteration, a particle randomly chooses any of its neighbors to target with higher probability to the nodes in which its team have higher domination level. Therefore, the particle ρ_j chooses its target node v_i with probabilities defined according to its team domination level on that neighbor $\rho_j^{\omega_\ell}$, as follows:

$$p(v_i|\rho_j) = (1-\alpha)\frac{W_{qi}}{\sum_{\mu=1}^{n} W_{q\mu}} + \alpha\frac{W_{qi}v_i^{\omega_\ell}}{\sum_{\mu=1}^{n} W_{q\mu}v_i^{\omega_\ell}}, \tag{5}$$

where q is the index of the node being visited by particle ρ_j and $\ell = \rho_j^f$, where ρ_j^f is the class label of particle ρ_j. $0 < \alpha < 1$ is a parameter that defines the weight of team domination levels and distances on the probabilities. When α is low, exploratory behavior dominates; and when α is high, defensive behavior dominates. Best classification performance is achieved when there is an equilibrium between exploratory and defensive behavior. Therefore, we usually set $\alpha \approx 0.5$.

Notice that particles also move when the node they are staying is going to be eliminated. In this case, it will be moved randomly to any of its neighbors, and its strength will be set according to (7).

Teams of particles compete for owning the network nodes. When a particle moves to another node, it increases the domination level of its team in that node, at the same time that it decreases the domination level of the other teams in that same node. The exception are the labeled nodes, which domination levels are fixed. Thus, for each selected target node v_i, the domination level $v_i^{\omega_\ell}(t)$ is updated as follows:

$$v_i^{\omega_\ell}(t+1) = \begin{cases} \max\{0, v_i^{\omega_\ell}(t) - \frac{\Delta_v \rho_j^\omega(t)}{c-1}\} \\ \quad \text{if } y_i = 0 \text{ and } \ell \neq \rho_j^f \\ v_i^{\omega_\ell}(t) + \sum_{q\neq\ell} v_i^{\omega_q}(t) - v_i^{\omega_q}(t+1) \\ \quad \text{if } y_i = 0 \text{ and } \ell = \rho_j^f \\ v_i^{\omega_\ell}(t) \quad \text{if } y_i \neq 0 \end{cases}, \tag{6}$$

where $0 < \Delta_v \leq 1$ is a parameter to control changing rate of the domination levels and ρ_j^f represents the class label of particle ρ_j. If Δ_v takes a low value, the node domination levels change slowly, while if it takes a high value, the node domination levels change quickly. Each particle ρ_j increases the domination level of its team ($v_i^{\omega_\ell}$, $\ell = \rho_j^f$) at the node v_i when it targets it, while it decreases the domination levels of other teams in this same node ($v_i^{\omega_\ell}$, $\ell \neq \rho_j^f$), always respecting the conservation law defined by Eq. (2). The domination levels of all labeled node v_i^ω are always fixed, as defined by the third case expressed by Eq. (6).

Particles get stronger when they move to nodes being dominated by their own team and they get weaker when they move to nodes dominated by other teams. Thus, at each iteration t, a particle strength $\rho_j^\omega(t)$ is updated as follows:

$$\rho_j^\omega(t+1) = v_i^{\omega_\ell}(t+1), \tag{7}$$

where v_i is the target node, and $\ell = \rho_j^f$, i.e., ℓ is the class label of particle ρ_j. Therefore, each particle ρ_j has its strength ρ_j^ω set to the value of its team domination level $v_i^{\omega_j}$ of the node v_i.

When a particle tries to move to the target node, it may be either accepted or rejected due to the competition mechanism. After modifying the target node domination levels and updating its own strength, the particle will be accepted in the target node only if the domination level of its team is higher than others; otherwise, a shock happens and the particle stays in the current node until the next iteration. There is an exception, though: the shock rule is not applied when the particle is moving because its current node is being eliminated.

The distance tables introduced in [37] are no longer needed in this new method. Old particles now move guided only by its team domination on its neighborhood. This also allows the particles to naturally migrate to other regions as the concepts drift, until they are removed to give place to a new particle.

Unlabeled nodes are labeled according to the team which has the highest domination level on them:

$$y_i = \arg\max_\ell v_i^{\omega_\ell}(t). \tag{8}$$

The final label of any node is given when they are removed from the network, i.e., a node stays in the network for a certain amount of time (iterations) so the teams of particles can fight for it. When it is time for a node to be removed from the network (to give its place to a new node created for a new data item), it receives its final label according to the team who owns it at that time. Labeled nodes also stay in the network spreading their label, until they are eliminated after the predefined amount of time.

Overall, the proposed algorithm can be outlined as showed in Algorithm 1.

Algorithm 1: The Proposed Algorithm

```
for each new batch of data items do
    for each new data item do
        if network size already reached maximum size then
            if the oldest node is unlabeled then
                Label the oldest node by using Eq. (8);
            if there is a particle in the oldest node then
                Move particle to any of the node's neighbors
                randomly;
            Remove the oldest node;
        Add a new network node for the new data item;
        Set new node domination levels by using Eq. (3);
        if the new node is labeled then
            if the amount of particles already reached
            maximum then
                Eliminate oldest particle;
            Create a new particle;
            Set new particle initial position at the new node;
            Set new particle strength by using Eq. (4);
    repeat
        for each particle do
            Select the target node by using Eq. (5);
            Update target node domination levels by using Eq.
            (6);
            Update particle strength by using Eq. (7);
            Move particle to the target node;
            if chock occurs then
                Move particle back to the previous node;
    until next batch of data items arrives;
```

III. Computer Simulations

In this section, we present simulation results to show the effectiveness of our method in semi-supervised classification with concept changes. The following parameters are used in these simulations: $\Delta_v = 0.1$ and $\alpha = 0.5$. These values were obtained by empirical optimization using the grid method.

For the first set of simulations, we generate data sets with 4 equiprobable classes with Gaussian distribution, by using PRTools [38] function `gendats`. For each experiment, $50,000$ data items (n) are generated in 500 batches, 100 data items in each of them. The 4 Gaussian kernels are always positioned in the border of an imaginary circumference, as showed in Figure 1. These kernels move clockwise from batch to batch, completing a whole lap when the last batch is generated.

The data items are presented to the algorithm in batches of 100 elements. 10% of these data items are presented with their respective labels and the remaining are presented unlabeled (a typical semi-supervised learning scenario). Each batch of elements will be converted into nodes of the network as they arrive, and after some time they will receive their final labels as they are replaced by new nodes (generated from new data items). The exception are the last batches, which are labeled after some time but are not removed from the network, as there are no new data items to replace them.

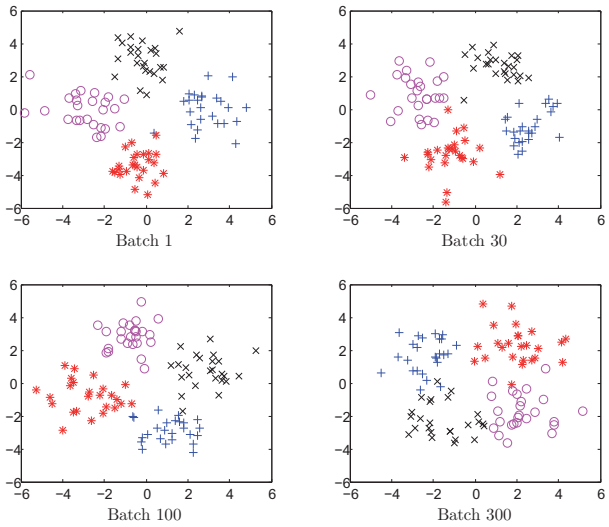

Fig. 1. Snapshots of the data sets

In this kind of problem, we have to work under some constraints, the amount of processing is limited by the interval between each batches arrival and by the hardware executing the algorithm. Therefore, we have fixed $100,000$ as the amount of particle movements which can be processed in the interval between each batch arrival. Notice that, the computational cost of processing a single particle movement is given by the amount of neighbors the node it is currently staying has [37]. We have control over the network average node degree by defining how many k-nearest neighbors are connected to the nodes. In these experiments, we fixed $k = 5$. Therefore, the time to process 100,000 particle movements will be fairly constant no matter the network size or amount of particles walking in the network.

Since we have some flexibility in the maximum network size ($v_{\max}$) and the maximum amount of particles ($\rho_{\max}$), we can tune these values in order to achieve better classification accuracy. Of course, allowing more particles in the network means that each individual particle will make less movements (the total amount of movements between each batch arrival has to be $100,000$). For simplicity, we are ignoring the time needed to re-arrange the network each time a new batch arrives, as it is negligible when the maximum network size is small. But if one decide to use larger networks (thousands of elements), a faster (non-exponential) method to rearrange the network will have to be used in order to keep up with the limitations of processing time.

So, in these simulations, the maximum network size is set to $v_{\max} = \{200; 400; 600; 800; 1,000; 1,200\}$, and the maximum amount of particles is set to $\rho_{\max} = \{20; 40; 60; \ldots; 200\}$. So, there are 400 possible combinations, and each of these configurations is repeated 50 times. The averages are showed in Figure 2.

By analyzing the Figure 2, we notice that the algorithm

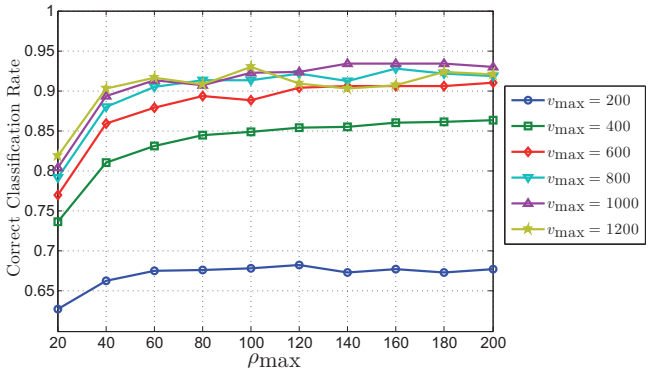

Fig. 2. Simulation 1: Slow Concept Drift. Correct Classification Rate with varying maximum network size ($v_{\max}$) and maximum amount of particles $\rho_{\max}$. $n = 50,000$.

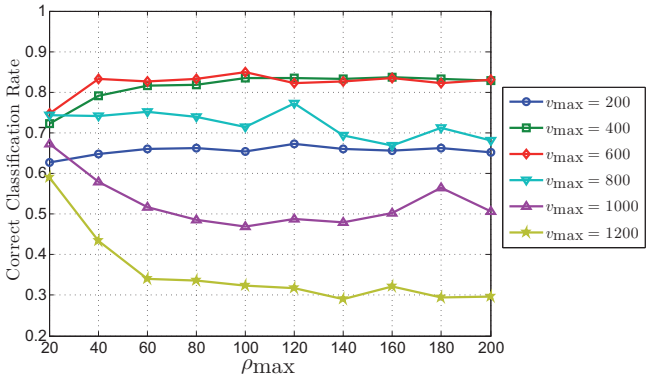

Fig. 3. Simulation 2: Fast Concept Drift. Correct Classification Rate with varying maximum network size ($v_{\max}$) and maximum amount of particles $\rho_{\max}$. $n = 10,000$.

benefits from more particles representing the same class. We can see improvements in the classification performance as the set of particles size $\rho_{\max}$ grows from small to medium, and tends to stabilize at some point beyond which there are no significant improvement.

The maximum network size ($v_{\max}$) is directly related to the amount of past knowledge that the algorithm can "remember". In the scenario above, where the concept drifts slowly, it is useful to have a large network. By analyzing Figure 2, we notice that the best classification performance is achieved when $v_{\max} = 1,000$. If the concept drifts were faster, the optimal size of the network would probably be smaller.

In order to confirm the assumption stated above, we run the same computer simulations again, but with only $10,000$ data items generated (n). In this scenario, there are only 100 batches of elements (instead of 500), but the Gaussian kernels used to generate the data items will still perform a complete clockwise lap. Therefore, the concept drifts five times as fast as in the previous computer simulations, so we expect a significant drop in classification performance when the network is large, leaving the best results to be achieved by smaller networks, which are quicker to "forget" past knowledge that are no longer valid. Again, each of the 400 combinations of $v_{\max}$ and $\rho_{\max}$ is repeated 50 times, and the averages are showed in Figure 3.

By analyzing the Figure 3, we notice that the best classification performance is now achieved when $v_{\max} = 600$. Beyond this optimal point, the classification performance gets worse as the network size grows, which confirms our assumption that, in this particular case, larger networks "remember" too much past knowledge, including those that are no longer valid, and this situation leads to wrong label propagation from the older nodes to the newer nodes.

IV. Conclusion

This paper proposes a new method for semi-supervised classification with concept drifts, i.e., in nonstationary environments. This method is biologically inspired and it uses teams of walking particles competing for network nodes. Each team represent a class problem and the corresponding particles cooperate with their teammates in order to spread their labels by walking in the network, at the same time that they expand their domain by competing with particle from other teams.

The current implementation of the algorithm receives the data items in small batches and it is specially suited for gradual or incremental changes in concept. It is a passive concept drift algorithm, as it naturally adapts to these changes, without any explicit drift detection mechanism. Unlike other methods, it does not rely on base classifiers with explicit retraining process, its built-in mechanisms provide a natural way of learning from new data, gradually "forgetting" older knowledge as older labeled data items became less influent on the classification of newer data items. It is also a single classifier approach, different from most other passive methods which rely on classifier ensembles.

As future work, we intend to build some mechanism to automatically select the parameters which control the sizes of the network and the set of particles, according to the data that is being fed to the algorithm. This could highly improve the performance of the algorithm in scenarios where the concepts may be stable for sometime and/or have different drift speeds through time. Notice that, with these improvements, the proposed algorithm would turn into an active concept drift algorithm.

Acknowledgment

This work was supported by the São Paulo State Research Foundation (FAPESP) and by the Brazilian National Research Council (CNPq).

References

[1] I. Zliobaite, "Learning under concept drift: an overview," *CoRR*, vol. abs/1010.4784, 2010.

[2] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Inf. Fusion*, vol. 9, pp. 56–68, January 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1297420.1297577

[3] G. Ditzler and R. Polikar, "Semi-supervised learning in nonstationary environments," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 31 2011-aug. 5 2011, pp. 2741 –2748.

[4] L. I. Kuncheva, "Classifier ensembles for detecting concept change in streaming data: Overview and perspectives," in *Proc. 2nd Workshop SUEMA 2008 (ECAI 2008)*, Patras, Greece, 2008, pp. 5–10.

[5] A. Bondu and M. Boullé, "A supervised approach for change detection in data streams," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 31 2011-aug. 5 2011, pp. 519 – 526.

[6] J. C. Schlimmer and R. H. Granger, "Incremental learning from noisy data," *Machine Learning*, vol. 1, pp. 317–354, 1986, 10.1007/BF00116895. [Online]. Available: http://dx.doi.org/10.1007/BF00116895

[7] B. Su, Y.-D. Shen, and W. Xu, "Modeling concept drift from the perspective of classifiers," in *Cybernetics and Intelligent Systems, 2008 IEEE Conference on*, sept. 2008, pp. 1055 –1060.

[8] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *In SIAM International Conference on Data Mining*, 2007.

[9] J. P. Patist, "Optimal window change detection," in *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, oct. 2007, pp. 557 –562.

[10] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *In SBIA Brazilian Symposium on Artificial Intelligence*. Springer Verlag, 2004, pp. 286–295.

[11] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *Discovery Science*, ser. Lecture Notes in Computer Science, V. Corruble, M. Takeda, and E. Suzuki, Eds. Springer Berlin / Heidelberg, 2007, vol. 4755, pp. 264–269.

[12] J. Kolter and M. Maloof, "Dynamic weighted majority: a new ensemble method for tracking concept drift," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, nov. 2003, pp. 123 – 130.

[13] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 226–235. [Online]. Available: http://doi.acm.org/10.1145/956750.956778

[14] M. Scholz and R. Klinkenberg, "Boosting classifiers for drifting concepts," *Intelligent Data Analysis*, vol. 11, pp. 3–28, January 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1367489.1367491

[15] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," *Knowledge-Based Systems*, vol. 18, pp. 187–195, August 2005. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2004.10.002

[16] K. Nishida, K. Yamauchi, and T. Omori, "Ace: Adaptive classifiers-ensemble system for concept-drifting environments," in *Multiple Classifier Systems*, ser. Lecture Notes in Computer Science, N. Oza, R. Polikar, J. Kittler, and F. Roli, Eds. Springer Berlin / Heidelberg, 2005, vol. 3541, pp. 509–509.

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

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

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

[20] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," in *Machine Learning*, vol. 39, 2000, pp. 103–134.

[21] A. Fujino, N. Ueda, and K. Saito, "A hybrid generative/discriminative approach to semi-supervised classifier design," in *AAAI-05, Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2005, pp. 764–769.

[22] A. Demiriz, K. P. Bennett, and M. J. Embrechts, "Semi-supervised clustering using genetic algorithms," in *Proceedings of Artificial Neural Networks in Engineering (ANNIE-99*. ASME Press, 1999, pp. 809–814.

[23] R. Dara, S. Kremer, and D. Stacey, "Clustering unlabeled data with soms improves classification of labeled real-world data," in *Proceedings of the World Congress on Computational Intelligence (WCCI)*, 2002, pp. 2237–2242.

[24] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *COLT: Proceedings of the Workshop on Computational Learning Theory*, 1998, pp. 92–100.

[25] T. M. Mitchell, "The role of unlabeled data in supervised learning," in *Proceedings of the Sixth International Colloquium on Cognitive Science*, 1999.

[26] Z.-H. Zhou and M. Li, "Tri-training: exploiting unlabeled data using three classifiers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.

[27] ——, "Semisupervised regression with cotraining-style algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 11, pp. 1479–1493, Nov. 2007.

[28] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley-Interscience, September 2008.

[29] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the Twentieth International Conference on Machine Learning*, 2003, pp. 912–919.

[30] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems*, vol. 16. MIT Press, 2004, pp. 321–328.

[31] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU-CALD-02-107, 2002.

[32] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 55–67, Jan. 2008.

[33] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann, 2001, pp. 19–26.

[34] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semisupervised learning on large graphs," in *Conference on Learning Theory*. Springer, 2004, pp. 624–638.

[35] M. Belkin, N. P., and V. Sindhwani, "On manifold regularization," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*. New Jersey: Society for Artificial Intelligence and Statistics, 2005, pp. 17–24.

[36] T. Joachims, "Transductive learning via spectral graph partitioning," in *Proceedings of International Conference on Machine Learning*. AAAI Press, 2003, pp. 290–297.

[37] F. A. Breve, L. Zhao, M. G. Quiles, W. Pedrycz, and J. Liu, "Particle competition and cooperation in networks for semi-supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, 2011, prePrint - DOI: 10.1109/TKDE.2011.119.

[38] R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D. Tax, and S. Verzakov, "Prtools4.1, a matlab toolbox for pattern recognition," Delft University of Technology, 2007. [Online]. Available: http://prtools.org