X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

Streaming Data Classification with the K-associated Graph

João R. Bertini Jr.; Alneu Lopes and Liang Zhao
Instituto de Ciências Matemáticas e de Computação - USP
{bertini,alneu,zhao}@icmc.usp.br

Abstract – Graph-based methods consist in a powerful form for data representation and abstraction, such approach has been each time more considered to solve machine learning related problems. This work presents a graph-based algorithm suitable for streaming data classification problems with non-stationary distribution. The proposed algorithm is an extension of a static algorithm which relies on the following concepts, a graph structure called optimal K-associated graph, that represents the training set as a sparse graph separated into components; and the purity measure for each component, which uses the graph structure to determine local data mixture level in relation to their classes. Here we address classification problems that exhibit modification on the underlying data distribution. This problem qualifies concept drift and worsens any static classifier performance. Hence, in order to maintain accuracy performance, it is necessary for the classifier to keep learning during application phase, for example, by implementing incremental learning. Experimental results, suggest that our algorithm presents advantages over the tested algorithms on streaming classification tasks.

Keywords – Graph-based Learning, Streaming Data Classification, Incremental Learning.

1. INTRODUCTION

Many real-world applications require online classification, where the task is to classify examples at the time they arrive [1] such as intrusion detection [2] and credit card fraud detection [3], among many others. Generally in those applications the underlying data distribution change over time, and often these changes make the model built on old data inconsistent with the new one. This problem, known as concept drift [4], [5], ruins the performance of the current classifier demanding frequently updating of the model. A trivial solution to this problem is to retrain the classifier over certain periods of time. This practice, however, brings at least two prohibitive drawbacks, (i) stopping the classifier for retraining may lose incoming data that needs to be classified and; (ii) retraining process usually is computationally expensive. Fortunately, frequent retraining of a classifier can be avoided using an incremental learning algorithm [6], which enables a classifier to acquire knowledge during classification phase without being retrained, i.e. new knowledge is added, continuously, to the classifier through a more efficient process than training itself. There is plenty of methods that implements incremental learning, a recent review can be found in [7].

Graph-based algorithms applied to data mining tasks have attracted great attention, in a broad range of areas, such as clustering, [8], semi-supervised learning [9] among others [10]. Basically, graph-based approach consists of representing a data set as a set of nodes and connections among them, such structure can further be used as a powerful representation of data; permitting to analyze and visualize local and global connection structure in data within a single framework. Motivated by the richness of graph-based representation as well as new methods provided by graph theory, this paper presents a classification approach which is carried out by finding an optimal purity measure during the graph construction phase. The obtained results of classification on stationary data sets have indicated its potential for dealing with classification problems, particularly in the presence of noise [11]. Moreover, an extension of the technique for dynamical classification is proposed, in which the model continually evolves during the classification phase. This approach, also known as incremental learning [6], allows the classifier to expand its data representation as well as to deal with applications in dynamical environment.

The remainder of the paper is organized as follow. Section 2 briefly exposes the concepts presented in [12] [11], basically covering the K-associated optimal graph, the purity measure and the KAOG classifier. Section 3 describes the proposed extension to cope with streaming classification. Section 4 presents the experiments comparing the proposed approach to four state-of-the-art ensemble method. It also presents some insight on how the classifier works, as well as a comparison between static and incremental approaches in a non-stationary environment. At last, Section 5 concludes the paper.

2. CLASSIFICATION BASED ON THE K-ASSOCIATED OPTIMAL GRAPH

This section briefly exposes the K-associated optimal graph which is used to estimate the class label of new instances, a fully description can be found in [11]. A *K-associated optimal graph* is the final graph structure obtained as a result of sequentially building of several K-associated graphs by increasing K by one, while keeping record of the components with the highest purity found. In order to present the K-associated graph, consider a labeled set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{ip}, c_i)$ the i-th vector-based data composed of p attributes and a class label $c_i \in \Omega = \{\omega_1, \ldots, \omega_\mu\}$, considering a μ-class problem. A *K-associated graph*, in turn, is defined as an unweighted directed graph $G^{(K)} = (V, E)$ where $V = \{v_1, \ldots, v_N\}$ is a set of vertices in which each vertex v_i represent an instance $\mathbf{x}_i$ of the input data set within its class label c_i. In order to obtain the edge set $E = \{e_{ij}\}$ let $\Lambda_{v_i,K}$ represents the set of label independent K nearest neighbors of vertex v_i, obtained through calculation on the data instances by using a given similarity measure. Then, the connections are performed by taking every vertex v_i as an origin, and connect to all destination vertex v_j that satisfies, $v_j \in \Lambda_{v_i,K}$ and $c_i = c_j$.

In order to present how a new instance is classified using the K-associated optimal graph, consider a new unlabeled data instance defined just as the labeled ones, as $\mathbf{y} = (y_1, y_2, ..., y_p)$, excepted that now the class c associated with the new instance $\mathbf{y}$ must be estimated. Also, let v_y be the vertex that represents $\mathbf{y}$. According to Bayes theory [13] the probability *a posteriori* of a new vertex v_y to belong to component C_α given the observed neighborhood of the the vertex v_y, Λ_{v_y} is defined by Eq. (1).

$$P(v_y \in C_\alpha|\Lambda_{v_y}) = \frac{P(\Lambda_{v_y}|v_y \in C_\alpha)P(v_y \in C_\alpha)}{P(\Lambda_{v_y})} \tag{1}$$

Now, we define each term on the right-hand side of Eq. (1). As each component C_α has its own value of K, denoted as K_α, as a result of being formed in a given K-associated graph, the neighborhood Λ_{v_y} must consider K_α. Therefore, the probability of the neighborhood of v_y, supposing that v_y belongs to C_α depends on K_α, and is defined as $P(\Lambda_{v_y}|v_y \in C_\alpha) = |\{\Lambda_{v_y,K_\alpha} \cap C_\alpha\}|/K_\alpha$. Now, the second numerator term of Eq. (1) corresponds to the component *a priori* probability. According to the definition of the purity measure, when normalized, it can act as *a priori* probability for classification. So, purity normalization considers only those components that v_y connects to, i.e. $\Lambda_{v_y,K_\beta} \neq \emptyset$, and can be stated as $P(v_y \in C_\alpha) = \Phi_\alpha / \sum_{\Lambda_{v_y,K_\beta} \neq \emptyset} \Phi_\beta$. Finally, the normalizing term is defined in Eq. (2).

$$P(\Lambda_{v_y}) = \sum_{\Lambda_{v_y,K_\beta} \neq \emptyset} P(\Lambda_{v_y}|v_y \in C_\beta)P(v_y \in C_\beta) \tag{2}$$

In general, there are more components than number of classes, according to Bayes optimal classifier, it is necessary to sum the *posteriori* probabilities of all components corresponding to the same class, as shown in Eq. (3), where $\hat{C_\alpha}$ stands for the class of the vertices in component C_α.

$$P(v_y|\omega_i) = \sum_{\hat{C_\alpha} = \omega_i} P(v_y \in C_\alpha|\Lambda_{v_y}) \tag{3}$$

Finally, the largest values among the *posteriori* probabilities reflects the most probable class for the new instance, according to Eq. (4), where $\varphi(v_y)$ stands for the class attributed for the vertex v_y and consequently to instance $\mathbf{y}$.

$$\varphi(v_y) = \textit{arg max}\left\{P(v_y|\omega_1), \ldots, P(v_y|\omega_\mu)\right\} \tag{4}$$

The classifier is suitable to multi-class ($\mu > 2$) environment due to the sparse component structure of the K-associated optimal graph and to the Bayesian-like way it is derived.

3. THE PROPOSED ALGORITHM FOR STREAMING DATA CLASSIFICATION

The algorithm proposed here, and named KAOGINC, is an extension of the algorithm proposed in [12], which uses a special graph referred to as principal graph, that evolves by adding new knowledge along time, without the need of being retrained. This dynamical evolution is done by growing the principal graph within the optimal graphs built from recent patterns. Each time an unlabeled pattern comes, it is classified by the non-parametric classifier, presented in the previous section.

In a formal way, consider a scenario where labeled and unlabeled data sets are presented as a data stream of the form, $S = \{X_1, Y_1, \ldots, X_T, Y_T\}$ where X_t stand for labeled data sets, while Y_t are unlabeled data sets whose classes must be estimated. The objectives of a classifier aimed at processing this kind of data stream are, keep an acceptable and stable performance regarding classification accuracy, while at the same time, keep low computational and memory costs. According to [6], in order to a classifier satisfy these requirements it must execute the following tasks during application, (i) add new knowledge incoming from labeled data throughout time, (ii) predict the classes of all new unlabeled data and (iii) discard old knowledge that no longer will be. In view of these tasks, consider the general scheme of the the proposed method as in Fig. 1.

Now, we see how KAOGINC cope with incremental learning by associating the aforementioned mentions three tasks of an incremental algorithm to the components depicted in Fig. 1. The task (i) refers to updating the classifier with new concepts; in the proposed algorithm the principal graph is updated every time when a labeled set is presented. The updating procedure consists in building a K-associated optimal graph from the labeled set at hand, X_t, and add it to the principal graph, as shown in Fig. 1(a). Task (ii) corresponds to classifying new unlabeled data, and is performed every time when an unlabeled instance is provided. In the proposed algorithm, this task is realized by using the principal graph as the structure to infer the classes of the new vertices as shown in Fig. 1(b). At last, task (iii) corresponds to component removal and is directly related to the size of the principal graph and to the classifier performance. As shown in Fig. 1(c) the proposed algorithm provide two mechanisms for eliminating misleading components of the principal graph. The principal graph is kept at reasonable size by implementing the pruning - which removes components associated to errors - and the not using rule that determines a maximal number of classifications that a component is allowed to stay in the graph without been used. Algorithm 1 presents the proposed algorithm in details.

Every time a labeled set X_t, is presented to the algorithm, a K-associated optimal graph is generated by using the $KAOG()$ algorithm, as described in [12]. Once built the current graph is added to the principal graph, noted as G_P. Notice that theses

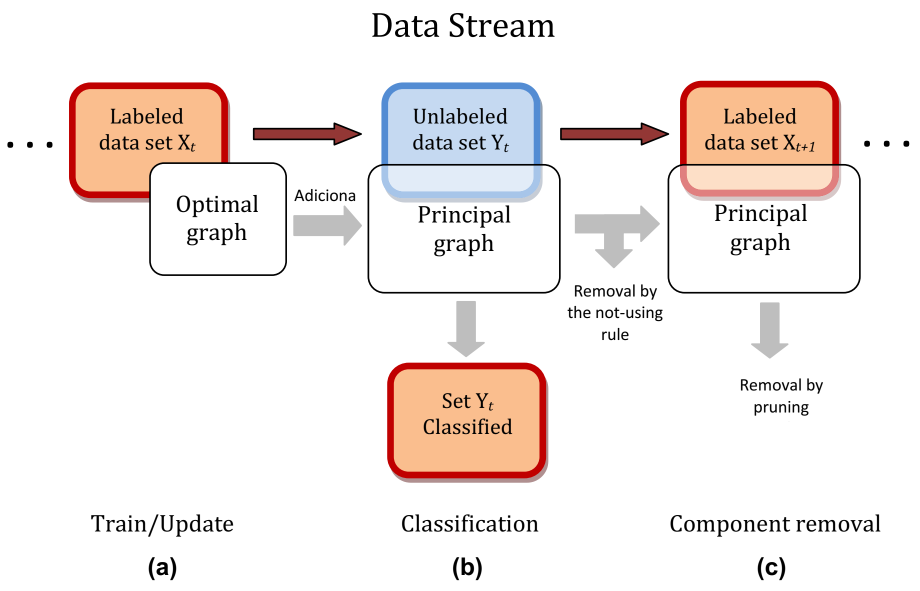

Figura 1: Data Stream processing scheme for the proposed algorithm KAOGINC.

Algorithm 1 Incremental Algorithm KAOGINC

Input: $S = \{X_1, Y_1, ..., X_T, Y_T\}$ {Data stream}; τ {Forgetting parameter - set by user}
Symbols: G_P {Principal graph}; $KAOG()$ {K-associated optimal graph builder}; $classifier()$ {Graph-based classifier}

```
1:  repeat
2:      if $X_t$ then
3:          $prunning(G_P, X_t)$
4:          $G_P \Leftarrow G_P \cup KAOG(X_t)$
5:      else
6:          for all $\mathbf{y}_j \in Y_t$ do
7:              $\varphi(\mathbf{y}_j) \Leftarrow classifier(G_P, \mathbf{y}_j)$
8:          end for
9:          for all $C_\alpha \subset G_P$ do
10:             if $t_\alpha > \tau$ then
11:                 $G_P \Leftarrow G_P - C_\alpha$
12:             end if
13:         end for
14:     end if
15: until $S = \emptyset$
```

graphs are in fact a set of disconnected components. The frequent addition of new components in the principal graph updates it with new knowledge and as a consequence avoid drop in performance due to concept drifts. However, it also enlarges the graph, which could lead to uncontrolled growing and consequently, invaliding the classifier. Therefore the algorithm has two methods to take components off the graph, the $pruning()$ method takes off the principal graph those components that are associated with error when the labeled set X_t is presented; and when $t_\alpha > \tau$, where t_α correspond to the number of classification that component C_α has not been used in classification. It turns out that by implementing both removing strategy the principal graph remains at a reasonable size along the time.

4. SIMULATIONS RESULTS

This section presents two groups of simulations; first we give an insight on the KAOG classifier and also a show a motivation for developing its incremental version within a comparison between then in a non-stationary benchmark domain. The second group comprehend the comparison between the proposed incremental algorithm, KAOGINC and four state-of-the-art algorithms in two benchmarks domains.

The KAOG classifier uses neighborhood relations in the classification of new data, this next example explores the differences between the algorithms KAOG and KNN. Consider the set with three classes artificially generated and presented in Fig. 2(a). The surfaces of the decision on the data set shown in Fig. 2(a) are shown in Fig. 2(b) to 1NN, Fig. 2(c) 15NN and Fig. 2(d) for the KAOG algorithm. Despite the limitations in revealing all the features of the algorithms, given a representation in two dimensions with little data, one can still notice some major differences between the algorithms.

As can be seen in Figs 2(b)-(d) methods based on nearest neighbors are inherently nonlinear and multiclass. Besides that, the algorithm is able to adjust KAOG locally to the data distribution, due to the way components are used in classification. As a result of the training process, the optimal graph is composed of components with the highest value of purity, obtained from the various components in the generation of graphs K-associated. Thus, the optimal graph can be viewed as a locally optimized structure that allows the classifier to act according to the local distribution of instances. A direct analogy between the algorithm

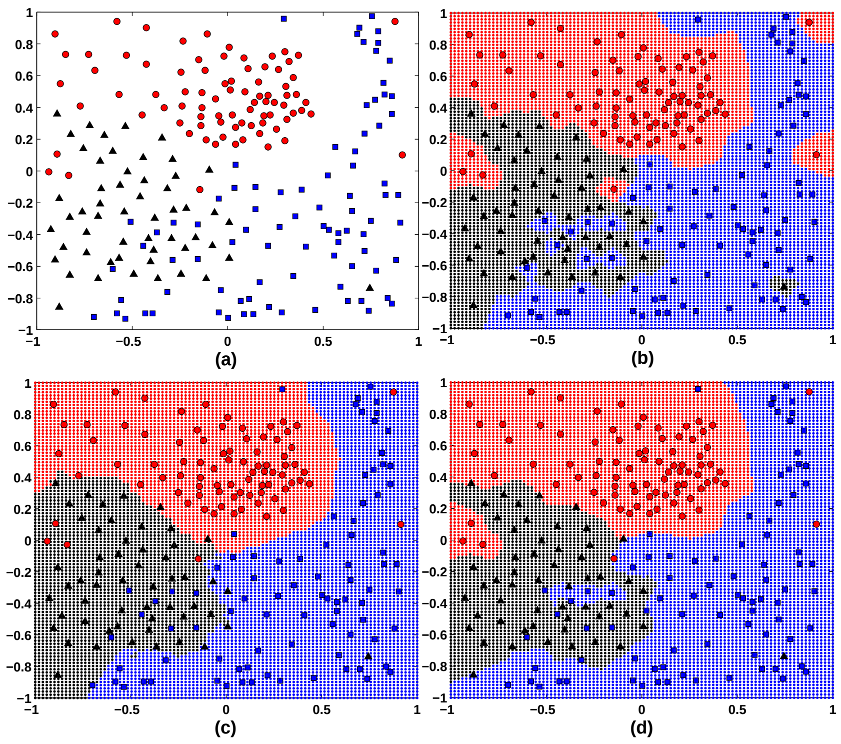

Figura 2: Figure (a) represents a randomly generated data set within three classes; Decision surfaces of (b) 1-NN KNN and (c) 15-NN and (d) KAOG.

and the algorithm KAOG KNN can be established by comparing their respective areas of decision. On KNN algorithm, smaller values of K produces decision surfaces quite adjusted to the sample data, whereas, high values for K produces more general decision surfaces. As the proposed algorithm can store components obtained from different values of K, is a consequence that the surface of locally present decision to feature a certain component. For example, in areas with noise the algorithm tries to find small groups that may form a component, avoiding the creation of several components of a single vertex - which would be similar to the classification made by the algorithm 1NN. While in the relatively pure, the algorithm tries to increase K in order to increase the influence of this area in the standings. It is worth mentioning that the static algorithm KAOG also has mechanisms for the detection and removal of *outliers*, which consists of removing the isolated vertices of the optimal graph.

For comparing the static and the incremental approach, consider the following. Both learning approaches start by training with an 80 patterns training set which results in an optimal network in both cases static and incremental. The rest of the SEA data set is divided in batches, 80 patterns for training batches and 60 for testing batches. The incremental algorithm receives the batches in a interchangeably manner, and performs incremental learning by adding new training pattern to the principal graph. The addition of a new pattern to the network may cause it to be no longer optimal but still, by doing so, it acquire new knowledge about the environment which enables it to deal with changes in the underlying concept. The static classifier instead, only receives the test batches. For every batch an error mean is calculated, the train statistic is done by presenting the next batch of train to the classifier as a test set. Figure 3 presents a comparison concerning our model in two different learning approaches, static and incremental, within four data sets with concept drift. The data sets were generated according to the following thresholds: Figure 3(a) $\theta = \{7, 9, 6, 9.5\}$, Fig. 3(b) with $\theta = \{8, 9, 7, 9.5\}$, Fig. 3(c) with $\theta = \{3, 5, 7, 9\}$ and Fig. 3(d) with $\theta = \{9.5, 8, 6, 4\}$. The results are a averaged by 20 runs of each learning approach considering different generations of the SEA data set.

As expected the static classifier presents a good accuracy with patterns of the same concept it had learned or a close concept. But, it has no mechanisms to recognize a change in the concept and consequently cannot react to it. The incremental classifier, however, is able to recognize a concept change and to restructure the network to deal with the new scenario. In Fig. 3(a), note that in the third concept the static classifier performs better than the incremental. This fact it is due to the third concept be a related to the first one, in which the static classifier was trained. The reason why the incremental classifier presents decay in performance is due to the fact that it just had adapted to the second concept. But as it can be seen in Fig. 3 the classifier also adjusts to the third concept as well. Figure 3(b) stands for the statistic in the data set with less abrupt changes in the concept, as a result the static classifier shows little changes in accuracy, but again the incremental classifier shows presents capability to recover. Figures 3(c) and (d) presents results concerning the raise of one class in relation to the other, Fig. 3(c) represent a raise of class 1, and Fig. 3(d) a raise of class 2. In both cases, the static classifier is unable to follow such changes and as consequence its accuracy gets worse in every concept change. The incremental classifier, however, recognizes the concept change and recovers its accuracy to acceptable values. As one can see the incremental classifier had shown a better performance, not only in concept changes but also on a stead concept it keeps getting better as the data stream keep coming. In the changing concepts it is expected a decay on the performance for any classifier, but the most important is to recover to a good level of accuracy.

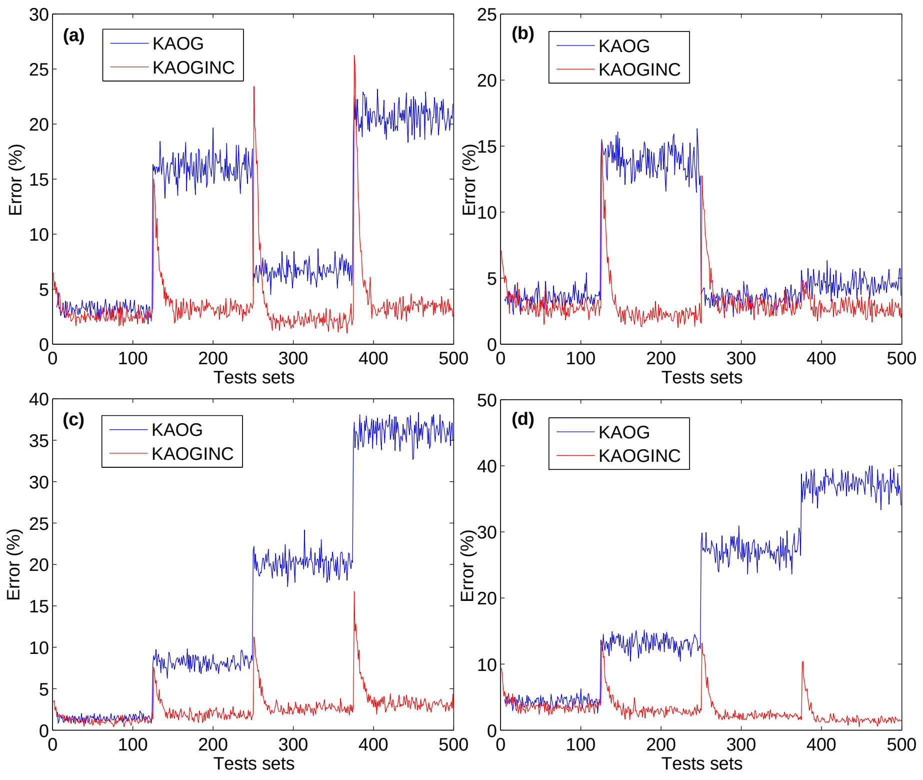

Figura 3: Error rates comparison between Static and Incremental learning on a data set with concept drift, considering the following thresholds (a) $\theta = \{7, 9, 6, 9.5\}$; (b) $\theta = \{8, 6, 8, 7.5\}$; (c) $\theta = \{3, 5, 7, 9\}$; (d) $\theta = \{9.5, 8, 6, 4\}$.

Consider now the second group of simulation, which consist in comparing the proposed algorithm KAOGINC for four other algorithms, named SEA [14], DWM-NB [15], WCEA [3] and OnlineTree2 [5]. The experiments considered three domains, the already cited SEA [14], the moving Sine and the moving Gaussians [16]. The results are a mean over 20 runs for each generation of each domain. Figure 4 illustrates the performance of the algorithms on the SEA domain, for every test set along the time.

Note in Fig. 4 that the OnlineTree2 algorithm performs well until the second concept drift occurs, and after that it takes too long to resume performance. This is due to the fact that the algorithm OnlineTree2 does not react well to abrupt changes of concept, since it takes several iterations to adapt. However, the ensemble based algorithms, like SEA, DWM-NB and WCEA have high adaptation capacity, given that the ensemble can restore itself with new classifiers in a number of iterations equal to half the capacity of the committee, for unweighted ensembles like SEA and less iterations than that for weighted ensembles, because weight may be used to favor those classifier trained on the newest concept, as is the instance of algorithms WCEA and DWM-NB. Still, the best performance is obtained by the proposed classifier KAOGINC, showing its high level of adaptability to concept drifts. Consider now the results for the Sine domain, presented in Fig. 5, which is composed of three abrupt changes of concept.

Notice, in Fig. 5, that the OnlineTree2 algorithm presents the lowest adaptation performance since it does not update as quick as the ensemble based algorithms or the algorithm KAOGINC. Also, the proposed algorithm achieved the best result, especially during the periods of static concept. Despite the high error rate at the points of concept drifts (reaching errors rate around 85%), the KAOGINC presents excellent recovery capacity, even better than the WCEA algorithm (Fig. 5), which is the one with the most stable performance for this domain. At last, the results for the Gaussian domain are depicted in Fig. 6.

As in the Sine domain, the Gaussian domain, depicted in Fig. 6, presents abrupt drift which requires quick recovering from the classifiers. The algorithm WCEA, once again, presents a quite stable performance with the cost of a slower recovering in comparison to the aforementioned algorithms, KAOGINC and SEA. The best performance, though, is presented by the DWM-NB algorithm as shown in Fig. 6, both in accuracy and recovering performance.

5. CONCLUSION

This paper proposed an incremental classifier based on the K-associated optimal graph and in the KAOG static classifier, named KAOGINC. This extension have proved that graph-based classification methods not only can be successfully used classifying stationary distributed data but also consists of a form of representation that allows frequent updating of concepts providing steady performance along time. Initial results on the classification of non-stationary data show a favorable scenario for the presented algorithm. Future works includes refining the graph updating process and consider real applications as well.

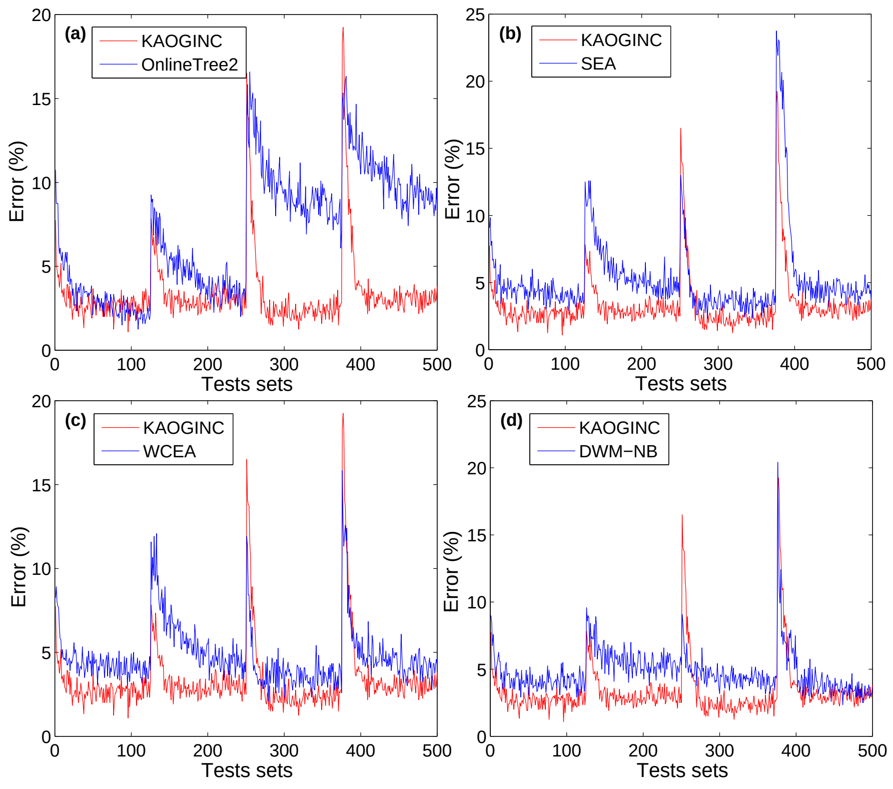

Figura 4: Comparing the algorithm KAOGINC against the algorithms, OnlineTree2, SEA, WCEA and DWM-NB in the SEA domain within the set of threshold $\theta = \{7, 9, 6, 9.5\}$.

REFERENCES

[1] C. Yang and J. Zhou. "Non-Stationary Data Sequence Classification using Online Class Priors Estimation". *Pattern Recognition*, vol. 41, pp. 2656–2664, 2008.

[2] T. Lane and C. Brodley. "Approaches to Online Learning and Concept Drift for User Identification in Computer Security". In *Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD'08)*, pp. 259–263, 1998.

[3] H. Wang, W. Fan, P. Yu and J. Han. "Mining Concept-Drifting Data Streams using Ensemble Classifiers". In *Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD'03)*, pp. 226–235, 2003.

[4] G. Widmer and M. Kubat. "Learning in the Presence of Concept Drift and Hidden Contexts". *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.

[5] M. Núnez, R. Fidalgo and R. Morales. "Learning in Environments with Unknown Dynamics: Towards more Robust Concept Learners". *Journal of Machine Learning Research*, vol. 8, pp. 2595–2628, 2007.

[6] C. Giraud-Carrier. "A Note on the Utility of Incremental Learning". *AI Communications*, vol. 13, no. 4, pp. 215–223, 2000.

[7] M. Maloof and R. Michalski. "Incremental learning with partial instance memory". *Artificial Intelligence*, vol. 154, pp. 95–126, 2004.

[8] S. Schaeffer. "Graph Clustering". *Computer Science Review*, vol. 1, pp. 27–34, 2007.

[9] M. Culp and G. Michailidis. "Graph-Based Semisupervised Learning". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 1, pp. 174–179, 2008.

[10] D. Cook and L. Holder. "Graph-based data mining". *IEEE Intelligent Systems*, vol. 15, no. 2, pp. 32–41, 2000.

[11] J. Bertini Jr., L. Zhao, R. Motta and A. Lopes. "A Nonparametric Classification Method based on K-Associated Graphs". *Information Sciences*, p. doi:10.1016/j.ins.2011.07.043, 2011.

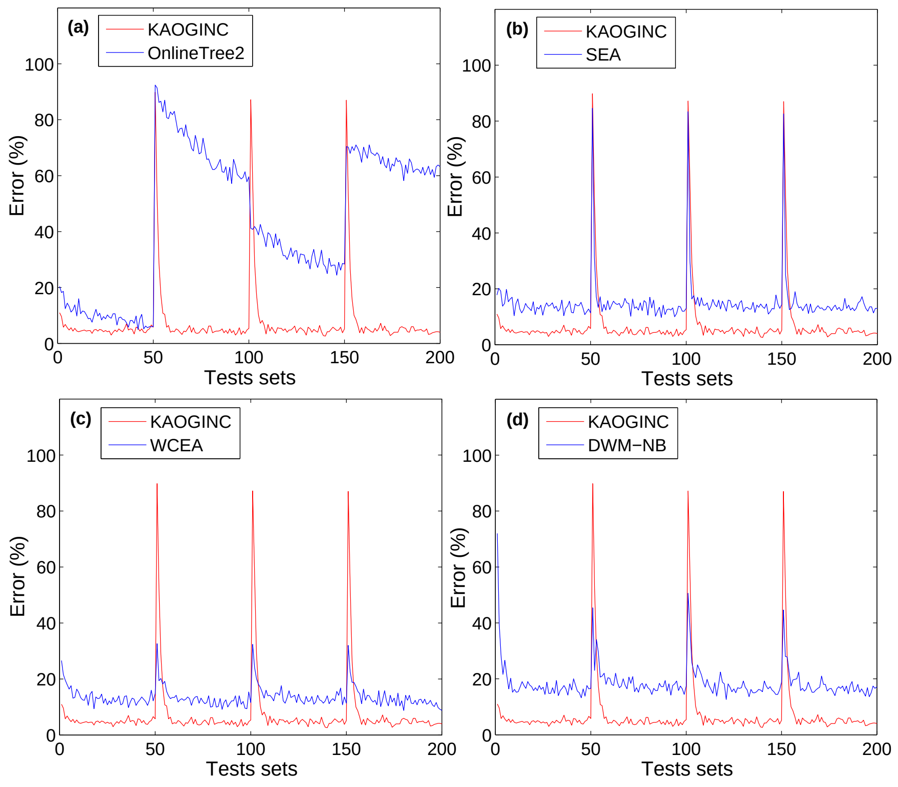

Figura 5: Comparing the algorithm KAOGINC against the algorithms, OnlineTree2, SEA, WCEA and DWM-NB in the moving Sine domain.

[12] A. A. Lopes, J. R. Bertini Jr., R. Motta and L. Zhao. "Classification Based on the Optimal K-Associated Network". In *Proc. Int'l Conf. Complex Sciences: Theory and Applications (COMPLEX'09)*, volume 4 of *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (LNICST)*, pp. 1167–1177. Springer, 2009.

[13] T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, 2009.

[14] N. Street and Y. Kim. "A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification". In *Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD'01)*, pp. 377–382. ACM N.Y., 2001.

[15] J. Z. Kolter and M. A. Maloof. "Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts". *Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[16] J. Gama, P. Medas, G. Castillo and P. Rodrigues. "Learning with Drift Detection". In *Proceedings of the Brazilian Symposium on Artificial Intelligence*, volume 3171, pp. 286–295. Springer, 2004.

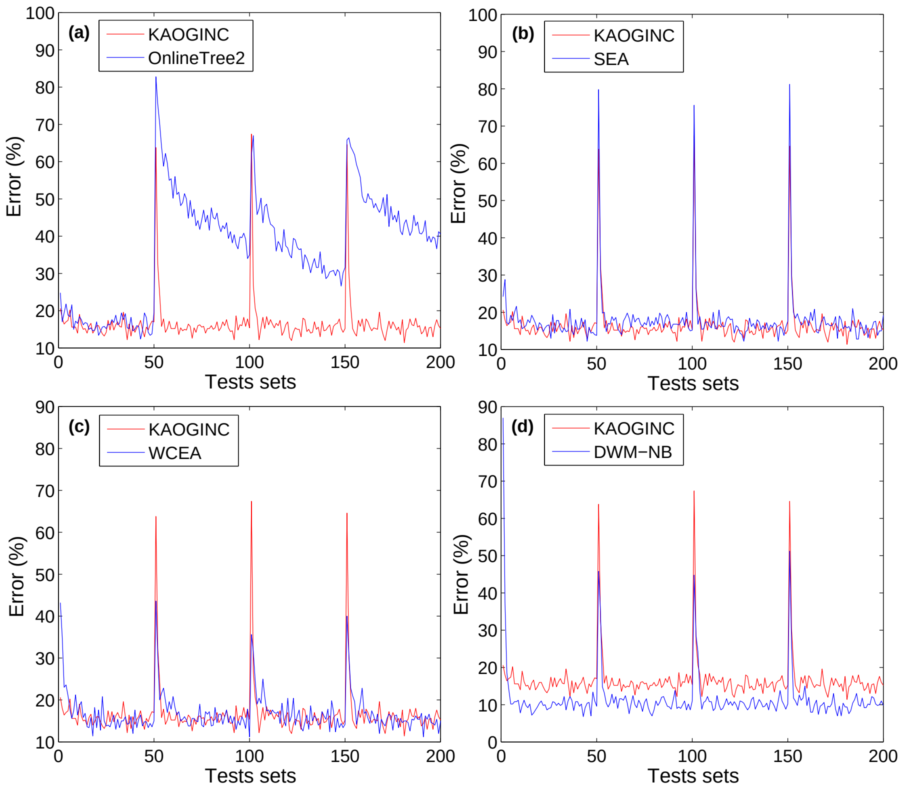

Figura 6: Comparing the algorithm KAOGINC against the algorithms, OnlineTree2, SEA, WCEA and DWM-NB in the moving Gaussians domain.