Semi-Supervised Learning with Concept Drift using Particle Dynamics applied to Network Intrusion Detection Data

Fabricio Breve
Institute of Geosciences and Exact Sciences (IGCE)
São Paulo State University (UNESP)
Rio Claro, Brazil
Email: fabricio@rc.unesp.br

Liang Zhao
Institute of Mathematics and Computer Science (ICMC)
University of São Paulo (USP)
São Carlos, Brazil
Email: zhao@icmc.usp.br

Abstract—**Concept drift, which refers to non stationary learning problems over time, has increasing importance in machine learning and data mining. Many concept drift applications require fast response, which means an algorithm must always be (re)trained with the latest available data. But the process of data labeling is usually expensive and/or time consuming when compared to acquisition of unlabeled data, thus usually 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 based on assumptions that the data is static. Therefore, semi-supervised learning with concept drifts is still an open challenging task in machine learning. Recently, a particle competition and cooperation approach has been developed to realize graph-based semi-supervised learning from static data. We have extend that approach to handle data streams and concept drift. The result is a passive algorithm which uses a single classifier approach, naturally adapted to concept changes without any explicit drift detection mechanism. It has built-in mechanisms that provide a natural way of learning from new data, gradually "forgetting" older knowledge as older data items are no longer useful for the classification of newer data items. The proposed algorithm is applied to the KDD Cup 1999 Data of network intrusion, showing its effectiveness.**

I. Introduction

Many machine learning algorithms are designed based on the assumption that the databases are static and data samples are independent. However, in practical situations, these assumptions usually do not hold. Some examples include climate prediction, fraud detection, network intrusion, energy demand, and many other real-world applications, in which concepts and data distributions may not be stable over time. Applying traditional methods to such problems would inevitably result in low performance. In order to handle these problems, the technique has to incrementally learn from streams of data fed to it either online or in small batches. These methods have to address two conflicting objectives: retaining previously learned knowledge that is still relevant and replacing any obsolete knowledge with updated information [1]–[5].

The term "concept drift" was coined by Schlimmer and Granger [6]. They formulated the incremental learning from noisy data and presented an adaptive learning algorithm STAGGER, one of the earliest attempts to solve this kind of problems. When all data items are sampled from the same 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. The main assumption on concept drift is its uncertainty in the future [1]. For instance, it is expected that sales of ice-cream are higher during summer, but the specific peak days are mostly unpredictable because it depends on temperature and other factors.

There are some attempts to categorize different kinds of drifts. Minku et al. [7] divided concept drifts in 14 categories based on criteria like severity, speed, predictability, frequency, and recurrence. Zliobaite [1] presents four different main structural drift types: *sudden drift*, in which the concept changes abruptly; *gradual drift*, in which there is a period of time where samples from both the old and new concept are mixed together, with the probabilities of a sample being from the new concept increasing through time; *incremental drift*, in which the concept changes slowly (stepwise), thus it is noticed only when looking at a longer time period; and *reoccurring contexts*, in which previously active concept reappears after some time.

Concept drift algorithms are usually classified in two major groups: active and passive algorithms. The active algorithms, which are also called trigger based algorithms, have some kind of detector to indicate the need of model change [8]–[12]. On the other hand, passive algorithms, also called evolving algorithms, do not detect changes. In this case, the drift is simply assumed, i.e., the learner evolves independently on triggers or detectors. Most techniques in this category are classifier ensembles [2], [13]–[17], in which adaptivity is achieved by assigning a weight for each model's output at each instant.

In many applications, it is not possible to collect and store all data that becomes available for later analysis. Moreover, some applications demands fast response, such as fraud detection in credit cards and natural disaster forecast. In order to properly detect and learn the concept drifts in these conditions, the incremental learning algorithms require that the data stream is constantly fed to it, either online or in batches. Typically, to achieve good performance regardless of shifting in concepts, the 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 acquisition of unlabeled data. In such scenarios, only a small fraction of the incoming data may be effectively labeled. Semi-supervised learning may be useful in these cases, as the algorithms in this category are specifically designed to learn from data sets where there are lots of unlabeled data and only a few labeled data items [18]–[20].

Semi-supervised learning methods may be divided in some categories, like generative models [21], [22], cluster-and-label techniques [23], [24], co-training and tri-training techniques [25]–[28], low-density separation models [29], and graph-based methods, which is the most active category in the recent years. Graph-based methods include Mincut [30], Local and Global Consistency [31], label propagation techniques [32], [33], among others. Though many semi-supervised graph-based methods were developed, most of them are similar and they may be seen as regularization frameworks [18], differing only in the particular choice of the loss function and the regularizer [30], [31], [34]–[37]. Moreover, most of these methods are also based on the assumption that the data is static. Therefore, building a semi-supervised learning approach to deal with concept drift is still an open challenge in machine learning.

Recently, a semi-supervised learning method based on particle competition and cooperation in networks was developed [38]. In this method, particles walk in the network trying to possess the nodes, i.e., marking their territory. Particles representing the same 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 class labels belong to different teams and compete against each other. Like many other semi-supervised learning algorithms, the particle competition and cooperation technique was developed to process static data. We have extended this approach to handle data streams and concept drift. These objectives are achieved by introducing rules: to generate new nodes, to eliminate older nodes, and to rearrange the connections in the network, as new data examples become available in small batches. In the extended model, new particles are also generated for new labeled data items, while older particles are eliminated after they fulfill their purpose of spreading their classes labels, among other improvements. In a preliminary work, the particles method with concept drift was applied to artificially generated data sets as proof of concept [39]. In this paper, the approach was further extended and applied to the KDD Cup 1999 Data of network intrusion.

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 suitable to incremental and gradual changes in concept [1]. It falls in the category of passive algorithms, as it naturally adapts itself to concept changes without any explicit drift detection mechanism. Different from most of other methods, 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 become 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, some computer simulation results are presented, showing the effectiveness of the proposed method. Finally, in Section 4 we draw some conclusions.

II. Model Description

In this section we describe the proposed algorithm. It receives the data items in small batches. These data items are then transformed into nodes of an undirected and unweighed network. 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 on that node. As the system executes, 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, they decrease the domination levels of other teams. Each team of particles will act cooperatively trying to dominate as many nodes as possible, while preventing intrusion of other teams in their territory. Finally, each unlabeled node will be labeled according to the team that have dominated it.

Given a data set, firstly, the data items are converted into network nodes and each node is 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 always has 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 $\mathbf{W}$:

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

where 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} to limit its growing. It is important that the network does not grow indefinitely, because it will slower the algorithm and the older nodes will affect the classification of newer nodes, which is not desirable in the case of classifying non stationary 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 generated to take their place. The labeling process will be explained later.

For each labeled node v_i (i.e., $y_i \neq 0$) arriving to 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 data class. 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 to make room to the newly generated particles. Each particle ρ_j has a variable $\rho_j^\omega(t) \in [0,1]$, called *particle strength* indicating how much the particle can affect the node that it is visiting at time t.

Each node v_i has a vector variable $\mathbf{v_i^\omega(t)} = \{v_i^{\omega_1}(t), v_i^{\omega_2}(t), \dots, v_i^{\omega_c}(t)\}$ 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, decrease the other teams' domination levels in that same node.

Each node v_i has the initial values 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 its initial strength is set to $\rho_j^\omega(t) = 1$, i.e., each particle starts with the maximum strength.

At each iteration, a particle chooses a neighbor node to target by using the following probabilities:

$$p(v_i|\rho_j) = (1-\alpha)\frac{W_{qi}}{\sum_{\mu=1}^{n} W_{q\mu}} + \alpha\frac{W_{qi}v_i^{\omega_\ell}}{\sum_{\mu=1}^{n} W_{q\mu}v_i^{\omega_\ell}}, \tag{4}$$

where the first term means that the particle randomly chooses a neighbor node to target, and the second term means that each particle ρ_j chooses a target node v_i with probabilities defined according to its team domination level on that neighbor $\rho_j^{\omega_\ell}$. In this equation, 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 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 $\alpha \approx 0.5$.

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 it decreases the domination level of the other teams in that same node. The exceptions 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:

$$v_i^{\omega_\ell}(t+1) = \begin{cases} \max\{0, v_i^{\omega_\ell}(t) - \frac{\Delta_v \rho_j^\omega(t)}{c-1}\} \\ \quad \text{if } y_i = 0 \text{ and } \ell \neq \rho_j^f \\ v_i^{\omega_\ell}(t) + \sum_{q\neq\ell} v_i^{\omega_q}(t) - v_i^{\omega_q}(t+1) \\ \quad \text{if } y_i = 0 \text{ and } \ell = \rho_j^f \\ v_i^{\omega_\ell}(t) \quad \text{if } y_i \neq 0 \end{cases}, \tag{5}$$

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. (5).

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's strength $\rho_j^\omega(t)$ is updated as follows:

$$\rho_j^\omega(t+1) = v_i^{\omega_\ell}(t+1), \tag{6}$$

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 $v_i^{\omega_j}$ at the node v_i.

Notice that particles also have to move when the node they are staying is going to be eliminated. In this case, the particle moves randomly to any of its neighbors, and its strength is set according to Eq. (6). But node domination levels do not change, i.e., Eq. (5) does not apply in this case.

When a particle tries to move to a 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 at 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 [38] 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 happens, until they are removed to give place to new particles.

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

$$y_i = \arg\max_\ell v_i^{\omega_\ell}(t). \tag{7}$$

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 that the teams of particles can fight for it. When it is the 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 is classified 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.

It is important to notice that even though the particles are generated from labeled nodes, they may stay in the network long after their corresponding nodes are removed. The opposite also applies, a particle may be removed even if its corresponding labeled node is still on the network. The specific behavior is highly dependent on the amount of labeled data items in the last batches and on v_{max} and ρ_{max} parameters, which are set according to the problem being addressed.

III. Computer Simulations

In this section, we present simulation results to show the effectiveness of the proposed method in semi-supervised classification with concept changes. The following parameters are used in these simulations: $\Delta_v = 0.1$, $\alpha = 0.5$, and $k = 5$. These values are obtained by empirical optimization using the grid method.

The first set of experiments is performed using the KDD Cup 1999 Data [40], a data set used in a competition to build a network intrusion detector model, which has been widely tested in the concept drift domain in recent years [41]–[47]. This database contains a set of $494,021$ data items to be audited, including a wide variety of intrusions simulated in a military network environment. The data set contains 42 attributes, 34 continuous and 8 categorical. The categorical attributes are converted into numerical attributes (a new attribute to each category). The data items are presented to the algorithm in batches of 100 elements. 10% of these data items are labeled and the remaining are presented unlabeled (a typical semi-supervised learning scenario). Each batch of elements is converted to 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 exceptions are the last batches, which are labeled but are not removed from the network, as there are no new data items to replace them. The labeled data items are divided into "good packets" or "bad packets".

In this kind of problems, we have to work under some constraints, the amount of processing is limited by the interval between each batch arrival and by the hardware executing the algorithm. Therefore, we fix $100,000$ 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 standing has [38]. We control the average node degree of the network by defining k. Since we fixed $k = 5$, 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 (ρ_{max}), we can tune these parameters 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 network size limit is small and the interval between batches arrival is large enough. In these simulations, the maximum network size is set to $v_{\text{max}} = \{200; 400; 600; 800; 1,000; 1,200\}$, and the maximum amount of particles is set to $\rho_{\text{max}} = \{20; 40; 60; \ldots; 200\}$. So, there are 400 possible combinations, and each of these configurations is repeated 4 times. The averages of correct classification rates are showed in Figure 1a.

By analyzing the Figure 1a, we see that the best classification performance is achieved with a small amount of particles ($\rho_{\text{max}} = 20$), and a small network ($v_{\text{max}} = 200$), which is a clue that the data is very volatile and older knowledge has to be discarded sooner. Interestingly, as the maximum network size (v_{max}) also increases, the optimal amount of particles (ρ_{max}) also increases. This indicates that more particles are needed to achieve optimal performance as the network becomes larger. The algorithm achieved over 99% correct classification rate in the best configuration, which is a pretty impressive result.

The maximum network size (v_{max}) is directly related to the amount of past knowledge that the algorithm can "remember". In the above scenario, it seems that packages from a given kind of attack usually comes in batches and there are enough labeled data in these batches, so the algorithm only needs a short memory of the last packages.

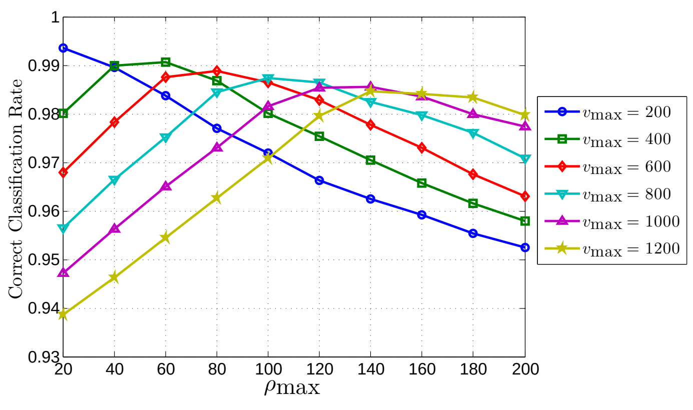
The second set of experiments is similar to the first one, except that the proportion of labeled samples is lowered to only 1% of the data set. The results are presented in Figure 1b. In this case, the best result is achieved with a larger network ($v_{\text{max}} = 1,200$) and a small amount of particles ($\rho_{\text{max}} = 20$). For all network sizes, the minimum amount of particles tested worked better. In this case, the best configuration lead to a correct classification rate of almost 98%.

In this last scenario, as labeled data is scarcer, it is better to keep older nodes longer to achieve better performance. However, particles representing older labeled data actually lower the algorithm performance. This indicates that having more particles to cover a larger network is good only when they are representing similar data. Particles representing types of data that are no longer present in the current batches may lead to worse performance.

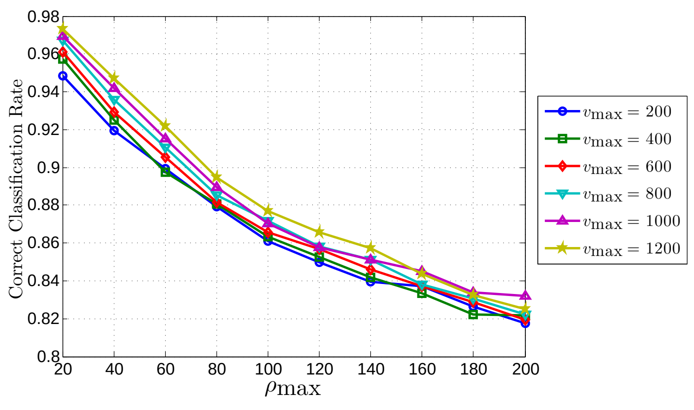
Notice that the parameters Δ, α, and k could also be fine-tuned as their optimal value may be slightly different depending on v_{max} and ρ_{max}, but they were kept fixed in order to keep the experiments clearer. Therefore, further performance improvement may be expected if all parameters are optimized together.

IV. Conclusion

This paper proposes a new method for semi-supervised classification with concept drift. It is inspired in the competitive and cooperative behavior of some animals and the way they mark and protect their territory. Each team of particles represents the labeled nodes of the same class. The particles in the same team cooperate with their teammates in order to spread their labels by walking and marking territory in the network. At the same time, each team try to expand its domain by competing against other teams.



(a)



(b)

Fig. 1. KDD Cup 1999 Data. Correct classification rate against maximum network size ($v_{\max}$) and maximum amount of particles $\rho_{\max}$. $n = 494,021$. (a) 10% labeled data items (b) 1% labeled data items

The algorithm receives small batches of data items from data streams and it is specially designed to handle gradual or incremental changes in concepts. It is a passive concept drift algorithm because it naturally adapts to concept changes without any explicit drift detection mechanism. Unlike other methods, it does not rely on base classifiers with explicit retraining process, it has built-in mechanisms to provide a natural way of learning from new data, gradually "forgetting" older knowledge as older labeled data items become less influent on the classification of newer data items. Different from most other passive methods, which rely on classifier ensembles, the proposed algorithm is a single classifier approach.

The method can also be easily applied to online data streams. In fact, if we use batches containing only a single item, it already does that. However, the overhead of network reconfiguration would be too high. So, our next step is to generate less computational intensive ways of reconfiguring the network (recalculating the k-nearest neighbors of each node).

As a future work, we also intend to build mechanisms to dynamically set the network sizes and amount of particles according to the data being fed to the algorithm. This mechanism can highly improve the performance of the algorithm in non stationary environments where the concepts may increase/decrease their evolving rhythm through time. In this sense, the algorithm would behave like an active algorithm, not by detecting individual drifts, but rather by detecting changes in drifting frequency and intensity.

Acknowledgment

This work was supported by the São Paulo State Research Foundation (FAPESP), the Brazilian National Research Council (CNPq), and the Foundation for the Development of Unesp (Fundunesp).

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] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 730–742, 2010.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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

[15] 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

[16] 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

[17] 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.

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

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

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

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

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

[30] 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.

[31] 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. [Online]. Available: http://www.kyb.tuebingen.mpg.de/bs/people/weston/localglobal.pdf

[32] 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. [Online]. Available: http://citeseer.ist.psu.edu/581346.html

[33] 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.

[34] 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.

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

[36] 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.

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

[38] 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.

[39] F. Breve and L. Zhao, "Particle competition and cooperation in networks for semi-supervised learning with concept drift," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 2012, pp. 1–6.

[40] S. Hettich and S. D. Bay, "The UCI KDD archive," 1999. [Online]. Available: http://kdd.ics.uci.edu

[41] S. Kaur, V. Bhatnagar, S. Mehta, and S. Kapoor, "Categorizing concepts for detecting drifts in stream," in *Proceedings of 15th International Conference on Management of Data (COMAD 2009)*, 2009.

[42] J. Gao, W. Fan, and J. Han, "On appropriate assumptions to mine data streams: Analysis and practice," in *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ser. ICDM '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 143–152. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2007.96

[43] X. Zhang and W. Wang, "Self-adaptive change detection in streaming data with non-stationary distribution," in *Advanced Data Mining and Applications*, ser. Lecture Notes in Computer Science, L. Cao, Y. Feng, and J. Zhong, Eds. Springer Berlin / Heidelberg, 2010, vol. 6440, pp. 334–345, 10.1007/978-3-642-17316-5_33. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17316-5_33

[44] E. J. Spinosa, A. P. de Leon F. de Carvalho, and J. a. Gama, "Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks," in *Proceedings of the 2008 ACM symposium on Applied computing*, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 976–980. [Online]. Available: http://doi.acm.org/10.1145/1363686.1363912

[45] G. Folino, C. Pizzuti, and G. Spezzano, "An adaptive distributed ensemble approach to mine concept-drifting data streams," in *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, vol. 2, oct. 2007, pp. 183 –188.

[46] Z. Ouyang, M. Zhou, T. Wang, and Q. Wu, "Mining concept-drifting and noisy data streams using ensemble classifiers," in *Artificial Intelligence and Computational Intelligence, 2009. AICI '09. International Conference on*, vol. 4, nov. 2009, pp. 360 –364.

[47] Y. Liao, V. R. Vemuri, and A. Pasos, "Adaptive anomaly detection with evolving connectionist systems," *Journal of Network and Computer Applications*, vol. 30, no. 1, pp. 60 – 80, 2007, network and Information Security: A Computational Intelligence Approach. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S108480450500041X