2011 10th International Conference on Machine Learning and Applications

Incremental Learning based on Growing Gaussian Mixture Models

Abdelhamid Bouchachia
University of Klagenfurt,Dept. of Informatics,
Klagenfurt, Austria
Email: hamid@isys.uni-klu.ac.at

Charlie Vanaret
ENSEEIHT engineering school,
Toulouse, France.
Email: charlie.vanaret@edu.uni-klu.ac.at

***Abstract*—Incremental learning aims at equipping data-driven systems with self-monitoring and self-adaptation mechanisms to accommodate new data in an online setting. The resulting model underlying the system can be adjusted whenever data become available. The present paper proposes a new incremental learning algorithm, called $2G2M$, to learn Growing Gaussian Mixture Models. The algorithm is furnished with abilities (1) to accommodate data online, (2) to maintain low complexity of the model, and (3) to reconcile labeled and unlabeled data. To discuss the efficiency of the proposed incremental learning algorithm, an empirical evaluation is provided.**

I. Introduction

Recently incremental learning (IL) under different naming (e.g., online, evolving, constructive, etc.) has attracted much attention from the machine learning, pattern recognition, data mining and the fuzzy logic communities. There have been attempts to use various architectures and various paradigms to construct IL models. For an overview, we refer the reader to [1], [2]. Despite this growing interest, IL as a research field is still in its infancy.

In this paper, we focus on a particular instance of IL, that is incremental classification (IC). In particular, the goal is to investigate a classification system that is *semi-supervised* and *incremental* at the same time. Why semi-supervised? Since in the context of IL, data arrive over time, we expect that the new data might not be necessarily labeled. This is very natural in self-monitoring systems where the feedback from the environment is scarce. The feedback here comes in the form of labeled data. Unlabeled data is fed to the system more frequently, due to its low cost.

Specifically the present paper suggests an incremental version of Gaussian mixture models that is equipped growing and shrinking mechanisms allowing to control the complexity and the accuracy of the model. Moreover, the approach is semi-supervised, since both labeled and unlabeled data can be online accommodated.

It is very important to note that GMM are appealing because they are flexible for modeling complex and nonlinear pattern variations, efficient in terms of memory, capable of model selection, and theoretically guaranteed to converge algorithms for model parameter estimation [3], [4], [5]; hence the versatility of the presented incremental algorithm.

The present work has been developed when the second author was spending his internship at the University of Klagenfurt, Austria

In the sequel, an overview of incremental GMMs are introduced in Sec. II. Section III describes the proposed algorithm. Then, an empirical evaluation based on synthetic as well as real-world data is presented in Sec. IV before a summary is highlighted in Sec. V.

II. Overview of Incremental GMM's

GMM's are considered as a model-based clustering method. In essence, such a clustering method perceives the data as a population with K different components, where each component is generated by an underlying probability distribution [4]. The density of an $D-$dimensional data point x_i from the j^{th} component is $f_j(x_i;\theta_j)$, where θ_j represents a vector of some unknown parameters associated with the component j. The density function associated is expressed as:

$$f(x_i|\theta) = \sum_{j=1}^{K} \tau_j f_j(x_i;\theta_j) \tag{1}$$

where τ_j is the weight of the j^{th} component such that $\sum_{j=1}^{K} \tau_j = 1$.

A typical case is when $f_j(x_i;\theta_j)$ is multivariate normal (Gaussian) density ϕ_j with θ_j representing the characteristics of each component j. These characteristics are the mean μ_j and the covariance matrix Σ_j. The density ϕ_j is then described as follows:

$$\phi_j(x_i|\mu_j,\Sigma_j) = \frac{exp\{-1/2(x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)\}}{\sqrt{det(2\pi\Sigma_j)}} \tag{2}$$

Gaussian mixtures are usually trained using the iterative Expectation Maximization (EM) algorithm. In the E-step, the posterior probability of data points is computed, while in the M-step, the parameters of the mixture $(\hat{\tau}_j,\hat{\mu}_j,\hat{\Sigma}_j)$ are computed. Such an iterative algorithm is not recommended in the context of this study since the aim is to avoid storage of the entire data (e.g. a continuous stream of data). Each data point is discarded once it has been processed. Therefore, only incremental variants of GMM can be applied. The existing approaches can be split into two categories, we name them here as: refinement-based methods and learning methods.

1) **Refinement-based methods**: In the first category, the methods rely generally on two abstract stages[1]: (i) Processing and integration of the new data into the current model, (ii) Refinement of the resulting model. In the later stage, the goal is to reduce the complexity of the model resulting from the former stage. Usually this refinement is realized by means of a merge and/or split operations that combine similar components of GMM. As mentioned in the previous section, many merge criteria have been used, e.g., the W-statistic for equality of covariance and Hotellings T^2-statistic for equality of mean [6], fidelity [7], weighting [8], Chernoff bound [8], Kullback-Leibler measure [9], mutual information [10], minimum description length [3].
2) **Learning methods**: In this category of methods, the idea is to apply an online variant of the EM algorithm [11] to train in an evolutionary and continual way the Gaussian mixtures as new data become available. Based on this idea, there have several proposals dedicated in particular to surveillance system and video analysis. Originally, the work by Stauffer and Grimson [12] has become the standard formulation for the online learning of Gaussian mixtures which adopts an online EM-approximation based on recursive filter and which is similar to online clustering methods. It relies basically on a learning rule having a standard scheme:

$$\theta(t) = (1-\eta(t))\theta(t-1) + \eta(t) \bigtriangledown (x(t); \theta(t-1)) \quad (3)$$

where t, θ, η and $\bigtriangledown(.)$ indicate respectively time, a model parameter (mean, variance), learning rate, and update amount that depends on the previous parameter value and the new data point. An analysis of the learning rate has been conducted in [13] showing in particular that when $\eta(t) = 1/t$, the algorithm will run into local optimum on a stationary data distribution and if $\eta(t) = \alpha$ (α is very small value $<< 1$), the most recent observations (belonging to a window of $L = 1/\alpha$ and guided by an exponential decay) will have more impact and forces the algorithm to take longer before it converges. The author in [13] proposes a solution that ensures temporal adaptability and fast convergence and letting η to converge to α instead of 0. More interestingly each Gaussian will be equipped with its own η computed based on the current likelihood estimates. This allows to handle Gaussians independently. The algorithm proposed by Lee in [13] differs from the original one by Stauffer and Grimson [14] with respect to the update rules of the learning rate η_j and the weight of the Gaussian components τ_j.

III. 2G2M: A Detailed Presentation

The algorithm shown in Alg. 1 extends the algorithm presented in [13] in many aspects. The novelties concern the processing of the labeling information, the explicit handling of multi-dimensional data, handling the complexity of the model generated as will be explained later, and finally the re-structuration to accommodate the named changes. The similarities concern mainly the online update procedure which is based on an online approximation of the standard expectation maximization method provided in [12].

[1]These two stages might be split into three as is the case in some proposals

The symbols appearing in Alg. 1 are defined as follows:

K	... the number of Gaussians $\phi_j,\ j = 1 \cdots K$
μ_j	... the center of the jth Gaussian
Σ_0	... the initial covariance matrix of the Gaussians ($= k_0 I_D$ proportional to the identity matrix of size D (number of features)
α	... the learning rate
T_Σ	... the closeness threshold (Eq. 4)
τ_j	... the weights of the jth Gaussian in the mixture model
c_j	... the sum of the expected posterior (a sufficient statistics in the expectation maximization method) of the jth Gaussian
L_j	... the label of the jth Gaussian(cluster)

Moreover, the algorithm is characterized by the following aspects:

A. Probability of Match

Eq. 4 has to be adapted to take the semi-supervised learning into account. We compute the probability of match of the input x_i with the jth Gaussian by considering the following cases:

- The input x_i is unlabeled
- The input x_i is labeled and jth Gaussian is unlabeled
- The input x_i and jth Gaussian have the same label

The Mahalanobis distance between an input x_i and a Gaussian $\phi_j(\mu_j, \Sigma_j)$ in Eq. 4 reads:

$$d_M(x_i, \phi_j) = \sqrt{(x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j)} \quad (15)$$

B. Processing Unlabeled Data

The $2G2M$ algorithm has to be adapted so that unlabeled data can be accommodated given the semi-supervised learning setting. The Gaussians are updated or created according to the labels of both the Gaussian and the incoming data sample.

- if the incoming data sample is not labeled, simply find the highest matching Gaussian, add the contribution of the data sample and update all of the Gaussians
- if the incoming sample is labeled, then
  - if the highest matching Gaussian is unlabeled, label it with the label of the sample
  - if the highest matching Gaussian and the incoming sample have different labels, then create a new Gaussian
  - if the highest matching Gaussian and the incoming sample have the same label, add the contribution of the data and update all of the Gaussians

Algorithm 1 : Steps of 2*G2M*

```
1: Set parameters: $K$, $\Sigma_0$, $\alpha$, $T_\Sigma$
2: Initialization:
   $\forall j = 1 \cdots K, (\tau,\ c,\ \mu,\ \Sigma,\ L)_j = (\alpha,\ 1,\ \infty,\ \Sigma_0,\ 0)$
3: Given a new input $x_i$, compute the probability of match of the input with each Gaussian: $\forall j = 1 \cdots K$
   $$p_j = \begin{cases} \tau_j \phi_j(x_i;\mu_j,\Sigma_j) & \text{if } d_M(x_i,\phi_j) < T_\Sigma \\ 0 & \text{otherwise} \end{cases} \quad (4)$$
4: Let $R = \{j | p_j > 0\}$ be the index set of Gaussians matching the input
5: createGaussian $\leftarrow$ ($R == \emptyset$) (i.e., no match is found: $\nexists j, p_j > 0$)
6: if not createGaussian then
7:     Compute the index of highly matching Gaussian:
       $$w = \underset{j=1\cdots K}{\text{argmax}}\{p_j\} \quad (5)$$
8:     if $x_i$ is labeled then
9:         if The $w$th Gaussian is not labeled then
10:            Assign the $w$th Gaussian the label of $x_i$
11:        else if The $w$th Gaussian and $x_i$ have different labels then
12:            Set createGaussian $\leftarrow$ $true$
13:        end if
14:    end if
15:    if not createGaussian then
16:        for $j = 1 \cdots K$ do
17:            Compute the expected posterior:
               $$q_j = \frac{p_j}{\sum_{k=1\cdots K} p_k} \quad (6)$$
18:            Update the parameters of the Gaussian:
               $$c_j = c_j + q_j \quad (7)$$
               $$\tau_j(t) = (1-\alpha)\tau_j(t-1) + \alpha q_j \quad (8)$$
               $$\eta_j = q_j\left(\frac{1-\alpha}{c_j} + \alpha\right) \quad (9)$$
               $$\mu_j(t) = (1-\eta_j)\mu_j(t-1) + \eta_j x_i \quad (10)$$
               $$\Sigma_j(t) = (1-\eta_j)\Sigma_j(t-1) + \eta_j(x_i - \mu_j(t-1))^2 \quad (11)$$
19:        end for
20:    end if
21: end if
22: if createGaussian then
23:    Decay the weight of all Gaussians
       $$\forall j = 1 \cdots K, \quad \tau_j(t) = (1-\alpha)\tau_j(t-1) \quad (12)$$
24:    Remove the less contributing Gaussian and create a new one initialized with the new input:
       $$m = \underset{j}{\text{argmin}}\{\tau_j\} \quad (13)$$
       $$(\tau, c, \mu, \Sigma, L)_m = (\alpha, 1, x_i, \Sigma_0, 0) \quad (14)$$
25: end if
26: Split largest Gaussian if volume $> T_{split}$
27: Merge closest Gaussians if KL distance $< T_{merge}$
28: Normalize the $\tau_j$'s
```

C. Refinement of the Model

To lower the complexity and enhance the accuracy of the model, we rely on the idea underlying refinement-based methods that use two operations: *split* and *merge* as explained in Sec. II.

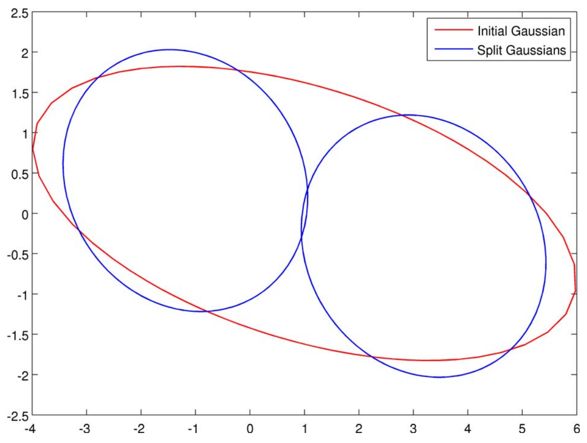


Fig. 1: Split of a Gaussian along the dominant principal component (-0.98, 0.18)

1) Split Operation: In order to keep the algorithm accurate and precise, large Gaussians (considered to be as "too important") are split into two smaller Gaussians. The chosen criterion for deciding the split of a Gaussian $\phi(\mu,\Sigma)$ refers to its volume:

$$V(\phi(\mu,\Sigma)) = det(\Sigma) \quad (16)$$

The Gaussian $\phi(\tau,\mu,\Sigma)$ whose V is maximal and above a given threshold T_{split} is split into two Gaussians ($\phi_1(\tau_1,\mu_1,\Sigma_1)$, $\phi_2(\tau_2,\mu_2,\Sigma_2)$) along its dominant principal component (Fig. 1). Let v the (normalized) eigenvector corresponding to the largest eigenvalue λ, and a a split factor that determines how far the centers of the two resulting Gaussians are apart from each other. Note that a is set to 0.8 in this paper.

$$\begin{cases} \Delta v = \sqrt{a\lambda}v \\ \tau_1 = \tau_2 = \frac{\tau}{2} \\ \mu_1 = \mu + \Delta v \\ \mu_2 = \mu - \Delta v \\ \Sigma_1 = \Sigma_2 = \Sigma - \Delta v \Delta v^T \end{cases} \quad (17)$$

2) Merge Operation: In order to lower the complexity, Gaussians of the same label can be merged. The merge operator combines the two closest Gaussians into one single Gaussian. It relies on the Kullback-Leibler divergence (*kld*) between two D-dimensional Gaussians ($\phi_1(\mu_1,\Sigma_1)$, $\phi_2(\mu_2,\Sigma_2)$) as a merge criterion, which reads as follows[15]:

$$kld(\phi_1,\phi_2) = log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) + tr(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^T\Sigma_1^{-1}(\mu_2-\mu_1) - D \quad (18)$$

where tr indicates the trace. Since *kld* is asymmetric, a symmetrized variant (*skld*) of the divergence is used for similarity estimation:

$$skld(\phi_1,\phi_2) = \frac{1}{2}\left(kld(\phi_1,\phi_2) + kld(\phi_2,\phi_1)\right) \quad (19)$$

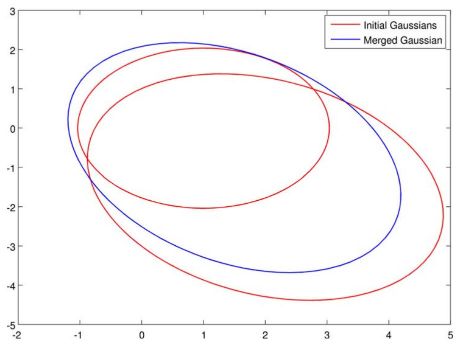


Fig. 2: Merge of Gaussians

The pair of Gaussians ($\phi_1(\tau_1, \mu_1, \Sigma_1)$, $\phi_2(\tau_2, \mu_2, \Sigma_2)$) whose *skld* is minimal and falling under a given threshold T_{merge} is combined into a new Gaussian $\phi(\tau, \mu, \Sigma)$ (Fig. 2) according to the following equations:

$$\begin{cases} f_1 = \frac{\tau_1}{\tau_1+\tau_2} \\ f_2 = \frac{\tau_2}{\tau_1+\tau_2} \\ \tau = \tau_1 + \tau_2 \\ \mu = f_1\mu_1 + f_2\mu_2 \\ \Sigma = f_1\Sigma_1 + f_2\Sigma_2 + f_1 f_2(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \end{cases} \tag{20}$$

D. *Cannot-Merge Link*

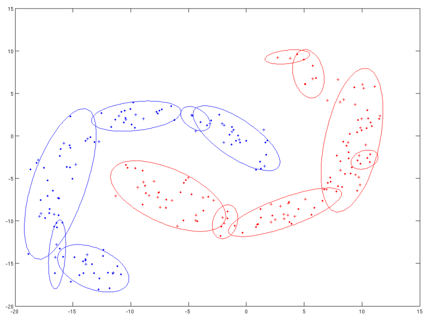
Due to the very nature of the split and merge operations, there is a risk that a Gaussian be split, then re-merged straight away. To avoid this phenomenon, we endow each split or merged Gaussian with a **cannot-merge link** (CML) similar to the idea of must-link and cannot-link in constraint-based clustering. CML is simply a positive integer that decreases over time and that disables temporarily subsequent split or merge operations on the same Gaussians as long as it has not reached 0. In the experiments discussed here it is empirically set to 50.

Fig. 3b shows the results of two split operations occurring in one cluster of each class of a banana shaped dataset. The accommodation of new data samples have modified the shapes and orientations of the two blue resulting clusters (Fig. 3c) and have not been subsequently merged. For the red class, the two clusters resulting from the split have not been significantly modified, but have been re-merged once the cannot-merge link became inactive (Fig. 3c).

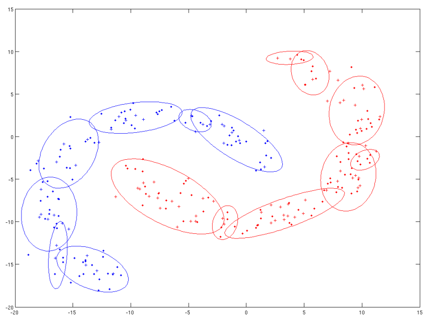
It is important to note that the *virtual* ellipses show approximately the region of influence of each Gaussian around a corresponding center, since the $2G2M$ algorithm delivers as output only the means, the covariance matrices and the mixing parameters.

IV. Numerical Simulations

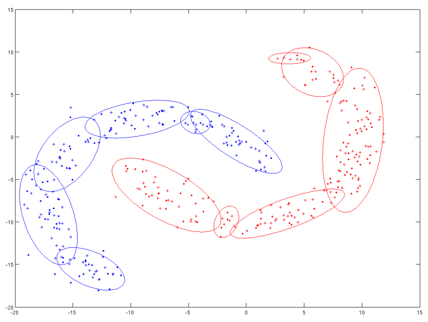
In this section, we present two sets of experiments. For the sake of illustration, the first set of experiments are related to the banana shaped data. In the second set of experiments, we use a real-world dataset pertaining to human activity recognition in the context of smart homes.



(a) Step before two split operations (in both classes)



(b) Step after two split operations (in both classes)



(c) Final clustering results

Fig. 3: Banana shaped dataset.

We will check several aspects to give insight into the approach. In particular, we study the behavior of the online adaptation which is measured by Eq. 21 that expresses the current error rate defined over time as:

$$e(t) = \frac{misses(t)}{seen\_so\_far(t)} \tag{21}$$

where $misses(t)$ represents the number of misclassified data until time t and $seen\_so\_far(t)$ represents the number of presentations (inserted inputs) seen so far. Note that in this

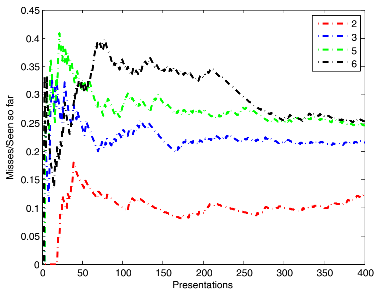


Fig. 4: Effect of the maximum number of Gaussians

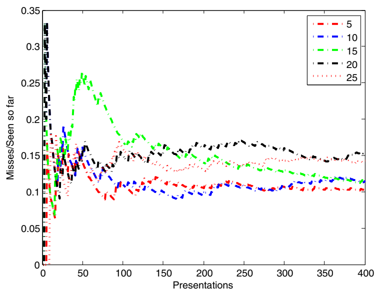


Fig. 5: Effect of the closeness threshold

paper the ratio of labeled data is set to 30% of the whole data randomly selected and presented. Due to space limitation the semi-supervision aspect has not been studied and will be in the future.

A. Synthetic Data: Banana

Using the banana dataset provides an insight into the behavior of the algorithm 2*G*2*M*. The effect of all parameters involved in the algorithm are observed.

Fig. 4 indicates that the ideal maximum number of clusters would be 2 which corresponds to the ground truth. Low values of the closeness threshold T_Σ (Eq. 4) is favored as shown in Fig. 5. Moreover, the effect of the merge threshold and the split threshold seem not to affect the accuracy in the long run (Fig. 6 and Fig. 7), while for the learning rate, larger values are better than low values (Fig. 8).

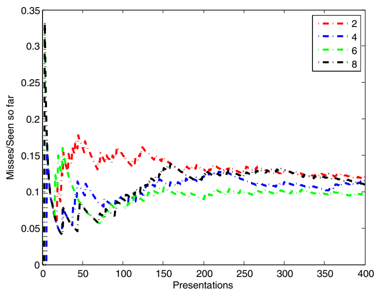


Fig. 6: Effect of the merge threshold

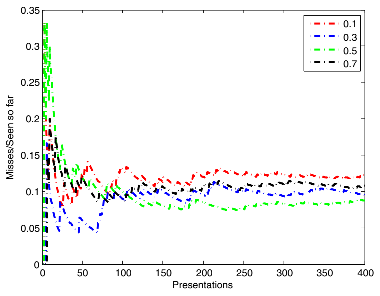


Fig. 7: Effect of the split threshold

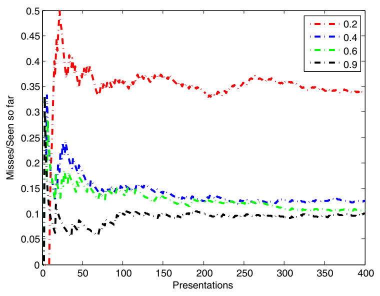


Fig. 8: Effect of the learning rate

B. Real-World Data: Human Activity Recognition

This dataset consists of 4 variables corresponding to sensors' input: hour, day, month and duration of the human activity in a smart home [16]. The output (class) corresponds to the activity. The dataset is highly imbalanced as one can notice in Tab. I.

Here again the effect of each of the parameters involved in 2*G*2*M* is observed. First, the maximum number of Gaussians corresponds to the true one as shown in Fig. 9. Concerning the closeness threshold, low values seem to produce better results (Fig. 10), whereas the merge threshold, the split threshold and the learning rate seem to have no significant effect as shown in Figs. 11, 12, and 13.

V. Conclusion

In this paper, we presented an incremental learning algorithm for estimating the generalized mixture models, called 2*G*2*M* and that stand for "Growing Generalized Mixture Models". The algorithm allows also to refine the models generated by using merge and split operations. The experimental

TABLE I: Human actions detected by sensors

Class	Action	Occurrence
1	leave house	33
2	use toilet	114
3	take a shower	23
4	go to bed	24
5	prepare breakfast	20
6	get a drink	20

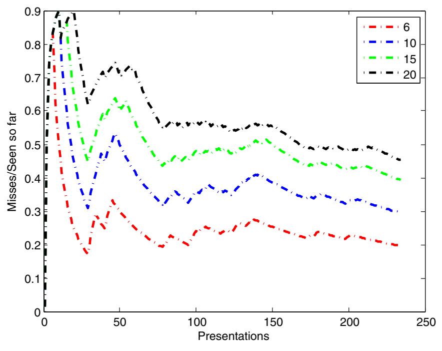


Fig. 9: Effect of the maximum number of Gaussians

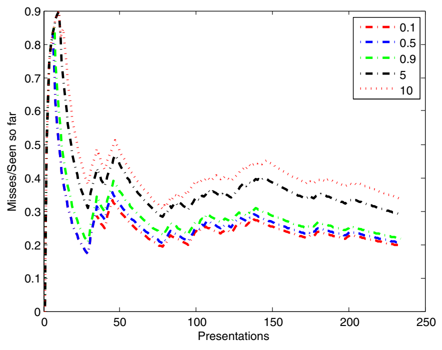


Fig. 10: Effect of the closeness threshold

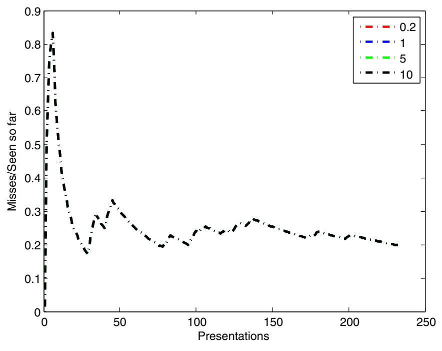


Fig. 11: Effect of the merge threshold

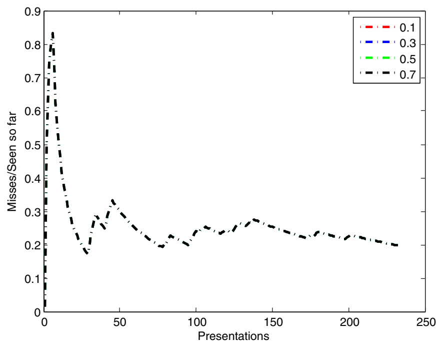


Fig. 12: Effect of the split threshold

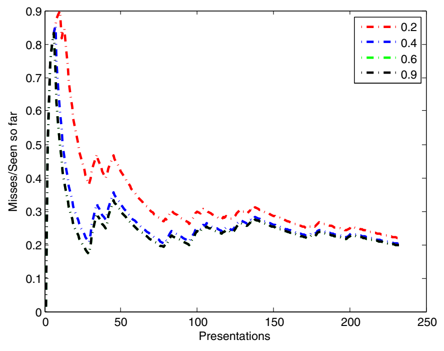


Fig. 13: Effect of the learning rate

evaluation showed a high accuracy of the algorithm. However, a thorough evaluation is still required.

References

[1] A. Bouchachia, *Encyclopedia of Data Warehousing and Mining*, 2nd ed. Idea-Group, 2008, ch. Incremental Learning, pp. 1006–1012.

[2] ——, "Incremental induction of fuzzy classification rules," in *IEEE Workshop on Evolving and Self-Developing Intelligent Systems*, 2009, pp. 32–39.

[3] O. Arandjelovic and R. Cipolla, "Incremental learning of temporally coherent Gaussian mixture models," in *Proceedings of the 16th British Machine Vision Conference*, 2005, pp. 759–768.

[4] J. Banfield and A. Raftery, "Model-based Gaussian and non-Gaussian Clustering," *Biometrics*, vol. 49, pp. 803–821, 1993.

[5] R. Gross, J. Yang, and A. Waibel, "Growing Gaussian mixture models for pose invariant face recognition," in *Proceedings of the International Conference on Pattern Recognition*. IEEE Computer Society, 2000, pp. 1088–1091.

[6] M. Song and H. Wang, "Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering," in *Intelligent Computing: Theory and Applications*. SPIE, 2005, pp. 174–183.

[7] A. Declercq and J. Piater, "Online learning of Gaussian mixture models - a two-level approach," in *Proceedings of the 3rd International Conference on Computer Vision Theory and Applications*, 2008, pp. 605–611.

[8] P. Hall and Y. Hicks, "A method to add Gaussian mixture models," University of Bath, Tech. Rep., 2004.

[9] S. Roberts, C. Holmes, and D. Denison, "Minimum-entropy data partitioning using reversible jump markov chain monte carlo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 909–914, 2001.

[10] Z. Yang and M. Zwolinski, "Mutual information theory for adaptive mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 4, pp. 396–403, 2001.

[11] N. Radford and G. Hinton, *Learning in graphical models*. MIT Press, 1999, ch. A view of the EM algorithm that justifies incremental, sparse, and other variants, pp. 355–368.

[12] C. Stauffer and W. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, 2000.

[13] D. Lee, "Effective Gaussian mixture learning for video background subtraction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 827–832, 2005.

[14] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 2, pp. 246–252, 1999.

[15] D. Schnitzer, A. Flexer, G. Widmer, and M. Gasser, "Islands of Gaussians: The self organizing map and Gaussian music similarity features," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR10)*, 2010.

[16] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kroese, "Accurate activity recognition in a home setting," in *Proceedings of the 10th International Conference on Ubiquitous Computing*, 2008.