Fuzzy integral to speed up support vector machines training for pattern classification

Hassiba Nemmour* and Youcef Chibani
Signal Processing Laboratory, Faculty of Electronic and Computer Sciences, University of Sciences and Technology Houari Boumediene, EL-Alia B. P. 32, 16111, Algiers, Algeria

Abstract. The major drawback of Support Vector Machines (SVMs) consists of the training time, which is at least quadratic to the number of data. Among the multitude of approaches developed to alleviate this limitation, several research works showed that mixtures of experts can drastically reduce the runtime of SVMs. The mixture employs a set of SVMs each of which is trained on a sub-set of the original dataset while the final decision is evaluated throughout a gater. The present work proposes a new support vector mixture in which Sugeno’s fuzzy integral is used as a gater to remove the time complexity induced by conventional gaters such as artificial neural networks. Experiments conducted on standard datasets of optical character and face recognition reveal that the proposed approach gives a significant reduction of the runtime while keeping at least the same accuracy as the SVM trained over the whole dataset.

Keywords: Support vector machines, pattern recognition, mixture, fuzzy integral, fuzzy measure

1. Introduction

Since Vapnik’s publication “the nature of statistical learning theory” [1], Support Vector Machines (SVMs) were extensively used for various applications of pattern classification. Although they were originally formulated to solve binary classification problems, SVMs can achieve higher generalization performance than conventional classifiers such as Naïve Bayes and artificial neural networks [2,3]. In fact, these classifiers proceed by minimizing the training error, which does not necessarily conduct to a minimal generalization error. On the contrary, SVMs employ the structural risk minimization which minimizes an upper bound on the generalization error that is constituted by the sum of the training error and a term related to Vapnik-Chervonenkis dimension of the learning machine [1,4]. Besides, in addition to the number of user-defined parameters, which is very small, SVMs have an elegant formulation that makes them suitable for problems with high dimension input space [5].

*Corresponding author. E-mail: hmemmour@lycos.com.

However, the time required for SVM optimization increases very quickly with respect to the size of the dataset. Theoretically, to learn \( l \) examples the SVM needs a training time that is about \( l^3 \). However, recent empirical investigations on state-of-the-art SVM implementations showed that it is much closer to \( l^2 \) than \( l^3 \) [6–8]. This finding is related to kernel evaluations, which are quadratic to the number of data. On the other hand, the time complexity becomes more important with multiclass problems. For instance, the runtime of the One-Against-All (OAA) implementation, which performs \( M \) binary SVMs to solve a \( M \)-class classification problem approximately equals \( M \times l^2 \). Then, in order to accelerate the SVM optimization several methods were developed over the past years [5]. The Sequential Minimal Optimization (SMO) algorithm proposed in [9] is the fastest one because it solves at each step the smallest possible optimization problem to avoid extra matrix storage. Nevertheless, in many applications such as handwriting and face recognition, SVM training is still a time-consuming task. Therefore, Flake and Lawrence [10] proposed the use of an initial caching prototype of sequential iterative optimization to speedup the execution time of SMO. Fur-
thermore, many other algorithms were proposed to accelerate the SVM training. Among them, we note the proximal SVM that is based on the idea of assigning data to the closer of two parallel planes instead of two hyperplanes [11]. The runtime evaluation for this algorithm showed that it can be two times faster than SMO. Based on the same idea, the Successive Over-Relaxation (SOR) algorithm allows a linear convergence of SVMs and can be 1.3 faster than SMO [12]. Thenafter, Joachims proposed in [13] the Cutting-plane scheme to allow a fast training of linear SVMs. However, this algorithm does not consider non-linear SVMs that are commonly used in pattern recognition. Moreover, Chakrabati et al. [14], demonstrated that even when the best parametric selection is used, linear SVMs can be beaten in both running time and accuracy by other classifiers.

Far from techniques that modify optimization algorithms, Dong et al. [15], used the divide-and-conquer principle to perform a parallel optimization on independent sub-sets of the dataset to quickly remove non-support vector data. Although sub-sets training is fast, many additional calculations such as feature extraction as well as a second global optimization step are required. Recall that the divide-and-conquer idea was already employed through mixtures of experts in order to drastically reduce the runtime of SVMs [6]. In the mixture, outputs of SVMs trained over different sub-sets of the dataset are selectively combined throughout a gater. Conventionally, the gater is an artificial neural network trained over the full dataset. However, in addition to be tied to a mean squared error loss function, another drawback of such a gater consists of the training duration, which can be very large. Thereby, authors proposed an improved version, which uses local neural network-gaters trained over their own sub-sets [7]. Nevertheless, the use of neural networks does not constitute an optimal choice in practice since they significantly extend the mixture runtime.

In order to overcome this limitation, we propose a new mixture scheme based Sugeno’s fuzzy integral. The fuzzy integral is an evidence fusion operator in which both objective evidence supplied by various sources and the expected worth of these sources with respect to the full dataset are considered in the fusion process [16, 17]. This rule does not concern the mixture objective but it was very successful for classifier combination in various fields such as handwriting recognition [18, 19] and change detection in remotely sensed imagery [21]. Presently, the fuzzy integral is regarded as a local gater, which combines objective evidences of SVM outputs according to their reliabilities expressed by fuzzy measures. In addition to its implementation simplicity, this fuzzy mixture has two main advantages. First, it is much faster since it avoids the training time of the neural network as well as experiments required to find its best architecture. The second advantage is that it combines SVMs by considering their worth with respect to the full database.

The remaining of this paper is arranged as follows. Section 2 reviews SVMs theory and introduces the proposed mixture scheme. Section 3 presents the results obtained for mixtures of binary SVMs as well as the OAA implementation where experiments are conducted on two applications of pattern recognition. Finally, the last sections discuss and draw the main conclusions of the paper.

2. Fuzzy integral-based mixture of SVMs

The idea of SVM mixture was introduced in Kwok’s paper on support vector mixtures [22] where multiple SVMs trained over the full dataset are linearly combined by a gating network. The case in which SVMs are trained on different parts of the dataset was introduced by Collobert et al. [6], in order to alleviate the time complexity. Thus, a complex learning problem is broken into a predefined number of sub-sets each of which is trained by a SVM module. The number of sub-problems is defined by the user while data that constitute each of them are randomly selected from the full dataset. Thenafter, SVM outputs are combined by a gater module according to:

\[
f(x) = \sum_{i=1}^{n} f_i(x) \cdot \pi(x) \tag{1}
\]

\(f_i(x)\): output of a SVM \(z_i\) for a sample \(x\).
\(\pi\): gater output for a sample \(x\).
\(n\): number of SVM modules (which is also the number of sub-sets).

Conventionally, the gater is an artificial neural network trained over the entire dataset. Since it is a learning machine, the neural network gives additional knowledge about data but it generates many practical limitations. First, the user is faced to a variety of learning rules and many experiments are required to find the best network configuration. Moreover, the related computation cost makes their use quite prohibitive in a mixture framework. To overcome these limitations, we propose a new mixture scheme that is based on the notion of fuzzy integral. Specifically, the fuzzy integral
is used as a local gater, which combines SVM outputs by considering the importance of each of them towards the final decision. This approach is described in more detail in the following section after a brief review on SVMs.

2.1. Support vector machines

Support Vector Machines (SVMs) construct binary classifiers from a set of labeled training samples defined by: \((x_i,y_i) \in \mathbb{R}^N \times \{+1\}\), where \(i = 1, \ldots, l\) \((l\) is the number of training data). SVMs seek the linear separating hyperplane with the largest margin by solving the following optimization problem [2,23]:

\[
\begin{align*}
\text{Minimize} & \quad \frac{1}{2} \|w\|^2 \\
\text{Subject to} & \quad y_i (w^T x_i + b) \geq 1 \quad \forall i
\end{align*}
\]  

(2)

\(T\) denotes the transpose, \(b\) is a bias while \(w\) is the normal to the hyperplane.

When inequalities in Eq. (3) do not hold for some data, the SVM is non-linearly separable. Then, the margin of separation is said to be soft and non-separable data \(\xi_i\) into the decision surface [4]. Then, the goal is to find a hyperplane which minimizes misclassifications while maximizing the margin of separation such that:

\[
\begin{align*}
\text{Minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{l} \xi_i \\
\text{Subject to} & \quad y_i (w^T x_i + b) \geq 1 - \xi_i
\end{align*}
\]  

(4)

\(C\) is a user-defined parameter that controls the tradeoff between the machine complexity and the number of non-separable data [4]. Commonly, a dual Lagrangian formulation of the problem in which data appear in the form of dot products, is used:

\[
\begin{align*}
\text{Maximize} & \quad L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\
\text{Subject to} & \quad \sum_{i=1}^{l} \alpha_i y_i = 0
\end{align*}
\]  

(7)

where \(\alpha_i\) are Lagrange multipliers.

The dual problem is useful when data are not linearly separable in input space. In this case, the linear separation is obtained by mapping data into a feature space via a kernel function such that: \(K(x_i, x) = \langle \phi(x_i), \phi(x) \rangle\). The dual problem becomes:

\[
L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)
\]  

(8)

Thereby, the decision function is expressed in terms of kernel expansion as:

\[
f(x) = \sum_{i=1}^{S_v} \alpha_i y_i K(x_i, x) + b
\]  

(9)

\(S_v\) is the number of support vectors which are training data for which \(0 < \alpha_j \leq C\). The optimal hyperplane corresponds to \(f(x) = 0\) while test data are classified according to:

\[
x \in \begin{cases} 
\text{positive class if } f(x) > 0 \\
\text{negative class if } f(x) < 0
\end{cases}
\]  

(10)

All mathematical functions which satisfy Mercer’s conditions are eligible SVM-kernels. Presently, the Radial Basis Function (RBF) is used.

\[
RBF(x_i, x) = \exp\left(-\frac{1}{2\sigma^2} \|x_i - x\|^2\right)
\]  

(11)

\(\sigma^2\) is the kernel variance.

Furthermore, various approaches were proposed to extend SVMs for multi-class classification problems [23]. The One-Against-All (OAA) is the earliest and the most commonly used implementation. For a \(M\)-class problem, it performs \(M\) SVMs each of which is designed to separate a class from all the others. The \(m^{th}\) SVM is trained with all of the examples in the \(m^{th}\) class with positive labels, and all other examples with negative labels. This leads to a training time that is \(M \times l^2\). Then, data are assigned to the positive class with the highest output as:

\[
\arg \max_{m=1}^{M} f_m(x)
\]  

(12)

For a thorough survey of SVMs, the reader may refer to [1,4].

2.2. Fuzzy integral-based mixture

The proposed fuzzy mixture employs the divide-and-conquer principle, which breaks a dataset in some sub-sets that are simpler to learn. SVMs trained over the different sub-sets are non-linearly combined through local gaters. These gaters are computed by using Sugeno’s fuzzy integral and its associated fuzzy measure, which do not involve any training stage or parametric selection.
2.2.2. Fuzzy measure

Let \( Z \) be a finite set of sources, a set function \( g : 2^Z \rightarrow [0, 1] \) is called fuzzy measure if \([18,19]\):

\[
\begin{align*}
g(\emptyset) &= 0, \quad g(Z) = 1 \\
g(A) &\leq g(B) \quad \text{if} \quad A \subset B
\end{align*}
\]

\( A \) and \( B \) are two elements of \( Z \). This fuzzy measure does not follow the addition (i.e. combination) rule, that is if \( A, B \subset Z \) so that \( A \cap B = \emptyset \):

\[
g(A \cup B) = g(A) + g(B)
\]

Therefore, Sugeno introduced a fuzzy measure depending on a parameter \( \lambda \) to express the degree of interaction between two elements satisfying the following additional property:

\[
g(A \cup B) = g(A) + g(B) + \lambda g(A)g(B)
\]

2.2.2. \( \lambda \)-fuzzy measure and fuzzy integral

Let \( Z = \{Z_1, \ldots, Z_n\} \) be a set of \( n \) SVMs trained over \( n \) different sub-sets of a dataset. For a given sample, let \( h_m(z_i) \) be the objective evidence of the SVM \( z_i \) for the class \( m \) and let \( g_m(z_i) \) be the fuzzy measure, which expresses its performance in this class. The set of SVMs is then rearranged such that the following relation holds: \( h_m(z_1) \geq \cdots \geq h_m(z_n) \geq 0 \). This leads to an ascending sequence of SVMs such as \( A_1 = \{z_1\} \), \( A_i = \{z_1, \ldots, z_i\} \), \( A_n = \{z_1, \ldots, z_n\} \). The corresponding fuzzy measures are constructed as \([4]\):

\[
\begin{align*}
g_m(A_1) &= g_m(z_1) \\
g_m(A_i) &= g_m(A_{i-1} \cup z_i) \\
&= g_m(A_{i-1}) + g_m(z_i) + \lambda g_m(A_{i-1}) g_m(z_i)
\end{align*}
\]

For each class, \( \lambda \) is calculated by solving a \( n \)-1 degree equation:

\[
\lambda + 1 = \prod_{i=1}^{n} \left(1 + \lambda g_m(z_i)\right)
\]

\( \lambda \) is the unique root that is different to zero and \( \lambda \in ]-1, \ldots, +\infty[ \). Note that Eq. (15) provides both the weight of a single SVM and the weight of a set of SVMs. In addition, there are no rules, which would be followed to attribute \( g_m \) values \([18]\). If sufficient knowledge about SVMs is available, the user can estimate subjectively the importance of each of them by using \( g \) values in the range 0 to 1. Also, if SVMs have the same importance, the same weight can be associated to each of them. In this case, the weight can be \( 1/n \) (where \( n \) is the number of SVMs or sub-sets). In the present work, since sub-sets are randomly generated from the original dataset a priori knowledge about SVM performance is not available. Indeed, SVM reliability depends on the number of training data of each class, which varies from a sub-set to another. Therefore, the recognition accuracy seems to be the best importance indicator that can be used as a fuzzy measure. The recognition accuracy corresponds to the ratio between the number of data that are correctly recognized to their total number. If high precision is achieved, fuzzy measure values are close to 1. Otherwise, they are close to 0.

Based on the \( \lambda \)-fuzzy measure, Sugeno introduced the fuzzy integral \( I_S \) so that: For a function \( h : Z \rightarrow [0, 1] \) and a class \( m \), the fuzzy integral with respect to a fuzzy measure \( g \) is defined as \([3-5]\):

\[
I_S(m) = \max_{i=1}^{n} \left[ \min_{i} \left( h_m(z_i), g_m(A_i) \right) \right]
\]

The fuzzy integral is interpreted as searching the maximal grade of agreement between objective evidences and expectations \([24]\). However, SVMs provide single outputs which favor either positive or negative classes while evidence values should be in the range 0 through 1. Therefore, to adapt SVM outputs with the fuzzy integral requirements, we propose in what follows a fuzzy class membership model to generate evidence values of SVM outputs.

2.2.3. Fuzzy membership model

Let \( f(z_i)(x) \) be the output of the SVM \( z_i \) obtained for a sample \( x \) to be classified. The fuzzy class membership degrees \( h_{m+}(z_i) \) and \( h_{m-}(z_i) \) associated to positive and negative classes are defined as follows (Notice that for multi-class problem, only \( h_{m+}(z_i) \) are required):

\begin{algorithm}
\begin{align*}
\text{Algorithm 1: Fuzzy class membership degrees} & \\
\text{If } f(z_i)(x) > 1 \text{ then } & \{ h_{m+}(z_i) = 1 \} \\
\text{Else} & \\
\text{If } f(z_i)(x) < -1 \text{ then } & \{ h_{m+}(z_i) = 0 \} \\
\text{Else} & \\
& \{ h_{m+}(z_i) = \frac{1 + f(z_i)(x)}{2} \} \\
& \{ h_{m-}(z_i) = \frac{1 - f(z_i)(x)}{2} \}
\end{align*}
\end{algorithm}

For a binary classification, the mixture requires the calculation of the fuzzy integral for both positive and negative classes using (17). Then, data are assigned to the class with the highest fuzzy integral value. Furthermore, to reduce the runtime of the OAA implemen-
tation, the mixture is applied to each binary problem designed to separate a class from all the others. In this case, only positive class evidences are required. As shown in algorithm (2), for all classes the same number of sub-sets is used while data are assigned to the positive class with the highest fuzzy integral value. Notice that in multi-class mixture, steps 2, 3 and 4 are performed for each problem designed to separate a class from the others.

Algorithm 2: Fuzzy support vector mixture

1. Divide the dataset into \( n \) random sub-sets.
2. Train SVM modules \((z_i)\) separately over the different sub-sets.
3. Generate fuzzy measures \( g_{m+} (z_i) \) (and \( g_{m-} (z_i) \) if binary mixture) of trained modules.
4. For a sample \( x \) to be classified do:
   1. Calculate separately, outputs \( f^m(x) \) of trained modules.
   2. Generate membership degrees \( h_{m+} (z_i) \) (and \( h_{m-} (z_i) \) if binary classification).
   3. Generate new fuzzy measures using (15).
   4. Calculate the fuzzy integral \( I_S (m+) \) (and \( I_S (m-) \) if binary classification).
5. Assign \( x \) according to:

\[
\text{Max}_{m=1}^{M} I_S (m) \quad \text{if multiclass mixture,}
\]

\[
\text{or Max} (I_S (m+) , I_S (m-)) \quad \text{if binary classification).}
\]

3. Experimental analysis

The effectiveness of the proposed fuzzy support vector mixture is investigated on two pattern recognition applications. In a first step, experiments are conducted for Optical Character Recognition (OCR). Then, to verify that the results are replicable for data with very high dimension, similar experiments are performed for face recognition. For both applications, the mixture is evaluated on both binary and multi-class problems using standard benchmark datasets. Besides, data are scaled in the range \([0,1]\) and presented to SVMs without any feature extraction or size reduction preprocessing. The mixture aim is to reduce the runtime while keeping at least the same performance as the single SVM trained over the full dataset. Tests are performed on a 2.8 GHz Pentium 4 CPU. Besides, all SVM modules use the RBF kernel for which \( \sigma \) and \( C \) are experimentally tuned at \( (\sigma = 5, C = 10) \). Moreover, the results obtained by Uniform Averaging (UA) mixture are given for comparison since the fuzzy integral behaves as an average weighted by fuzzy measures. In addition, the average rule is one of the most effective rules for combining classifiers [25]. For this reason, it was the initially employed as automatic gater for uniform mixtures of experts. Furthermore, performance evaluation is carried out by using the following criteria:

- Training Time (TT) and Recognition Time (RT) acceleration is expressed by the Speedup Index, which results from dividing the time required by the single SVM (which is the SVM trained on the full dataset) over the time required by the mixture.
- Error Difference (ED) which is obtained by subtracting the error rate of the single SVM (i.e. trained over the full dataset \( E_{n1} \)) from the mixture error rate \( (E_m , n \geq 2) \) is the number of modules):

\[
ED_m = E_{n1} - E_{1}
\]

The mixture improves accuracy if its respective ED is negative while it is considered as reliable as the single SVM if the ED is zero.

3.1. Optical Character Recognition

OCR is performed on the well-known US Postal Service handwriting recognition task (USPS). This dataset contains normalized gray-level images of handwritten digits \((0, \ldots, 9)\), extracted from US postal envelopes. All images are segmented and normalized to a size of \(16 \times 16\) pixels yielding 256 dimensional datum vector. Data are partitioned into 7291 training images and 2007 test images. Note that many test samples are corrupted so that even human cannot classify correctly; some of them are depicted in Fig. 1.

The binary classification problem is designed to separate a randomly selected class (positive class that is the numeral ‘6’) from all the others. There are 664 training images for the numeral ‘6’ versus 6627 images for the second class. Thereby, a set of mixtures in which the number of SVM modules varies from 2 to 20 are performed.

Figure 2 plots the ED of both Uniform Averaging (UA) and Fuzzy Integral (FI) mixtures. We remark that for all mixtures UA produces lower accuracy than the single SVM since the respective ED are more than zero. In consequence, this outcome highlights limitations of the average rule. On the contrary, the FI mixture improves the performance accuracy by 0.2% using 2, 3 and 4 modules while it provides the same accuracy as the single SVM from 5 to 8 modules. However,
beyond 8 modules we notice a little decrease in mixture accuracy since the ED is about +0.04. Hence, fuzzy measures tend to keep a track of the behavior of each SVM in the mixture with respect to the full database. However, when sub-sets become very small, their respective SVM modules have weak generalization capabilities over the full database. Thus, a trade-off between runtime reduction and preserving accuracy should be taken into consideration. Furthermore, performance results of FI mixtures in terms of speed-up index obtained for the different mixtures are illustrated in Fig. 3. Roughly, the speedup grows proportionally to the number of modules. With 2 modules, the mixture is 20 times faster than the single SVM when using 2 modules, and becomes 90 times faster with 20 modules in both training and recognition stages.

For multi-class OAA, mixtures were performed by considering 2, 3 and 4 modules. Figure 4 depicts ED rates obtained for both UA and FI mixtures. It is easy to see that the FI improves the error rate to more than 4% and 2% using 2 and 3 modules, respectively. On the contrary, with 2 modules the UA gives approximately the same performance as the single SVM but with 3 modules its ED exceeds 16%. Furthermore, Fig. 5 summarizes speedup evaluations provided by the FI mixture. We can note that the best acceleration is obtained with 4 modules where the mixture is 3 times and 5 times faster than the single SVM in TT and RT, respectively. Specifically, the RT is sped up because of the number of SV, which is smaller with the mixture. In fact, the SV set contains all information a given classifier needs for constructing the decision function. Commonly the number of SV makes up about 5% to 15% of the whole dataset [26]. Nevertheless, experiments show that the larger this dataset the smaller this ratio. The reason why the total number of SV which are selected from all mixture sub-sets is smaller than the number of SV employed by the SVM trained on the full dataset.

Furthermore, handwriting recognition becomes realistically a challenging task when it deals with alphanumeric characters because of the high ambiguity between some cursive uppercase letters and handwritten digits. In order to investigate the behavior of the proposed approach with respect to problems associated
with alphanumeric characters, we incorporated into the USPS dataset handwritten uppercase letters.

Letters are extracted from the C-Cube\(^1\) (Cursive Character Challenge) dataset which contains 57293 cursive characters manually extracted from cursive handwritten words, including both upper and lower case versions of each letter. A set of 6018 uppercase letters were extracted and normalized according to USPS data. This leads to a training set with 11734 samples and 3582 test samples. Some of these samples are shown in Fig. 6 where it is easy to see their similarity with some handwritten digits.

The results obtained for ED and speedup evaluations are quite similar to those obtained for digits recognition (See Figs 7 and 8). In fact, the FI mixture improves at once, the runtime and error rate by using 2 and 3 modules. Comparatively to USPS dataset where the ED is about $-3.5\%$ and $-2\%$, respectively, for alphanumeric data it is about $-1.2\%$ and $-0.5\%$. This reveals that the FI mixture efficacy decreases with the increase of classes in the problem. On the contrary, in all experiments the UA mixture achieved weak performance expressed by high ED rates. Also, variations of speedup index confirm again the ability of mixture to reduce the runtime. Specifically, for the training stage the speedup varies from 7 to 10 times with 2 and 3 modules while for the recognition stage, it varies from 1.9 to 3.1 times, comparatively to the single SVM.

Since alphanumeric characters present high interclass ambiguity, it was straightforward to see how the proposed approach behaves with such complexity. Thereby, Fig. 9 depicts variations of ED rates in alphanumeric classes obtained for both UA and FI mixtures. It is easy to see that the UA mixture performs better with digits than letters. More precisely, it improves the accuracy in some digit classes (ED $\leq 0$) while it gives poor accuracies with letters whose ED rates are often more than $+10\%$. This outcome explains why the UA mixture has weak performance compared to the single SVM. On the contrary, the FI mixture provides more efficient results especially with 2 and 3 modules where it achieves at least the same accuracy as the single SVM in 26 classes. Notice that a negative ED corresponds to an improvement in the error rate of the class and so conducts to less confusion with the others. The most critical classes are L, O, R, and T for which the FI provides positive ED in all mixture tests. In fact, these classes have only few samples in some sub-sets because of the random partition of the dataset. In spite of this, using 2 and 3 modules, the FI mixture outperforms the single SVM and thus handles better the ambiguity of alphanumeric characters.

### 3.2 Automatic Face Recognition

To verify the applicability of the proposed approach to other tasks of pattern recognition as well as in order to check how it scales with respect to data dimension, similar experiments were conducted on Cambridge ORL.
Fig. 9. Error difference evaluations for mixture approaches respective to alphanumeric data.

Fig. 10. Data Examples (a) ORL dataset, (b) UMIST dataset.
and UMIST face recognition datasets. UMIST dataset contains 20 classes (a class corresponds to a person) for which the number of samples varies between 19 and 48 images taken in a range of poses from profile to frontal views (see Fig. 10.a). ORL dataset contains 40 classes each of which has 10 images taken at different times and including variations in facial expressions like open and close eyes as well as smiling or no smiling (Fig. 10.b). All images were taken against a dark homogeneous background and have a size of $(112 \times 92)$ pixels yielding a 10304 dimensional datum vector.

To perform mixture tests for a binary face recognition problem, for each person (or class) 5 images were randomly selected from the two datasets in order to constitute the training set. The remaining images were used to form the test set. The positive class corresponds to a randomly chosen class (Presently, the first image of (Fig. 10.b) corresponds to the positive class while all other classes are grouped to form the negative class). Since it is composed of 5 training images, we duplicated these images and placed them in uniformly separated positions of the dataset to avoid having sub-sets without any examples of this class. This leads to 399 training images.

As depicted in Fig. 11, from 2 to 7 modules, the FI mixture achieves similar accuracy as the single SVM. Beyond 7 modules, it seems less efficient since its ED goes from 0.33% with 8 modules to 2.33% with 20 modules. This latter is the most critical case because it does not classify correctly all test samples of the positive class which constitute 1.66% of test data. Besides, the comparison with the UA mixture reveals that for 2 modules, both mixtures achieve the same performance. From 3 modules, UA mixture produces higher ED which reflects lower accuracy rates. Furthermore, the inspection of speedup indices depicted in Fig. 12, shows that the runtime reduction grows proportionally to the number of modules. Hence, up to 7 modules, the FI mixture is as accurate as the single SVM while being more than 100 times faster.

For the OAA implementation, mixture tests were independently performed for each dataset. UMIST dataset contains 20 classes represented by 287 training images and 288 test images while the ORL is composed of 40 classes each of which has 5 training and 5 test images. The ED evaluations for all mixtures are drawn in Fig. 13. For UA mixture provides an ED that equals zero in the best cases (2 and 3 modules), which means that this approach does not give any accuracy improvement. Also, the FI mixture provides a similar performance for ORL dataset. Nevertheless, with the UMIST dataset, it allows a significant accuracy improvement yielding a gain of 16% in error rate using only 2 modules. This gain decreases with the increase of the sub-sets number since with 3 modules the FI mixture performs slightly better than the single SVM while it achieves a bad performance with 4 modules. Furthermore, the examination of the speedup values, which are plotted in Fig. 14, shows an important acceleration in the TT. For both datasets this acceleration goes from 3.4 times using 2 modules to more than 8 times using 4 modules. On the contrary, the RT acceleration is relatively small especially for ORL dataset where it is less than 2 times for all mixtures. This outcome can be explained by the fact that the number of training samples is small (5 samples for each class). This makes both SVM trained over sub-sets and the SVM trained over the full dataset using almost all training data as support vectors. Therefore, the computation time of the decision function is approximately the same for both
The fuzzy integral proceeds by combining objective evidences of SVM outputs with respect to their reliabilities. In fact, due to its low computational requirement, the uselessness of both training and parametric selection as well as the ease of implementation, it constituted an optimal candidate to substitute neural network gaters. Besides, since all previous mixture approaches were employed with binary SVMs, the present work tried to investigate the applicability of mixture for a multi-class SVM implementation which is the OAA approach.

The effectiveness of the proposed mixture was tested on two different applications which are OCR and face recognition, using standard datasets. Experiments were directed in order to investigate its behavior with respect to two major problems that are characterized by data with very high ambiguity and data with very large dimension. The first problem was investigated with...
OCR tests where the mixture was performed on the well-known USPS dataset of handwritten digits whose learning constitutes a challenging task. Then, cursive uppercase letters were injected into the USPS in order to extend this dataset and increase the intrinsic ambiguity between classes. On the other hand, the problem of data dimension was investigated with face recognition experiments by using data in their original form without dimensionality reduction. The main observations that can be drawn from all experiments are summarized as follows:

- The speedup index grows proportionally to the number of modules. In particular, the runtime is significantly reduced while keeping the same performance as the single SVM trained over the full dataset. Besides, for both binary and multi-class problems, 2 and 3 modules constitute the optimal number of modules (or sub-sets) where the mixture makes the best tradeoff between time reduction and accuracy preservation.

- The fuzzy support vector mixture efficacy strongly depends on the number of samples representing classes in the different sub-sets. In fact, if classes are sufficiently represented in all sub-sets, the mixture outperforms the single SVM in both runtime and accuracy. On the contrary, it improves the runtime to the detriment of accuracy. This explains why the mixture improves the error rate in some OCR tests while it does not give similar results for face recognition (Recall that for face recognition the number of data is about 5 samples per class, only).

- From binary to multi-class SVMs the mixture tends to lose its speedup capabilities. In fact, it works quite well with binary problems since it is about 90 times faster than the single SVM. However, with multi-class SVMs the speedup is less than 10 times for all dataset. In addition, the number of classes is itself another factor that controls the mixture performance. OCR tests showed that the mixture can give a gain of 2% in error rate for numerical classes (10 classes) while for alphanumeric classes (36 classes) the gain does not exceed 1.2%.

Again, the comparison with the uniform averaging mixture highlighted the role of fuzzy measures in improving or keeping performance. As reported in [25], the average is a combination rule that can nullify the effect of poorly performing classifiers by assigning adequate weights for each of them. Nevertheless, the advantage of the fuzzy integral is that it gives through fuzzy measures the weight of individual classifiers as well as the weight of a set of classifiers which is very important for the mixture concept. In consequence, the UA mixture provides relatively poor ability to deal with data variability since in most experiments it was less accurate than the single SVM. Also, the fuzzy support vector mixture is a promising approach to reduce the training time of large-scale problems. Also, in comparison with several previous works performed on SVM classifiers, it gives more satisfactory results in terms of runtime reduction.

5. Conclusion

In this paper we proposed a new fast SVM mixture based on fuzzy integral. The main advantage lies in the fact that the gater function is computed at decision without needing a training stage as required by the neural network gaters. Experiments directed on various benchmark datasets provided very encouraging results and showed that it can significantly reduce the runtime of large-scale problems while keeping at least the same accuracy as the single SVM. Ultimately, in addition to its implementation simplicity, another advantage of this method is that it can be easily parallelized (Each SVM can be performed on a separate CPU) to be again more much faster than a single SVM.

References