SFNClassifier: a scale-free social network method to handle concept drift

Conference Paper · March 2014

CITATIONS
20

READS
179

3 authors:

Jean Paul Barddal
Pontifícia Universidade Católica do Paraná (PUC-PR)
57 PUBLICATIONS **880** CITATIONS
SEE PROFILE

Heitor Murilo Gomes
The University of Waikato
58 PUBLICATIONS **968** CITATIONS
SEE PROFILE

Fabrício Enembreck
Pontifícia Universidade Católica do Paraná (PUC-PR)
143 PUBLICATIONS **1,418** CITATIONS
SEE PROFILE

SFNClassifier: A Scale-free Social Network Method to Handle Concept Drift

Jean Paul Barddal
Escola Politécnica - Pontifícia Universidade Católica do Paraná
Rua Imaculada Conceição, 1155
80215-901
Curitiba, Brazil
jean.barddal@pucpr.br

Heitor Murilo Gomes
Programa de Pós-Graduação em Informática - Pontifícia Universidade Católica do Paraná
Rua Imaculada Conceição, 1155
80215-901
Curitiba, Brazil
hmgomes@ppgia.pucpr.br

Fabrício Enembreck
Programa de Pós-Graduação em Informática - Pontifícia Universidade Católica do Paraná
Rua Imaculada Conceição, 1155
80215-901
Curitiba, Brazil
fabricio@ppgia.pucpr.br

ABSTRACT

In this paper, we present a new ensemble method, the Scale-free Network Classifier (SFNClassifier), that is conceived as a dynamic sized scale-free network. In Data Stream Mining, ensemble-based approaches have been proposed to enhance accuracy and allow fast recovery from concept drift. However, these approaches are based on both update and polling heuristics that do not present good accuracy results in arbitrary domains and do not represent explicitly the similarity between classifiers. The representation of the ensemble as a network allows us to extract centrality metrics, which are used to perform a weighted majority vote, where the weight of a classifier is proportional to its centrality value. Based on empirical studies, we concluded that SFNClassifier has comparable results to other ensemble-learners in terms of accuracy and outperformed the other methods in processing time.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning, Induction*

General Terms

Algorithms

Keywords

Data Stream Mining, Concept Drift, Ensemble Classifiers, Classification, Social Network Analysis

1. INTRODUCTION

Let S denote a data stream which makes one instance $i = (\vec{x}, y)$ available after t time units, where $\vec{x}$ defines an attribute vector and y the class value. Let A denote a concept that maps an attribute vector $\vec{x}$ to a class value y. A concept drift occurs when an underlying concept A is replaced by a new concept B, abruptly or gradually. A drift is denominated abrupt if after only one instance the new concept becomes stable. Conversely, a drift is gradual if there is a drift window, where instances can belong to both concept A or to concept B. Intuitively, at the beginning of a drift window there is a higher probability that instances belongs to concept A. As we move towards its end, the probability that it belongs to concept B raises. The window end occurs when concept B becomes stable. To model the probability that every new instance i drawn from S belongs to concept A or B we use the concept drift framework stated in [6] where the sigmoid function (1) is used to model these probabilities.

$$f(t) = 1/(1 + e^{-s(t-t_0)}) \quad (1)$$

In [6] authors observe that (1) has a derivative at time t_0 equal to $f'(t_0) = s/4$ and that $tan\ \alpha = f'(t_0)$, thus $tan\ \alpha = s/4$. Also, $tan\ \alpha = 1/W$ and as $s = 4\ tan\ \alpha$ then $\alpha = 4/W$, namely t_0 (time of drift), W (length of drift window) and α (phase angle). In this sigmoid model, there are only two parameters to be specified: t_0 and W.

Aiming the concept drift problem, ensemble-based methods were proven a good approach to handle concept drift. In these methods, a set of classifiers is trained and classifier's predictions are combined in order to obtain a global prediction. The maintenance of the ensemble (addition and removal) depends on each algorithm. Yet, the majority of the algorithms declared in the state-of-the-art keep a static ensemble size. Thus, too few or too many classifiers can be generated, affecting memory, processing time and accuracy. In our approach the ensemble is a network of classifiers that evolves naturally according to the scale-free model [2]. The network allows the usage of prominence metrics, which determine the importance of each actor in the network and are used to poll classifiers' votes.

The remainder of this work is organized as follows. In Section 2, we present state-of-the-art algorithms for data stream classification. Section 3 introduces the major aspects

of social networks. Section 4 focuses on the SFNClassifier algorithm. Section 5 presents the experimental evaluation and a discussion over the results obtained. Finally, Section 6 presents our conclusions and asserts future work.

2. RELATED WORK

Most of the existing work on ensemble classifiers relies on developing algorithms to improve overall classification accuracy that copes with concept drift explicitly [6] or implicitly [16, 15, 25]. As discussed in [17], an ensemble classifier can surpass an individual classifier's accuracy if its component classifiers are diverse and achieve a classification error below 50%. An ensemble is said diverse if its members misclassify different instances. Another important trait of an ensemble refers to how it combines individual decisions. If the combination strategy fails to highlight correct and obfuscate incorrect decisions then the method is jeopardized. In addition, when dealing with data streams it is important to consider concept drifts and include some kind of adaptation strategy into the ensemble.

2.1 DWM

In [16] authors presented the Dynamic Weighted Majority (DWM) algorithm. DWM maintains a set of classifiers that is dynamic, i.e. it can grow and shrink. Every classifier has a weight, which increases every time it predicts correctly and decreases otherwise according to a parameter β. The ensemble prediction is given by a weighted majority vote. Classifiers are removed from the ensemble if their weights are below a user-defined threshold α. New classifiers are added if the ensemble prediction is incorrect. Since adding classifiers after every incorrect prediction can yield too many additions and removals for noisy data streams, authors introduced a parameter, namely the period size, to determine after how many instances an ensemble update will take place.

2.2 Online Bagging

The Online Bagging algorithm was introduced in [21] as an adaptation of the batch ensemble classifier Bagging [21]. Originally, a bagging ensemble is composed of k classifiers, which are trained with subsets (bootstraps) D_j of the whole training set D. These subsets are formed by sampling with replacement from training set D. However, sampling usually is not feasible in a data stream configuration, since that would require storing all instances before creating subsets. Authors in [21] observe that the probability of each instance to be selected for a given subset is approximated by a Poisson distribution with $\lambda = 1$.

2.3 ADWIN Bagging

In [6] authors presented the ADWIN Bagging ensemble-base method to handle concept drift. This method uses the ADWIN algorithm to detect changes in the data stream. ADWIN keeps a variable-length window of recently seen items, with the property that the window has the maximal length statistically consistent with the hypothesis "there has been no change in the average value inside the window". The ADWIN change detector is parameter- and assumption-free in sense that it automatically detects and adapts to the current rate of change. The ADWIN Bagging is the Online Bagging method presented in [21] with the addition of the ADWIN algorithm as a drift detector. When a drift is detected, the worst classifier of the ensemble is removed and a new classifier is added and trained with instances chosen by the Online Bagging method.

2.4 Leveraging Bagging

As an improvement to ADWIN Bagging algorithm, authors in [5] developed the Leveraging Bagging. Leveraging Bagging uses the ADWIN algorithm to detect drifts, but enhances the training method by proposing two improvements. First, the Bagging performance is leveraged by increasing the resampling process, which is done by using the Online Bagging method where the weights of the resamples are increased by largening the variable λ. Second, randomization is added at the output of the ensemble using output codes. For each possible class value, a binary string of length n is assigned and then an ensemble of n classifiers is created. Each classifier learns one bit for each position in this binary string. When a new instance arrives, it is assigned x to the class whose binary code is closest. Each classifier in the ensemble uses a random code, which creates different functions, increasing the diversity of the ensemble.

3. SOCIAL NETWORKS

Social Network Theory has been applied in many research fields, mainly due to its precise and formal description of structural variables. Despite of the concern of social network analysis to subjective topics, such as individual behavior in society, its building block, the social network, can be precisely represented as a graph (or digraph). A simple social network is characterized by a set of actors, one or more relations and a set of connections between pairs of actors for each relation. The relation defines which connections exists between pairs of actors. Actors and their actions are seemed as interdependent, i.e. they are not interpreted as autonomous independent units.

There are three major network models presented in the bibliography. The first is denominated random [11], which construction is based in the hypothesis that the existence of a connection between a pair of nodes is given by a probability p. In [24] authors demonstrated the small-world model based in the studies of "small world" conducted in [19]. This model ties in attributes of both random and regular networks. Thus, this topology presents a high clustering coefficient (as regular networks) and a small average path length (as random networks). Finally, the scale-free [2] topology aims on modelling the networks of real-world situations with higher accuracy than the random and small-world networks. Thus, authors modeleded the construction (assembly) and evolution (growth) of the network as follows.

Growth: starting with a determined network size (n), for every t time unit, a new actor is added to the network establishing connections with different actors.

Preferential attachment: When choosing the actors which a new actor will connect to, it is assumed a probability $\prod$ that states the chances that this actor receives a new connection. The value of $\prod$ depends on the degree centrality of the actor and is calculated by:

$$\prod(i) = \frac{d_i}{\sum_j^N d_j}$$

where d_i stands for the degree metric for the i^{th} actor and $\sum_j^N d_j$ is the sum of the degree metrics for every actor in the network N.

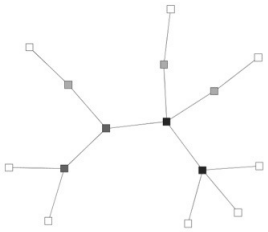

Figure 1: Example of a scale-free network where darker colored nodes are prominently more relevant.

Due to the preferential attachment process, scale-free networks are "dominated" by a few vertices denominated hubs [7]. Thus, scale-free network degree distribution follows the distribution: $p(k) \sim k^{(-\lambda)}$, where $p(k)$ is the probability of a random node being attached to k other nodes and $\{\lambda \in \Re \mid 2 \leq \lambda \leq 3\}$. Figure 1 shows an example of a scale-free network and its hubs highlighted in darker colors.

In order to determine the importance of actors in social networks centrality metrics were developed. These metrics determine numerically the importance of actors in a social network and which are more prominent than others. Centrality metrics such as degree, betweenness, closeness, eigenvector and pagerank are further discussed in [20].

4. SFNCLASSIFIER

Scale-free Network Classifier (SFNClassifier) is an ensemble-based method that copes with concept drift and uses social networks aspects and metrics to establish predictions based on the scale-free network construction model. The hypothesis is that prominent classifiers are more likely to establish new connections hence improving its centrality. As in DWM [16] and SAE [13], our proposal is based in a period size parameter p that determines how many instances will be evaluated before a network update takes place.

Initially, the algorithm instantiates a single classifier e for learning the first period. During a period, predictions are calculated using each classifier prediction weighted by its centrality metric, e.g. degree, closeness, betweenness or eigenvector. This polling process is based in the prominence of classifiers. In case the global prediction is correct, the correct predictions counter ϕ is incremented; otherwise, this instance is added to set of misclassified instances V. The set of misclassified instances V is used to train a new classifier at the end of the period, improving the diversity of the ensemble since the new classifier has observed only these instances. After each prediction, every classifier on the network is trained with the same instance i. Since all classifiers are trained with the same instance i, it is possible that the concepts learned in different classifiers converge. Nevertheless, since the ensemble is dynamic, and classifiers are added and removed in different period, their concepts usually do not converge unless the period size p is rather big.

After a period, the network evolves and the evolution process is divided into: classifier removal, rewiring process and classifier addition. The classifier removal process determines whether and which classifier in the network should be removed based on the expected accuracy rate (θ) parameter. If the global accuracy rate of the network (Φ) is under the threshold θ and k is bigger than the minimum network size m_{size} (typically a small number), then the most inaccurate classifier in the network is removed. When a classifier is removed, it is possible that the network becomes a disconnected graph, thus affecting the centrality metrics of all other classifiers. Therefore, a rewiring process is executed, and for each neighbor that was attached to that removed classifier, the preferential attachment process is used to reorganize the network and keep it entirely connected.

When both the removal and rewiring processes are completed, the routine *trainExpert* instantiates a new classifier using the set of misclassified instances V. At conventional scale-free networks, the preferential attachment probability of a node receiving a new connection is proportional to its degree metric, i.e. the greater amount of connections a node has, the bigger probability it has to establish a new connection. The new classifier is added to the network N using an adaptation of the preferential attachment law:

$$\prod(i) = \frac{\Phi_i}{\sum_j^N \Phi_j}$$

where Φ_i stands for the accuracy of the i^{th} actor and $\sum_j^N \Phi_j$ is the sum of every actors' accuracy in the last period of evaluation.

Thus, classifiers that performed well in the last period tends to establish connections with new classifiers, raising their centrality metrics. Hence, higher accuracy classifiers

Algorithm 1 SFNClassifier pseudo-code. **Input:** stream of instances S that provides an instance i every t moments, the period size p, a minimum network size m_{size} and a expected accuracy threshold θ. **Local variables:** a network $N = \{n_1, n_2, ..., n_k\}$, where k is the variable ensemble size, a single classifier e, an instance $i = (\vec{x}, y)$, a network correct predictions counter ϕ, a set of misclassified instances observed during the current period V and Φ the ensemble accuracy in the current period.

```
N ← {e}
loop
  for j = 1 to p do
    i ← next_instance(S)
    prediction ← getPredictions(N, i)
    if prediction = y then
      φ ← φ + 1
    else
      V ← V ∪ i
    end if
    for all n in N do
      Train(n, i)
    end for
  end for
  if Φ < θ then
    if k > m_size then
      removeExperts(N)
    end if
    newExpert ← trainExpert(V)
    N ← N ∪ {newExpert}
    calculateMetric(N)
  end if
  V ← ∅
  φ ← 0
end loop
```

become more prominent and their votes have bigger importance during the polling process. SFNClassifier is able to calculate degree, betweenness, closeness, eigenvector and pagerank centralities. Each centrality metric directly affects the accuracy results since each one distributes weights for nodes in different ways. Centrality metrics such as betweenness ranks non-central vertices with zero. As new classifiers are added to the network at non-central positions, the usage of betweenness centrality metric would not use these new classifier's predictions. Other metrics such as degree, closeness and pagerank also weights non-central nodes yet eigenvector acquainted the best average accuracy results for evolving streams.

Finally, once both classifiers removals and additions are done, it is possible to calculate the centrality metric for the next period and both statistics, correct classifications counter (ϕ) and V are cleared. The pseudo code for SFNClassifier is presented in Algorithm 1.

Another important feature of SFNClassifier is that it is not bound to a particular base learner, although it may output better results when an unstable learner is used (such as a decision tree) since these are able to yield different models given small differences on the input data (enhances ensemble diversity).

5. EMPIRICAL RESULTS

To compare SFNClassifier with other methods, we developed a validation environment with three synthetic data generators in different configurations and a real database. This validation environment centers on accuracy and processing time. Firstly, the data streams generators and the real data streams are briefly introduced.

5.1 RTG

The Random Tree Generator (RTG) generator [9] builds a decision tree randomly by choosing the split nodes and assigning class labels for the leaves. Having this model set, instances are created with random attribute values, and the class value is defined by visiting each node of the tree until a leaf is reached.

5.2 SEA

This generator was presented in [23] with the objective of testing the Streaming Ensemble Algorithm (SEA). SEA generates instances with three attributes, which values range from 0 to 10. These instances can be interpreted as a point in a three dimensional space. It occurs that only the first two attributes are relevant to determine the class label. Concept drifts are simulated on SEA by changing the threshold (θ) that determines whether the class value is 0 or 1. In Equation (2) we present the threshold comparison to the sum of the first two attributes in the dataset: f_1 and f_2, which determines the class of a given instance.

$$class = \begin{cases} 1, & f_1 + f_2 \leq \theta \\ 0, & otherwise \end{cases} \quad (2)$$

5.3 AGRAWAL

This generator creates instances with 6 attributes where half are nominal and half continuous. Attributes' values are randomized and 10 different concept functions are stated in [1] aiming on classifying these instances.

Experiment identifier	Stream configuration			
	Data generator	# of drifts	Length of drift window (W)	Time of drift (t_0)
RTG-10	RTG (10 attributes)	0	N/A	N/A
RTG-100	RTG (100 attributes)	0	N/A	N/A
AGRAWAL-1	AGRAWAL	1	1	500,000
AGRAWAL-2	AGRAWAL	1	1,000	500,000
AGRAWAL-3	AGRAWAL	3	1 for each drift	250,000 500,000 750,000
AGRAWAL-4	AGRAWAL	3	1,000 for each drift	250,000 500,000 750,000
SEA-1	SEA	1	1	500,000
SEA-2	SEA	1	1,000	500,000
SEA-3	SEA	3	1 for each drift	250,000 500,000 750,000
SEA-4	SEA	3	1,000 for each drift	250,000 500,000 750,000
SPAM CORPUS	None (real dataset)	1	750	1,500

Table 1: Experiments configurations

5.4 Spam Corpus

The Spam Corpus database was developed in [14] as result of a text mining process on an online news dissemination system. Part of this work intended on creating an incremental filtering of emails classifying them as spam or not, and based on this classification, deciding whether this email was relevant for dissemination among users. This dataset has 9,324 instances and 39,917 boolean attributes, such that each attribute represents the presence of a single word (the attribute label) in the instance (e-mail).

5.5 Experiments

Table 1 presents the experiments configurations for every stream generator and the Spam Corpus dataset. RTG-10 and RTG-100 both stand for Random Tree Generator where they contain 10 and 100 attributes, respectively. In both, half of the attributes are nominal and half continuous. Every synthetic stream configuration in our experiments contains 1 million instances. We adopted the Prequential procedure [12] due the the monitoring of the evolution of performance of models over time although it may be pessimistic in comparison to the holdout estimative. Nevertheless, authors in [12] observes that the prequential error converges to an periodic holdout estimative [4] when estimated over a sliding window. Along these lines, we determined a sliding window of 100 thousand instances for synthetic streams and 1 thousand for the Spam Corpus experiment. We positioned the concept drifts in synthetic experiments such that instances are equally distributed for each concept. Concept drifts positions (t_0) and lengths (W) are presented in Table 1.

In order to compare the algorithms, we give results for a single Hoeffding Tree [9], DWM, SFNClassifier, Online, ADWIN and Leveraging Bagging algorithms. The presence of the Hoeffding Tree is reasonable since there are streams where the base learner outperforms ensemble methods. The DWM parameters are: base learner adopting the Hoeffding Tree, update period $p = 1,000$, $\beta = 0.5$, and delete threshold $\alpha = 0.01$. SFNClassifier parameters are: base learner as the Hoeffding Tree, update period $p = 1,000$, expected accuracy rate $\theta = 0.95$, minimum network size $m_{size} = 3$ and the eigenvector centrality metric. The adoption of the eigenvector centrality is due to its relative weighting to new

Stream configuration	Average accuracy (%)					
	Hoeffding Tree	DWM	SFNClassifier	Online Bagging	ADWIN Bagging	Leveraging Bagging
RTG-10	90.06±2.62	76.44±9.57	96.25±1.51	96.66±2.58	96.66±2.58	80.02±11.27
RTG-100	95.93±0.97	92.98±3.82	96.48±0.86	94.13±3.95	94.48±3.40	57.03±5.82
AGRAWAL-1	91.12±4.08	75.39±15.22	88.79±7.03	91.23±3.90	93.99±1.25	93.68±1.01
AGRAWAL-2	90.26±4.29	74.16±15.20	84.00±9.94	91.07±3.66	92.94±2.70	93.68±1.01
AGRAWAL-3	88.63±8.77	74.30±15.58	90.07±7.12	89.99±6.75	94±2.54	94.35±0.99
AGRAWAL-4	88.01±9.00	76.48±10.12	86.84±9.00	88.75±9.46	94.87±1.81	94.47±1.41
SEA-1	88.63±1.58	87.16±1.96	89.01±1.47	89.01±1.96	89.78±1.12	89.77±1.10
SEA-2	88.60±1.58	86.98±2.10	89.04±1.58	88.99±1.64	89.54±1.21	89.70±1.11
SEA-3	86.45±4.49	87.25±1.74	89.06±1.29	86.97±4.78	88.83±3.05	89.47±1.97
SEA-4	88.28±2.53	89.09±0.96	89.75±1.43	86.94±4.82	89.22±2.60	90.08±0.97
SPAM CORPUS	80.35	75.71	87.44	82.74	88.79	93.13

Table 2: Average accuracy

classifiers.

For a precise comparison, 30 executions on each configuration were made. In each execution, the random seed used to randomize attributes on the generators is different, causing differences on each execution. The averages and standard deviations shown in Table 2 were calculated using the results of these 30 executions. Nevertheless, Anderson-Darling [3], Lilliefors [18] and Shapiro-Wilk [22] normality tests has shown that the distributions are not normal. Therefore, we used the Friedman non-parametric test [8] with the Bonferroni-Dunn $1 \times N$ pivotal post-hoc test [10] to create a rank for the algorithms and determine whether there is statistical differences between SFNClassifier and the others with a 95% confidence. Thus, we were able to determine that there is no statistical difference in terms of accuracy between SFNClassifier and the state-of-the-art algorithms, except for DWM. We emphasize the fact of Leveraging Bagging algorithm showing inferior results in both RTG experiments. In both RTG experiments we can observe that there is no statistical difference between SFNClassifier and the higher accuracy bagging algorithms.

In real cases, such as spam detection, it may be necessary to choose a classifier not based only on its accuracy but also on its processing time and space required. It may seem reasonable that a higher number of classifiers may help achieving good results, but SFNClassifier has an average ensemble size of four classifiers and yet presents good overall accuracy. Keeping a smaller set of classifiers helps both in processing time and space requirements once it demands less processing and memory, once there are less classifiers to train and maintain. According to Table 3, we can observe that SFNClassifier has under-average processing time, determining a difference between it and the others. All experiments were performed on a 2.26GHz Intel Core 2 Duo based computer with 4GB of memory. In Figure 2 we present the results for the SEA-3 experiment, where we can notice the fast recovery of SFNClassifier from the drifts in comparison to the other algorithms.

Stream configuration	Classifiers average processing time (s)					
	Hoeffding Tree	DWM	SFNClassifier	Online Bagging	ADWIN Bagging	Leveraging Bagging
RTG-10	9.70	54.43	58.63	71.35	95.86	295.84
RTG-100	116.24	218.91	414.79	549.01	313.55	1612.63
AGRAWAL-1	5.83	33.31	48.80	48.90	69.69	183.15
AGRAWAL-2	6.00	33.80	49.22	59.01	71.69	186.13
AGRAWAL-3	7.56	33.54	49.65	69.52	81.38	231.80
AGRAWAL-4	8.25	32.72	49.37	80.92	78.47	226.17
SEA-1	2.69	15.94	31.70	34.40	38.38	139.81
SEA-2	2.67	15.75	31.66	34.66	38.69	138.61
SEA-3	2.72	15.78	30.21	36.05	48.90	142.30
SEA-4	4.17	13.63	30.52	35.78	48.61	143.53
SPAM CORPUS	190.70	260.74	917.46	1524.08	2071.07	3661.81

Table 3: Average processing time

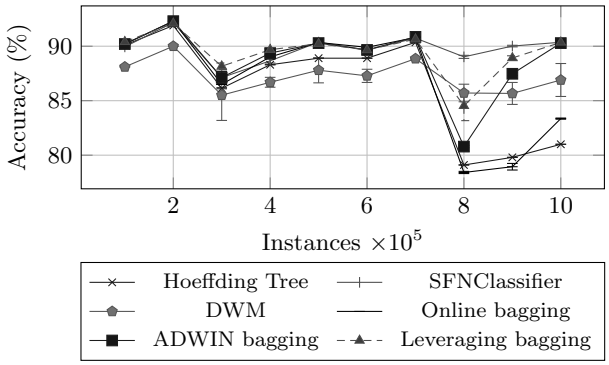

Figure 2: Results for the SEA-3 experiment.

6. CONCLUSIONS

According to the "no free lunch" theorem, it is well known that no classifier will ever perform better than all others for every problem. Thus, the comparison presented has shown that our proposal is competitive in terms of overall accuracy, quickness to re-adapt to concept drifts and processing time. One could argue that the accuracy results doesn't improve the state-of-the-art results, but yet it achieved comparable results lessening the ensemble size determining a decrease also in processing time. In our experiments, we could see that the Leveraging Bagging [5] had problems with RTG experiments, where no drifts occurred. Thus, we emphasize the importance of SFNClassifier showing good accuracy with stationary streams. Jutting the synthetic data streams, a real database was used, in order to show the effectiveness of our proposal on real situations.

One limitation of this work is the impossibility to perform the task of regression. Future work includes the development of a similar approach aiming the regression task also for generic domains. We also pretend on investigating the behaviour of the algorithm for multi-class and multi-label domains and improving the removal strategy in order to make

the algorithm perform better under the circumstances where a simple base learner has shown better overall results. We also realize the importance of the period parameter, where while its value is too low, it becomes possible that the algorithm detects concept drifts where they do not happen and otherwise, whereas it is too high, concept drifts may occur and the network will not evolve properly. A detailed study of the algorithm in different stream configurations with concept drift will be developed in order to evidence the semantics between each centrality metric and the ensemble topology and votes. We also plan to study the classifiers average lifetime, other topology metrics and concept drift detection methods in order to improve the algorithm.

7. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swani. Database mining: A performance perspective. *IEEE Trans. on Knowledge and Data Engineering*, 5(6):914–925, Dec. 1993.

[2] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. In *Reviews of Modern Physics*, pages 139–148. The American Physical Society, Jan. 2002.

[3] T. W. Anderson and D. A. Darling. Asymptotic theory of certain goodness of fit criteria based on stochastic processes. *Annals of the Mathematical Statistics*, 23(2):193–212, Jun. 1952.

[4] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.

[5] A. Bifet, G. Holmes, and B. Pfahringer. Leveraging bagging for evolving data streams. In *Machine Learning and Knowledge Discovery in Databases*, pages 135–150. ECML PKDD, Sep. 2010.

[6] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148. ACM SIGKDD, Jun. 2009.

[7] C. D. Correa, T. Crnovrsanin, and K.-L. Ma. Visual reasoning about social networks using centrality sensitivity. *IEEE Trans. on Visualization and Computer Graphics*, 18(1):106–120, 2012.

[8] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, Dec. 2006.

[9] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM SIGKDD, Sep. 2000.

[10] J. Dunn and O. J. Dunn. Multiple comparisons among means. *American Statistical Association*, pages 52–64, 1961.

[11] P. Erdos and A. Rényi. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[12] J. Gama and P. Rodrigues. Issues in evaluation of stream learning algorithms. In *Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 329–338. ACM SIGKDD, Jun. 2009.

[13] H. M. Gomes and F. Enembreck. Sae: Social adaptive ensemble classifier for data streams. In *IEEE Symp. on Computational Intelligence and Data Mining*, pages 1–10. IEEE, 2013.

[14] I. Katakis, G. Tsoumakas, and I. Vlahavas. An adaptive personalized news dissemination system. *Journal of Intelligent Information Systems*, 32(2):191–212, Apr. 2009.

[15] J. Z. Kolter and M. A. Maloof. Using additive expert ensembles to cope with concept drift. In *ICML '05 Proc. of the 22nd international conference on Machine learning*, pages 449–456. ACM, Jun. 2005.

[16] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. In *The Journal of Machine Learning Research*, pages 123–130. JMLR, Jan. 2007.

[17] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley and Sons, New Jersey, 2004.

[18] H. W. Lielliefors. On the kolmogorov-smirnov test for normality with mean and variance. *Journal of the American Statistical Association*, 62(318):399–402, Jun. 1967.

[19] S. Milgram. The small world problem. *Psychology Today*, 1(1):61–67, May 1967.

[20] M. E. J. Newman. *Networks: an introduction*. Oxford University Press, Oxford, 2010.

[21] N. C. Oza and S. Russell. Online bagging and boosting. In *Artificial Intelligence and Statistics*, pages 105–112. Society for Artificial Intelligence and Statistics, Jan. 2001.

[22] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3):591–611, Dec. 1965.

[23] W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-classification. In *Proc. of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM SIGKDD, Aug. 2001.

[24] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, Jun. 1998.

[25] G. Widmer and M. Kubate. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, Apr. 1996.