IGMM-CD: a gaussian mixture classification algorithm for data streams with concept drifts

Brazilian Conference on Intelligent Systems, IV, 2015, Natal.
http://www.producao.usp.br/handle/BDPI/49973

IGMM-CD: A Gaussian Mixture Classification Algorithm for Data Streams with Concept Drifts

Luan Soares Oliveira, Gustavo E. A. P. A. Batista
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, SP, Brazil
{luanso,gbatista}@icmc.usp.br

***Abstract*—Learning concepts from data streams differs significantly from traditional batch learning, because in data streams the concepts to be learned may evolve over time. Incremental learning paradigm is a promising approach for learning in a data stream setting. However, in the presence of concept drifts, outdated concepts can cause misclassifications. Although several incremental Gaussian mixture models methods have been proposed in the literature, we notice that these algorithms lack an explicit policy to discard outdated concepts. In this paper, we propose a new incremental algorithm for data stream learning based on Gaussian Mixture Models. The proposed method is compared to various algorithms widely used in the literature, and the results show that it is competitive with them in various scenarios, overcoming them in some cases.**

***Keywords*—Incremental learning, data stream, concept drift, gaussian mixture model**

I. Introduction

Learning concepts from data streams differs significantly from traditional batch learning. In batch learning there is an implicit assumption that the concept to be learned is static and does not evolve significantly over time. Therefore, it is possible to collect a set of training data and induce a model that will be applied to future unknown data, assuming that both data samples have the same distribution.

In data stream learning, the concepts to be learned may evolve over time. This evolution is called *concept drift* [1]. Thus, the creation of a fixed training set is no longer applicable. In fact, many data streams are potentially infinite, so we can not accumulate all the examples that occur in the stream in a training set. Usually, we assume that each example in a data stream is seen only once, and then discarded [2].

The incremental learning paradigm is a promising approach for learning in a data stream setting [1], [3]–[6]. In incremental learning, the classification model is updated for each new event, without the need to review all previous training examples. Thus, incremental algorithms can be highly efficient in terms of memory usage.

In this paper, we propose a new incremental algorithm for data stream learning based on Gaussian Mixture Models (GMM). Although other incremental algorithms for Gaussian mixture models have been proposed in the literature, such as IGMM [3], we notice that these algorithms lack an explicit policy to discard outdated concepts.

In the presence of concept drifts, outdated concepts can cause misclassifications, increase processing times and memory requirements. This can be easily observed in the GMM model, since all concepts are described by Gaussian components. The components that represent outdated concepts are likely to cause misclassification since they do not represent the current data distribution; increase processing time since the GMM inference model requires to verify all components that match a new instance; and require additional memory to store the component parameters.

We use IGMM as starting point and evaluate its behavior in the presence of concept drifts. We then propose a policy to discard outdated concepts and evaluate its performance in terms of classification accuracy and processing time in comparison to IGMM. We name our method IGMM-CD (IGMM with support to Concept Drifts).

The proposed method is also compared to various algorithms widely used in the literature, available in the Massive Online Analysis (MOA)[1] environment, such as Naive Bayes, Hoeffling Adaptive Tree [7]–[9]. The results show that the proposed algorithm is competitive with them in various scenarios, overcoming them in some cases.

This paper is organized as follows: Section II describes the conventional GMM, with its fundamental equations. Section III shows the incremental version of GMM (IGMM) proposed by [3]. The changes proposed in this paper are described in Section IV. Section V presents the results and comparisons of IGMM-CD with the state-of-the-art in data stream classification. Finaly, Section VI presents our conclusions and directions to future research.

II. Gaussian Mixture Model

The Gaussian mixture model is a statistical modeling tool that has been successfully used in diverse applications in supervised and unsupervised tasks.

In classification, each example, seen as a D dimensional characteristics vector, is assigned to a pre-determined set of classes with a certain probability of belonging to each class. The GMM algorithm seeks to maximize the likelihood function, which provides a measure of the way in which the probability distribution function (*pdf*) fits the dataset. Each *pdf* consists basically of three parameters: w_i (weight or priori probability of Gaussian component i), μ_i (mean of the

[1] http://moa.cms.waikato.ac.nz/

978-1-5090-0016-6/15 $31.00 © 2015 IEEE
DOI 10.1109/BRACIS.2015.61

Gaussian component i) and Σ_i (variance or covariance of the Gaussian component i) [10]. Given these parameters, the algorithm will seek to estimate them so that the adjustment of the *pdf* on the training data is the best possible, i.e., the value of the likelihood function is as large as possible. To perform this estimation it is common to use the Expectation-Maximization (EM) algorithm, which ensures that the likelihood function is not decreasing and converges at least to a local maximum.

The EM algorithm works in two steps that are repeated until a convergence is reached, and at each iteration the parameters w_i, μ_i and Σ_i are adjusted to maximize the likelihood function.

Lets $\theta = (w_i, \mu_i, \Sigma_i)$ be the set of Gaussian mixtures parameters, x a D-dimensional continuous-valued data vector (i.e. measurement or features), the Gaussian mixture model is a weighted sum of M Gaussian component densities given by Equation 1.

$$p(x|\theta) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i) \tag{1}$$

with the restriction that $\sum_{i=1}^{M} w_i = 1$.

In this equation $g(x|\mu_i, \Sigma_i)$ represents each D-dimensional Gaussian component, defined by Equation 2.

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} exp\left\{\frac{1}{2}(x-\mu_i)^\intercal \Sigma_i^{-1}(x-\mu_i)\right\} \tag{2}$$

As already mentioned, the EM algorithm is used to estimate the θ parameter set that maximizes the likelihood of the Gaussian mixture model given some observed data. Therefore, consider a training set X = $(x_1, x_2, x_3, ..., x_n)$, the likelihood function assumes independence between events [10] and can be expressed as in Equation 3.

$$\mathcal{L}(\theta, X) = \prod_{i=1}^{n} p(x_i|\theta) \tag{3}$$

Such function is not linear on θ. This makes its direct maximization not possible. However, we can use EM to estimate the parameter set.

The EM algorithm uses a given initial θ to estimate a new parameter set $\overline{\theta}$ such that $\mathcal{L}(\overline{\theta}, X) \geq \mathcal{L}(\theta, X)$. In the next iteration, the new model becomes the initial model, and the process is repeated until it reaches a convergence criterion.

Estimation of the θ parameter set of each EM iteration follows the Equations 4, 5 and 6.

$$\overline{w}_i = \frac{1}{n}\sum_{j=1}^{n} Pr(i|x_j, \theta) \tag{4}$$

$$\overline{\mu}_i = \frac{\sum_{j=1}^{n} Pr(i|x_j, \theta)x_j}{\sum_{j=1}^{n} Pr(i|x_j, \theta)} \tag{5}$$

$$\overline{\Sigma}_i = \frac{\sum_{j=1}^{n} Pr(i|x_j, \theta)x_j^2}{\sum_{j=1}^{n} Pr(i|x_j, \theta)} - \overline{\mu}_i^2 \tag{6}$$

given a posteriory probability of a component i expressed as Equation 7.

$$Pr(i|x_j, \theta) = \frac{w_i g(x_j|\mu_i, \Sigma_i)}{\sum_{k=1}^{M} w_k g(x_j|\mu_k, \Sigma_k)} \tag{7}$$

It is important to note that the original EM algorithm is not incremental. Thus, the update equations (4, 5 and 6) require knowledge of the probabilities of all the examples viewed so far. This requirement is incompatible with the data stream scenario, where the algorithm can receive examples anytime.

In the incremental algorithm, the order in which the examples are given is important. Therefore, the notation so far will be modified to add an overwritten (t) indicating the time instant at which the variable assumes a determined value.

III. Incremental Gaussian Mixture Model

Similarly to the EM algorithm, the **I**ncremental **G**aussian **M**ixture **M**odel (IGMM) [3] performs the modeling of Gaussian mixtures probability distributions. However, the incremental approach allows each Gaussian component parameters to be adjusted as soon as each new example is computed, followed by approximately incremental equations. Thus the model may be updated as new relevant information arrives in the data stream.

At this point, since the incremental algorithm creates a new model for each incoming example, we need to extend our notation. From here on we use the superscript (t) to indicate the different models in time.

IGMM uses some measures to control the number of components required to represent the data already seen. The model begins with a single component having *a priori* probability, $w_1^{(1)} = 1$ and $\mu_1^{(1)}$ equals to the first observed example. Also a standard diagonal covariance matrix is used as $\Sigma_1^{(1)} = \sigma_{ini}^2 I$, with σ_{ini} being previously defined by the user and I being the identity matrix.

New components are added to the model using a minimal likelihood criterion. Each new data point $x^{(t)}$ is checked if it fits some Gaussian component i with probability $p(x^{(t)}|i) = g(x^{(t)}|\mu_i^{(t-1)}, \Sigma_i^{(t-1)})$ above a minimum value. If $p(x^{(t)}|i)$ does not reach the minimum value for all components, the data point is considered new information and a Gaussian component is added to the model with parameters set as described above.

The value of the minimum likelihood criterion C_{ver} may be defined as a fraction of the maximum likelihood function, so that the addition of a new component occurs when:

$$p(x^{(t)}|i) < \frac{C_{ver}}{(2\pi)^{D/2}|\Sigma_i^{(t-1)}|^{1/2}}, \forall i \tag{8}$$

This parameter has a simple intuition, it indicates how "distant" an example $x^{(t)}$ should be from $\mu_i^{(t-1)}$ to not be considered as a member of component i.

However, points that meet this criterion and thus fit into an existing component must be incorporated into the model, leading to the need to update its parameters. The IGMM is based on an incremental version of the EM process.

When an example $x^{(t)}$ matches one or more components, it is necessary to know the *a posteriori* probabilities of these components for all the already analyzed examples to update the Gaussian mixture parameters, as Equations 4, 5 and 6 show. Unfortunately, storing all these values for a data stream would be inconceivable. The IGMM solves this problem by storing a variable named sp for each existing component. According to [3], such variables must be reset periodically to avoid possible saturation. Equation 9 shows the update rule for this variable.

$$sp_i^{(t)} = sp_i^{(t-1)} + Pr(i|x^{(t)}, \theta^{(t-1)}) \tag{9}$$

Given the sum of *a posteriori* probabilities for each component, IGMM can replace Equations 4, 5 and 6 by its approximately incremental versions. Equations 10, 11 and 12 list the incremental formulas for $w^{(t)}$, $\mu^{(t)}$ and $\Sigma^{(t)}$.

$$w_i^{(t)} = \frac{sp_i^{(t)}}{\sum_{j=1}^{M} sp_j^{(t)}} \tag{10}$$

$$\mu_i^{(t)} = \mu_i^{(t-1)} + \frac{Pr(i|x^{(t)}, \theta^{(t-1)})}{sp_i^{(t)}}(x^{(t)} - \mu_i^{(t-1)}) \tag{11}$$

$$\Sigma_i^{(t)} = \Sigma_i^{(t-1)} - (\mu_i^{(t)} - \mu_i^{(t-1)})(\mu_i^{(t)} - \mu_i^{(t-1)})^\intercal + \frac{Pr(i|x^{(t)}, \theta^{(t-1)})}{sp_i^{(t)}}[(x^{(t)} - \mu_i^{(t)})(x^{(t)} - \mu_i^{(t)})^\intercal - \Sigma_i^{(t-1)}] \tag{12}$$

IGMM has two parameters that need to be defined before its execution. According to [3], the parameter σ_{ini} is not critical and needs only to be large enough to prevent components to cover just one example. However, setting a value for the C_{ver} parameter is more critical since it influences the algorithm sensitivity to concept drifts.

IV. IGMM-CD - Incremental Gaussian Mixture Model with Concept Drift

In this section we present our proposal. We divide the explanation in two parts. The first one describes the modifications necessary to use the IGMM as a classification algorithm. The second part explains how we adapted IGMM to concept drifts.

A. Incremental Gaussian Mixture Model for Classification

IGMM as defined in the previous section is a valuable tool for estimating the *pdf* of a data sample. It can be used in a large set of tasks, such as clustering. In that case, we could assume that each Gaussian component defines a cluster and each point can belong to a cluster with a certain probability.

In this paper, we are interested in classification problems. Therefore, we must first expand the IGMM model to classification tasks. This can be done by creating a GMM for each class or by maintaining a single GMM for all classes and annotating each Gaussian component with a class label. We describe the second approach in this section; although, both approaches will certainly result in very similar outcomes.

We present our extension of IGMM for classification using algorithms. Therefore, we can better detail each necessary step as well as the correct order of the steps.

We note here that to classify a new event in a data stream, we first receive the unlabeled example $x^{(t)}$ and provide a classification based on the current GMM model which has the parameter set $\theta^{(t-1)}$. Once a classification is provided we assume that we will receive the true class label for $x^{(t)}$ and are in position to update the model parameters to $\theta^{(t)}$. This sequence of events is very important and updating the model before providing a classification would be unfair and lead to over-optimistic performance results.

Algorithm 1 presents the general framework of our proposal. Notice that although we keep a single GMM for all classes, the algorithm only updates the components that match the minimum likelihood criterion and belong to the same class as the current example.

Require: $X = (x^{(1)}, \ldots, x^{(n)})$ {Ordered sequence of examples for learning}
$Y = (y^{(1)}, \ldots, y^{(n)})$ {Ordered sequence of true class labels of X}
σ_{ini} {Initial covariance matrix}
C_{ver} {Minimum likelihood criterion}
Ensure: $\bar{Y} = (\bar{y}^{(1)}, \ldots, \bar{y}^{(n)})$ {Ordered sequence of predicted labels}
$\theta^{(1)} = \emptyset$
for $x^{(t)} \in X$ **do**
  $\bar{y}^{(t)} \leftarrow classify(x^{(t)}, \theta^{(t-1)})$
  $y^{(t)} \leftarrow true\_class(x^{(t)})$
  $new\_component \leftarrow TRUE$
  for $i \leftarrow 1$ to M **do**
    if $p(x^{(t)}|i) \geq \frac{C_{ver}}{(2\pi)^{D/2}|\Sigma_i|^{1/2}}$ **and** $class(i) = y^{(t)}$ **then**
      $new\_component = FALSE$
    end if
  end for
  if $new\_component = TRUE$ **then**
    $create\_new\_component(x^{(t)}, y^{(t)}, \sigma_{ini})$
  else
    $\theta^{(t)} = update\_components(x^{(t)}, \theta^{(t-1)})$
  end if
  $remove\_outdated\_components()$
end for

Algorithm 1: IGMM-CD

There are four sub-routines that need to be defined. The sub-routine *classify* returns a predicted label $\bar{y}^{(t)}$ for each example $x^{(t)}$ in the stream. We evaluate two possibilities for this sub-routine. The first one, named *global* uses a *maximum a posteriori* (MAP) criterium to return the class with highest sum of *a posteriori* probabilities. Algorithm 2 presents a pseudo-code for this procedure.

Require: $x^{(t)}$ {Example to be classified}
$\theta^{(t-1)}$ {Set of GMM parameters}
Ensure: c is the MAP class
return $arg\ max_{c}\ p(c|x^{(t)}, \theta^{(t-1)}) = \sum_{i=1}^{M} g(x^{(t)}|\mu_i^{(t-1)}, \Sigma_i^{(t-1)}) w_i^{(t-1)} [class(i) = c]$;

Algorithm 2: Global *classify* sub-routine

where, $[class(i) = c]$ stands for the Iverson bracket, i.e., it evaluates to 1 if the class of component i is c and 0 otherwise.

We wonder in a data stream with concept drift scenario, whether a more *local* approach of updating components and classifying examples would be beneficial. In particular, examples that represent new concepts should have little or no effect in updating components which represent outdated concepts. Such an approach would allow a faster creation of new components to represent recent concepts. Although this "local" approach can be implemented in different degrees, here we just evaluate the extreme case in which only one component is updated and used to provide a classification. Algorithm 3 presents a pseudo-code for the local classification procedure.

Require: $x^{(t)}$ {Example to be classified}
$\theta^{(t-1)}$ {Set of GMM parameters}
Ensure: c is the class provided by component with highest probability
return $class(arg\ max_{j}\ g(x^{(t)}|\mu_j^{(t-1)}, \Sigma_j^{(t-1)}) w_j^{(t-1)})$

Algorithm 3: Local *classify* sub-routine

In a similar way, we have evaluated two procedures for updating the components. The first one is "global" and updates all components that belong to the same class as the current example. The update rules are the same as the ones defined by Equations 10, 11 and 12. Algorithm 4 list them for clarity reasons.

Require: $x^{(t)}$ {Current example}
$y^{(t)}$ {True class label of $x^{(t)}$ }
$\theta^{(t-1)}$ {Set of GMM parameters}
Ensure: $\theta^{(t)}$ {Updated GMM parameter set}
$sp^{(t)}$ {Updated sp parameter}
for $i \leftarrow 1$ to M **do**
if $class(i) = y^{(t)}$ **then**
$sp_i^{(t)} = sp_i^{(t-1)} + Pr(i|x^{(t)}, \theta^{(t-1)})$
$\mu_i^{(t)} = \mu_i^{(t-1)} + \frac{Pr(i|x^{(t)}, \theta^{(t-1)})}{sp_i^{(t)}}(x^{(t)} - \mu_i^{(t-1)})$
$\Sigma_i^{(t)} = \Sigma_i^{(t-1)} - (\mu_i^{(t)} - \mu_i^{(t-1)})(\mu_i^{(t)} - \mu_i^{(t-1)})^\intercal + \frac{Pr(i|x^{(t)}, \theta^{(t-1)})}{sp_i^{(t)}}[(x^{(t)} - \mu_i^{(t)})(x^{(t)} - \mu_i^{(t)})^\intercal - \Sigma_i^{(t-1)}]$
$w_i^{(t)} = \frac{sp_i^{(t)}}{\sum_{j=1}^{M} sp_j^{(t)}}$
end if
end for

Algorithm 4: Global *update_components* sub-routine

Algorithm 5 presents the pseudo-code for the "local" update of components. The idea is to update the same component that provided the classification. In both cases (global and local), when $\bar{y}^{(t)} \neq y^{(t)}$, IGMM-CD does not update the component, but creates a new component based on $x^{(t)}$ and $y^{(t)}$ values as described in Algorithm 1.

Require: $x^{(t)}$ {Current example}
$y^{(t)}$ {True class label of $x^{(t)}$ }
$\theta^{(t-1)}$ {Set of GMM parameters}
Ensure: $\theta^{(t)}$ {Updated GMM parameter set}
$sp^{(t)}$ {Updated sp parameter}
$i \leftarrow arg\ max_{j}\ g(x^{(t)}|\mu_j^{(t-1)}, \Sigma_j^{(t-1)}) w_j^{(t-1)}$
$sp_i^{(t)} = sp_i^{(t-1)} + Pr(i|x^{(t)}, \theta^{(t-1)})$
$\mu_i^{(t)} = \mu_i^{(t-1)} + \frac{Pr(i|x^{(t)}, \theta^{(t-1)})}{sp_i^{(t)}}(x^{(t)} - \mu_i^{(t-1)})$
$\Sigma_i^{(t)} = \Sigma_i^{(t-1)} - (\mu_i^{(t)} - \mu_i^{(t-1)})(\mu_i^{(t)} - \mu_i^{(t-1)})^\intercal + \frac{Pr(i|x^{(t)}, \theta^{(t-1)})}{sp_i^{(t)}}[(x^{(t)} - \mu_i^{(t)})(x^{(t)} - \mu_i^{(t)})^\intercal - \Sigma_i^{(t-1)}]$
$w_i^{(t)} = \frac{sp_i^{(t)}}{\sum_{j=1}^{M} sp_j^{(t)}}$

Algorithm 5: Local *update_components* sub-routine

The create_new_component can be trivially implemented by adding a new component to the data structure that maintains all Gaussians components. As this sub-routine is simple and implementation-dependent, we will not provide an algorithm for it.

The last sub-routine is remove_outdated_components. In our framework, this subroutine is performed for each new example. We present a few different criteria in the next section.

B. Gaussian Component Removal Criteria

As we will demonstrate empirically in the next section, the original IGMM has a tendency to create a large number of components in the presence of concept drifts.

There are several possible approaches in the literature to deal with this problem, such as merge of similar components [6], [11], [12], the use of negative examples to eliminate concepts considered wrong and/or outdated [13] and removal information using criteria such as time or relevance.

In this paper we propose adding a third parameter T to the algorithm. This parameter is a limit of allowed components in the model. So when there is a very large number of components, those ones having the smallest *a priori* probabilities are eliminated. The maximum allowable amount is based on the parameter T and the number of existing classes in the model, i.e. the model allows, on average, T components representing each class. When the model is represented by more than $T \times numberOfClasses$ components, the algorithm eliminates the components with lowest *a priori* probability (w), since these components are represented by the smallest number of examples.

This change, although simple, allows to control the model growth and consequent the computational cost of the algorithm. On the other hand, it establishes a limit for model complexity and favors the perpetuation of more dense components.

V. Results and Discussions

We organize this section into two main parts. In the first part we compare IGMM-CD with the original IGMM. We

briefly show that IGMM leads to over-complex models in datasets with concept drifts. Unfortunately, the run times of IGMM are so large that we could make this comparisons just to the smallest datasets.

In the second part, we compare IGMM with the state-of-the-art algorithms in data stream learning using the MOA environment. We prepared a paper website [14] with detailed results, code and data for reproducing all our experiments. Our implementation is integrated with the MOA environment to facilitate the reproduction of our results.

We start this section describing the datasets used in the experiments.

A. Datasets

We performed our experiments using real and synthetic datasets. Table I shows some of the characteristics of the synthetic datasets used. These datasets were obtained from [15]. The author created videos showing the variation over time of these datasets [16]. The column "Change every X examples" indicates the number of examples between consecutive changes of concept.

TABLE I: Datasets

Dataset	Number of classes	Number of attributes	Number of examples	Change every X examples
1CDT	2	2	16,000	400
2CDT	2	2	16,000	400
1CHT	2	2	16,000	400
2CHT	2	2	16,000	400
4CR	4	2	144,400	400
4CRE-V1	4	2	125,000	1,000
4CRE-V2	4	2	183,000	1,000
5CVT	5	2	40,000	1,000
1CSurr	2	2	55,283	600
UG_2C_2D [17]	2	2	100,000	1,000
MG_2C_2D [17]	2	2	200,000	2,000
FG_2C_2D [18]	2	2	200,000	2,000
UG_2C_3D [17]	2	3	200,000	2,000
UG_2C_5D [17]	2	5	200,000	2,000
GEARS_2C_2D [17]	2	2	200,000	2,000

The synthetic datasets were created with Gaussian distributions that move in the feature space over time. Therefore, they simmulate chances in $p(x|c)$, by increasing or decreasing the values of the attributes over time. This gives origin to different drift patterns, some simple, such as a Gaussians that move away from each other with time, or more complex, such as Gaussians that rotate around each other. The videos created by the author provide a visual and intuitive description of each dataset.

The real datasets used in our experiments are the following:

- **Poker-lsn** It has 1,000,000 instances and 10 attributes. Each instance is an example of a poker hand containing five cards taken from an ordinary deck of 52 cards, each card is described by two attributes: suit and rank;
- **ElecNormNew** Dataset described by [19]. These data were collected from New South Wales electricity market in Australia. In this market, prices are not fixed and are affected by supply and demand of the market. They are adjusted every five minutes. The dataset contains 45312 instances. Each class identifies the price change relative to a moving average of the last 24 hours. The normalized version of the dataset was used, having 8 attributes and 2 classes.
- **Keystroke** a stream dataset of keystroke dynamics based on *CMU* data [20]. In *CMU* data, 51 users type the password "tie5Roanl" plus the *Enter* key 400 times captured in 8 sessions performed in different days. The dataset has 10 features extracted from the *flight time* for each pressed key. The *flight time* is the time difference between the instants when a key is released and the next key is pressed. In the stream dataset, [15] randomly chose 4 users and merged them respecting the chronological order in a total of 1,600 examples.

B. Comparison with IGMM

Initially, the incremental Gaussian Mixture Model [3] was implemented as shown in Section IV but without use of *remove outdated components()* method, keeping in the model all Gaussian components created. Basically this IGMM is just an adaptation for classification of the original that is designed for clustering. However, at the beginning of the experiments it was realized that such approach did not have an approximately constant computational cost over time, increasing as new data were presented to the algorithm and new Gaussian components were created. Such behavior is associated to the fact that there is no components elimination strategy over time. In presence of concept drifts the Gaussian components associated to data generated from obsolete concepts are represented in the model, increasing the computational cost and possibly reducing the accuracy of the model.

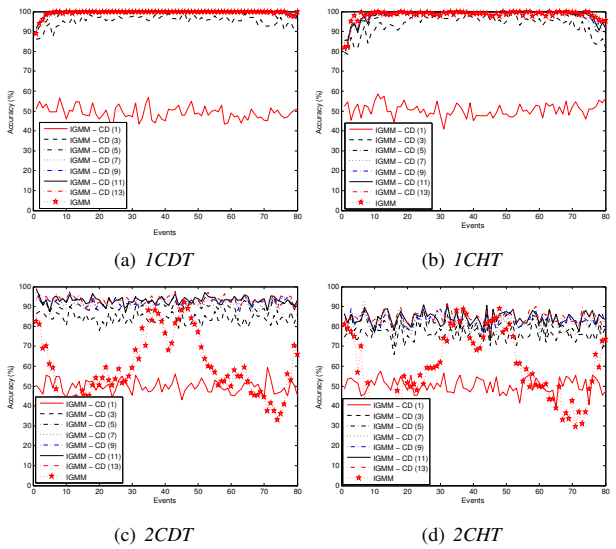
We start with a comparison to the original IGMM. Unfortunately, due to the high computational costs of IGMM in concept drift data, we were able to perform the experiments only on the smallest datasets. We use the global setting with IGMM since it better conforms with the original concept of mixture models.

Table II shows the total time to process the datasets used in the experiments, while Figure 1 shows the accuracy of the algorithms IGMM and IGMM-CD over time. The accuracy of each point represents the mean classification accuracy over a group of 200 examples. The lines for IGMM-CD also show the effect of varying the parameter *T*.

TABLE II: Accumulated time cost results (in seconds) for IGMM and IGMM-CD with local update. For IGMM-CD we also present the effect of varying the parameter *T*

	IGMM	IGMM-CD			
		T value			
Dataset		1	5	9	13
1CDT	26.42	0.21	0.76	1.26	1.74
2CDT	886.60	0.21	0.76	1.21	1.71
1CHT	69.55	0.21	0.76	1.21	1.73
2CHT	1543	0.20	0.78	1.23	1.65

The results show that IGMM can take up to approximately half-hour to process a simple dataset with only 16,000 instances. The use of the T parameter reduces the processing



(a) *1CDT* (b) *1CHT*

(c) *2CDT* (d) *2CHT*

Fig. 1: Accuracy variation over time for IGMM (Without Elimination) and IGMM-CD with local update and T parameter variation in datasets (a) *1CDT*, (b) *1CHT*, (c) *2CDT* and (d) *2CHT*

time to around one second. Such speed-up does not lead to classification accuracy degradation (see Figure 1). We can see that IGMM-CD has similar or better performance than IGMM for all but the smallest values of T.

Due to the high computational demand of IGMM, we will not include this method in the next experiments. We also perform the remainder experiments with $T = 7$ since this parameter seems to be able to significantly decrease processing time without affecting classification performance.

C. Comparison with the State-of-the-art

The state-of-the-art algorithms used in our experiments are Naive Bayes, Hoeffling Adaptive Tree [7]–[9], Active Classifier [21], [22], Drift Detection Method, Perceptron and Accuracy Updated Ensemble [23]. The choice of these algorithms is based both on the availability of MOA environment as in a search for different approaches, since the list includes naturally incremental algorithms, such as Naive Bayes, methods based on ensemble, as the Accuracy Updated Ensemble, based on detecting concept drifts as Drift Detection Method, decision trees, such as Hoeffling Tree Adaptive, and others. Table III shows the comparative results between the algorithms.

The results show that IGMM-CD can perform similarly to the state-of-the-art. Note that IGMM-CD is a single classifier with no drift detection mechanism, but performs much better than Naive Bayes that has similar characteristics. In contrast, IGMM-CD sometimes perform poorly in the global or local setting alternatively. This seems to indicate that neither approach is the best for data streams and a midterm should be a better solution. We will further analyze this in future research.

It is worth noting that a study on the effects of varying the parameter T is relevant, since such variation can improve the performance of the algorithm. Table IV shows some examples where the variation of the parameter T provided an improvement in IGMM-CD accuracy.

TABLE IV: Accuracy - mean (standard deviation) over the full flow - for IGMM-CD with global update (G) and local update (L) with T parameter variation

Dataset	**IGMM-CD (G)**	T	**IGMM-CD (L)**	T
ElecNormNew	85.32 (4.25)	1	85.32 (4.25)	1
FG_2C_2D	93.54 (3.78)	13	91.33 (3.69)	13
GEARS_2C_2D	91.58 (4.48)	13	86.33 (4.66)	13
1CSurr	96.04 (4.08)	11	95.79 (1.80)	13
Keystroke	77.56 (10.41)	13	39.06 (6.66)	7
MG_2C_2D	90.58 (8.31)	13	88.7 (8.26)	13
poker_lsn	74.19 (8.04)	1	73.91 (8.27)	9
4CRE-V1	95.78 (7.95)	3	96.65 (5.64)	13
4CRE-V2	89.54 (9.73)	5	88.38 (9.57)	13
4CR	99.89 (0.26)	3	99.89 (0.24)	13
1CDT	99.74 (0.90)	7	99.65 (1.17)	13
2CDT	92.42 (3.29)	5	93.37 (2.09)	13
1CHT	98.88 (2.57)	7	98.76 (2.78)	13
2CHT	82.77 (5.92)	5	84.64 (3.07)	13
5CVT	67.00 (7.98)	1	88.44 (3.37)	13
UG_2C_2D	94.40 (4.28)	11	92.61 (4.83)	13
UG_2C_3D	92.65 (7.63)	13	90.99 (8.05)	13
UG_2C_5D	81.20 (12.41)	3	87.41 (8.07)	13

These results are interesting due to the fact that by varying the (T) parameter it was possible to obtain good results for all datasets using IGMM-CD with both global and local updates. Furthermore, T allows to control the computational cost of processing new examples because the number of components per class is directly proportional to the processing time to update the model and to classify a new event. We believe this parameter is the most relevant to the algorithm performance, having great influence on its performance.

VI. Conclusion

In this paper we proposed a novel Gaussian mixture model classification algorithm for data streams with concept drifts. Our algorithms differ from previous proposal by having an explit policy to remove outdated components.

We showed that IGMM-CD can provide accurate results in comparison to the state-of-the-art. However, it is highly dependent of the parameter T, which controls the number of components per class. This parameter will be further analysed in future research. Our method would be benefited by a simple heuristic that could allow us to adjust this parameter for different datasets, or an adaptive approach that would adjust T during the data stream.

In terms of efficiency, IGMM-CD is significantly faster than the original incremental algorithm since it controls the maximum number of components. However, we can further improve the efficiency of our method by exploring simple ideas such as the replacement of full covariance matrix by a diagonal covariance matrix. This change would result in a great savings of processing time, since the calculation of a matrix inverse requires costly algorithms, typically $O(n^3)$. This strategy has already been used in the literature, resulting in better efficiency without compromising efficacy [24]. However, we still need to evaluate the impact of such a change in the algorithm accuracy.

TABLE III: Accuracy - mean (standard deviation) over the entire data stream - for IGMM-CD with global update (G) and local update (L), *active learning*, *drift detection*, *tree*, *naive bayes*, *perceptron* and *ensemble*

Dataset	IGMM-CD (G)	IGMM-CD (L)	Active	Drift Det.	Tree	N. Bayes	Perceptron	Ensemble
ElecNormNew	79.12 (7.68)	57.28 (11.07)	75.49 (11.67)	81.19 (9.7)	**83.79 (8.29)**	73.20 (12.09)	79.11 (5.90)	77.30 (11.42)
FG_2C_2D	90.75 (5.85)	88.64 (4.05)	92.64 (4.23)	91.28 (5.06)	95.07 (2.66)	84.97 (10.84)	75.00 (2.89)	**95.39 (2.32)**
GEARS_2C_2D	89.53 (5.32)	85.18 (3.70)	95.83 (1.36)	95.82 (1.37)	97.81 (1.42)	95.82 (1.37)	96.00 (1.38)	**98.99 (1.12)**
1CSurr	95.39 (3.78)	93.30 (2.49)	95.82 (3.73)	**97.65 (2.01)**	96.85 (2.61)	65.49 (16.02)	66.44 (9.08)	96.72 (3.56)
Keystroke	58.75 (17.24)	39.06 (6.66)	81.69 (5.59)	73.88 (7.93)	83.25 (7.36)	63.56 (11.03)	**85.94 (3.83)**	75.94 (13.84)
MG_2C_2D	87.94 (10.24)	86.61 (8.97)	88.69 (8.05)	88.36 (8.59)	92.69 (6.05)	55.45 (33.37)	47.73 (11.89)	**93.19 (5.85)**
poker_lsn	73.01 (7.87)	**73.24 (9.49)**	60.93 (19.24)	62.01 (19.01)	66.90 (16.30)	59.48 (19.60)	0.39 (1.32)	66.82 (18.12)
4CRE-V1	27.03 (38.85)	95.80 (6.49)	94.73 (8.73)	97.10 (4.73)	77.39 (32.81)	22.20 (37.40)	**98.18 (4.68)**	95.16 (9.05)
4CRE-V2	41.67 (34.38)	87.6 (9.43)	83.81 (11.13)	88.08 (8.45)	88.96 (8.90)	24.17 (31.82)	**92.19 (7.83)**	91.70 (8.24)
4CR	29.23 (38.01)	99.80 (0.32)	99.41 (2.79)	99.87 (0.51)	99.57 (0.83)	24.84 (38.31)	98.93 (5.58)	**99.94 (1.36)**
1CDT	99.74 (0.90)	99.46 (1.51)	99.60 (0.97)	99.65 (0.89)	99.65 (0.86)	99.65 (0.89)	**99.85 (0.64)**	99.41 (2.89)
2CDT	59.09 (17.75)	91.84 (2.15)	90.28 (6.29)	**94.11 (3.06)**	85.85 (10.85)	59.56 (16.14)	55.25 (13.23)	86.17 (6.34)
1CHT	98.88 (2.57)	98.24 (3.33)	98.47 (2.07)	98.58 (2.02)	98.49 (2.33)	98.57 (2.05)	**99.20 (2.02)**	98.81 (3.74)
2CHT	81.32 (7.27)	82.74 (3.51)	77.42 (9.95)	**86.36 (4.68)**	84.21 (5.72)	59.46 (13.88)	57.56 (12.55)	78.72 (5.98)
5CVT	44.92 (28.77)	87.87 (2.93)	83.93 (8.64)	**89.63 (4.98)**	87.36 (5.37)	66.05 (15.27)	64.42 (12.10)	86.62 (5.55)
UG_2C_2D	93.14 (5.55)	90.94 (5.10)	94.22 (5.41)	94.82 (4.21)	95.61 (3.80)	58.09 (30.81)	75.43 (20.20)	**95.89 (3.84)**
UG_2C_3D	91.03 (9.26)	89.44 (8.50)	93.92 (5.85)	94.39 (5.60)	94.58 (5.53)	61.38 (28.70)	91.89 (7.54)	**94.94 (5.47)**
UG_2C_5D	68.88 (15.72)	84.44 (8.46)	91.67 (6.37)	92.86 (5.42)	92.78 (5.57)	79.17 (11.45)	89.21 (8.30)	**93.42 (5.34)**

We note that a preset parameter T is a very simple approach to component removal. Other possibilities are removal by component age (older components are removed first) and classification use (components little used for classification are removed first), etc. We could also use more sophisticated approaches such as component fusion, which would allow to join two or more smaller components together instead of removing them. As future work we also intend to evaluate these schemes to eliminate Gaussian components and provide a empirical comparison among them.

REFERENCES

[1] I. Zliobaite, "Learning under concept drift: an overview," Faculty of Mathematics and Informatics of Vilnius University, Lithuania, Tech. Rep., 2009.

[2] M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *ACM Sigmod Record*, vol. 34, no. 2, pp. 18–26, 2005.

[3] M. R. Engel, Paulo Martins e Heinen, "Incremental learning of multivariate gaussian mixture models," in *Proceedings of the 20th Brazilian Conference on Advances in Artificial Intelligence*, ser. SBIA'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 82–91.

[4] J. R. B. Júnior, "Classificação de dados estacionários e não estacionários baseada em grafos," Ph.D. Thesis, Instituto de Ciências Matemáticas e de Computacão, Universidade de São Paulo, 2010.

[5] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, oct. 2011.

[6] A. Bouchachia and C. Vanaret, "Incremental learning based on growing gaussian mixture models," in *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, vol. 2. IEEE, 2011, pp. 47–52.

[7] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 97–106.

[8] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *Advances in Intelligent Data Analysis VIII*. Springer, 2009, pp. 249–260.

[9] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Data stream mining: A practical approach," The University of Waikato, Tech. Rep., May 2011.

[10] D. Reynolds, "Gaussian mixture model," *Encyclopedia of Biometrics*, pp. 659–663, July 2009.

[11] M. Song and H. Wang, "Highly efficient incremental estimation of gaussian mixture models for online data stream clustering," in *Defense and Security*. International Society for Optics and Photonics, 2005, pp. 174–183.

[12] O. Arandjelovic and R. Cipolla, "Incremental learning of temporally-coherent gaussian mixture models," *Society of Manufacturing Engineers (SME) Technical Papers*, pp. 1–1, 2006.

[13] M. Kristan, D. Skočaj, and A. Leonardis, "Online kernel density estimation for interactive learning," *Image and Vision Computing*, vol. 28, no. 7, pp. 1106–1116, 2010.

[14] L. S. Oliveira, "IGMM-CD: A Gaussian Mixture Classification Algorithm for Data Streams with Concept Drifts," https://sites.google.com/site/igmmcd9/home, 2015, [Online; accessed on 24-May-2015].

[15] V. Souza, D. Silva, J. Gama, and G. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," *SIAM International Conference on Data Mining (SDM)*, pp. 873–881, 2015.

[16] V. M. A. Souza, "Nonstationary Environments - Archive," https://sites.google.com/site/nonstationaryarchive/home, 2015, [Online; accessed on 24-Abril-2015].

[17] K. B. Dyer, R. Capo, and R. Polikar, "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 1, pp. 12–26, 2014.

[18] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 10, pp. 2283–2301, 2013.

[19] M. Harries and N. Wales, "Splice-2 comparative evaluation: Electricity pricing," The University of South Wales, Tech. Rep., 1999.

[20] K. Killourhy and R. Maxion, "Why did my detector do that?!" in *Recent Advances in Intrusion Detection*. Springer, 2010, pp. 256–276.

[21] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with evolving streaming data," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 597–612.

[22] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Worst-case analysis of selective sampling for linear classification," *The Journal of Machine Learning Research*, vol. 7, pp. 1205–1230, 2006.

[23] D. Brzeziński and J. Stefanowski, "Accuracy updated ensemble for data streams with concept drift," in *Hybrid Artificial Intelligent Systems*. Springer, 2011, pp. 155–163.

[24] F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-García, D. Petrovska-Delacrétaz, and D. A. Reynolds, "A tutorial on text-independent speaker verification," *EURASIP journal on applied signal processing*, vol. 2004, pp. 430–451, 2004.