A Boosting Approach to Multiview Classification with Cooperation

Sokol Koço and Cécile Capponi[*]

CNRS – LIF UMR 6166 – Aix-Marseille University
Technopole Chateau-Gombert – 39 rue Joliot Curie,
CMI – 13453 F-Marseille cedex 13
{sokol.koco,cecile.capponi}@lif.univ-mrs.fr
http://www.lif.univ-mrs.fr

Abstract. In many fields, such as bioinformatics or multimedia, data may be described using different sets of features (or views) which carry either global or local information. Some learning tasks make use of these several views in order to improve overall predictive power of classifiers through fusion-based methods. Usually, these approaches rely on a weighted combination of classifiers (or selected descriptions), where classifiers are learned independently. One drawback of these methods is that the classifier learned on one view does not communicate its failures within the other views. This paper deals with a novel approach to integrate multiview information. The proposed algorithm, named Mumbo, is based on boosting. Within the boosting scheme, Mumbo maintains one distribution of examples on each view, and at each round, it learns one weak classifier on each view. Within a view, the distribution of examples evolves both with the ability of the dedicated classifier to deal with examples of the corresponding features space, and with the ability of classifiers in other views to process the same examples within their own description spaces. Hence, the principle is to slightly remove the hard examples from the learning space of one view, while their weights get higher in the other views. This way, we expect that examples are urged to be processed by the most appropriate views, when possible. At the end of the iterative learning process, a final classifier is computed by a weighted combination of selected weak classifiers.

This paper provides the Mumbo algorithm in a multiclass and multiview setting, based on recent theoretical advances in boosting. The boosting properties of Mumbo are proved, as well as some results on its generalization capabilities. Several experimental results are reported which point out that complementary views may actually cooperate under some assumptions.

Keywords: Boosting, Multiview Learning, Supervised Learning, Classification.

[*] This work is supported by the French agency ANR, project DECODA, contract no 2009-CORD-005-01, and the French business clusters Cap Digital and SCS.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part II, LNAI 6912, pp. 209–228, 2011.

1 Introduction

In many application domains of machine learning, such as bioinformatics or multimedia indexing, data may be described by several sources or views [1], [2]. When facing a classification or a regression task, the use of these views might be of great interest, since each view is supposed to carry some information that the other views would not embed. Fortunately, there exist many methods to select the most informative sources, or set of features, that either best discriminate data concepts or best describe one concept among others [3] [4].

Most of these selective methods are statistically founded, which means that they tend to disregard localized – isolated – information although it could be useful to compensate the lack of performance on some (group of) learning examples. Indeed, real-life data descriptions are often noisy. When the noise rate of a set of feature descriptions reaches a threshold, which depends on the problem, no learning algorithm has been proved, neither theoretically nor empirically, to be able to overcome the noise disruption on the generalization capabilities of classifiers. Yet, multiview learning approaches should enable some localized failures of classifiers trained on one view, to be compensated by – or subordinated to – the abilities of classifiers on the other views.

Up to now, several approaches of multiview learning have been developed, most of them in the semi-supervised setting. The first of them was the well-known *Co-Training* algorithm [5], which was based on far too much restrictive assumptions [6]. Other semi-supervised multiview algorithms have then been developed and theoretically founded, all of them promoting the agreement between views [7] [8] [9], except for the semi-supervised boosting approach presented in [10]. No compelling application on real-life problems has promoted these approaches yet, although many problems would actually need a multiview learning process.

In addition, in the supervised setting, leveraging the performances of classifiers learned on different views has mainly been performed through fusion-based methods, either early or late fusion [11] [12]. Early fusion consists in grouping (selected) descriptions of the different views into a large vector, and then to learn a classifier on this resulting view. On the opposite, late fusion allows one classifier per view to be learned, while the final classifier is a combination of them. Usually, late fusion performs better than early fusion. Yet, none of them leads to good performances when the views are of unbalanced informative content, for weaker views tend to reduce the final performances. An empirical comparison of these methods applied on multimedia problems is presented in [13].

Whatever the fusion-based approach is, it relies on a weighted combination of classifiers (or selected descriptions), where classifiers are learned independently. One drawback of these methods is that the classifier learned on one view does not communicate its failures to the other views. Besides, views must be independent in order for the combined classifier to be most accurate.

Yet, we think it could be interesting that, when the classifier on a view fails on a region of examples in the instance space, it could entrust the other views with the classification of these examples. One of the major difficulties is then to delimit the concerned subareas of the instance space, without loss of generalization capabilities. Instead of precisely locating these subareas, we propose an algorithm based on boosting [14] [15] whose principle is to slightly remove the hard examples from the learning space of one view, while their weights get higher in the other views. This way, we expect that examples are processed by the most appropriate views.

In order to implement this principle, we designed Mumbo as a boosting-like algorithm which works from different views on data. Each view is associated with a weak learner, and one distribution per view is maintained (section 2). The distributions are updated at each iteration in such a way that views communicate the ones to the others their capability of processing learning examples. Hence, not only the distribution update in one view takes into account the performances of that view in classifying the learning examples, but it also embeds the performances of the other views on these examples. The properties of Mumbo are discussed and proved in section 3: both empirical and generalization errors are bounded. In order to warrant the boosting properties, and the generalization error bound, we define a global distribution of examples that reflects the overall behaviour of the algorithm within a given hypothesis space. In section 4, we present experimental results of Mumbo on synthetic data, which confirm that Mumbo is a boosting algorithm, better than other basic fusion approaches. Before concluding, we discuss this approach with other methods, and give some clues to improve Mumbo (section 5).

2 The Mumbo Algorithm

2.1 Principles and Assumptions

Mumbo is a multiview boosting algorithm: each example of the learning sample S is represented by several independent sets of features. Each one of these representations is called a *view*. Eventually, these views are used to train models, which are then used to classify other examples. Even though these models are learned on different representations of the same examples, they are by no means equal performance wise. Classifiers learned on some views perform better than those learned on other views, due to the noise in the data and/or views, or the lack of information, etc., which may be different from one representation space to another. In other words, we may define the strength of a view as the possibility to learn a good classifier on that view. At the same time, the weakness of a view may reflect the impossibility of learning a good classifier from the instance space defined by this view.

More formally, let S be a sample of n tagged examples chosen according to some distribution $\mathcal{D}$ over $X \times Y$, where X is some instance space and Y is some class space. Let V be a view, $\mathcal{H}$ be the space of all the hypothesis that we can learn on V and h be the best classifier that we can learn on this space. Finally, let ρ be the error of random guessing over S and $\sigma_V \leq \rho$ be the lower bound of the error of h on S. We define the notion of weak and strong view as follows : V is called a strong view if σ_V is near 0 and V is called a weak view if $\gamma_V = \rho - \sigma_V$ is near 0.

Mumbo has been designed in order to learn a classifier in a multiview setting, where views are supposed to be of different strengths. More specifically, we suppose that among the views, there exists one strong major view V, and several weaker minor views $v_1, \cdots, v_z$.

In our setting, γ_V is supposed to be greater than $\gamma_{v_1}, \cdots, \gamma_{v_z}$.

For example, in speech recognition, three usual views for describing a speech (or dialog) to be classified, are known to be of unequal strength [16]. The major view is the lexical analysis of a speech (syntactic trees, for example); other minor views may be the prosodic information, and syntactic information. Although the major view allows to learn rather good classifiers, researchers in speech recognition still use minor views for learning in order to compensate the failures of the major view in case of noise disruptions [17].

As pointed out in the introduction, the basic principle of Mumbo is to encourage each view v to focus on the examples that are hard to process in other views, and easy to process in v. Hence, it assumes that if one representation space does not embed information on one (set of) examples, part of that information can be provided by other representation spaces.

2.2 Framework and Notations

In this paper, we present Mumbo within the framework defined by [15], where basically γ denotes the edge of a classifier with regards to random. We use the following typings:

- matrices are denoted by bold capital letters like $\mathbf{C}$; element of row i and column j in matrix $\mathbf{C}$ is denoted $C(i, j)$, and $C(i)$ is the row i of $\mathbf{C}$.
- $\mathbf{M} \cdot \mathbf{M'}$ denotes the Frobenius inner product of two matrices.
- the indicator function is denoted by $\mathbb{1}[\cdot]$, and the cartesian product is denoted by $X_1 \times X_2$.

Let $S = \{(x_1, y_1), ..., (x_n, y_n)\}$ be the learning sample, where $x_i \in X$, and $y_i \in Y$ is the class of the example x_i. The set of classes is $Y = \{1, ..., k\}$. The set of features X is made up of different subsets: $X = X_1 \times ... \times X_m$. Each subset represents a view, as in [18]. Then, the representation of example x_i within view m is written $x_{i,m}$[1].

In this paper we use the definition of weak classifier as defined in [15], that is a classifier whose performance on a cost matrix is better than that of some edge-over-random γ baseline.

Definition 1 (edge-over-random baseline and cost matrix, by Mukherjee et al. [15]). *The edge-over-random baseline* $\mathbf{B} \in \mathcal{B}_\gamma^{eor} \subseteq \mathbb{R}^{n \times k}$*, where* $\mathcal{B}_\gamma^{eor}$ *is the space of edge-over-random baselines, is* γ *more likely to predict the correct label rather than an incorrect one on every example* i*:* $\forall l \neq y_i, B(i, y_i) \geq B(i, l) + \gamma$*, with equality holding for some* l*.*

The edge-over-random cost matrix $\boldsymbol{C}$ *puts the least cost on the correct label,* i.e. *the rows of the cost matrix come from the set* $\left\{c \in \mathbb{R}^k : \forall l, c(y_i) \leq c(l)\right\}$.

[1] When possible, we simplify $x_{i,m}$ to x_i in the scope of view m.

Definition 2 (edge-over-random baseline in the Mumbo setting). *The edge-over-random baseline used in this paper is a cost matrix $\boldsymbol{U}_\gamma$ defined as follows: $\boldsymbol{U}_\gamma(i,l) = (1-\gamma)/k + \gamma$ if $l = y_i$ and $\boldsymbol{U}_\gamma(i,l) = (1-\gamma)/k$ if $l \neq y_i$.*

The $\boldsymbol{1}_{h_{t,m}}$ matrix is the prediction matrix defined as $\boldsymbol{1}_{h_{t,m}}(i,l) = 1$ if $h_{t,m}(i) = l$ and $\boldsymbol{1}_{h_{t,m}}(i,l) = 0$ if $h_{t,m}(i) \neq l$.

Definition 3 (edge-condition, by Mukherjee et al. [15]). *Let $\mathbf{C} \subseteq \mathbb{R}^{n \times k}$ and matrix $\mathbf{B} \in \mathcal{B}_\gamma^{eor}$, an eor-baseline; we say that a weak classifier h satisfies the edge condition if $\mathbf{C} \cdot \mathbf{1}_h \leq \mathbf{C} \cdot \mathbf{B}$*

In the case of binary classification, the ith row of the baseline $\mathbf{U}_\gamma$ is $(\frac{1}{2}(1-\gamma), \frac{1}{2}(1+\gamma))$ if the label of example i is $+1$, and $(\frac{1}{2}(1+\gamma), \frac{1}{2}(1-\gamma))$ if the label of example i is -1. A given classifier h satisfies the edge condition if $\sum\limits_i \mathbf{C}(i, h(i)) \leq \sum\limits_i \left\{(\frac{1}{2} - \frac{\gamma}{2})\mathbf{C}(i, \overline{y}_i) + (\frac{1}{2} + \frac{\gamma}{2})\mathbf{C}(i, y_i)\right\}$ and [15] shows that this condition is equivalent to the usual weak learning condition for binary classification.

2.3 The Core of Mumbo

Mumbo (algorithm 1) is an attempt to promote the collaboration between major and minor views, in order to enhance the performances of classifiers usually learned only on the major view. It is a boosting algorithm theoretically founded on the framework presented in [15].

Y is not limited to $\{-1, +1\}$, since we are in the multiclass setting, based on the theoretical approach of [15], whose one of the main ideas is to replace the weights of the examples with a cost matrix. We use the same idea here: $\mathbf{C}$ is a cost matrix so that $\mathbf{C}(i,l)$ is the cost of assigning the label l to the example i. Since we deal with more than one view, we use one cost matrix $\mathbf{C}_j$ per view j, and a global cost matrix $\mathbf{C}_G$. Thus $m+1$ cost matrices are maintained.

Mumbo runs for T rounds: at each round t, a weak learner is trained on each view v, which returns m weak classifiers $h_{t,m}$. These weak classifiers must satisfy the weak learning condition given in definition 3. For each $h_{t,j}$, we compute a parameter $\alpha_{t,j}$ that measures its importance depending on the edge of the $h_{t,j}$ on the cost matrix $\mathbf{C}_{t,j}$ (*c.f.* algorithm 1).

As stated before, one of the main ideas of Mumbo is to have some sort of collaboration between the different views. This idea is implemented in two different parts of this algorithm: first during the update of the m cost matrices, and second when choosing the classifier h_t selected at round t in the final combination.

The update of each cost matrix depends on all the classifiers. The i^{th} line, corresponding to the example x_i in the matrix of the view j, is updated only if the classifier learned on this view classifies correctly x_i OR if all the $m-1$ other weak classifiers misclassify it. Intuitively this means that a view gives up on the hardest examples and lets the other views handle them. In the scenario of one major and several minor views, this allows the minor views to focus on the hardest examples of the major view.

Algorithm 1. Mumbo: MUltiModal BOosting

```
Given
  – $S = \{(x_1, y_1), ..., (x_n, y_n)\}$ where $x_i \in X_1 \times X_2 \times ... \times X_m$, $y_i \in \{1, ..., k\}$
  – $m$ weak learning algorithms $WL$
  – T the number of iterations
  – a baseline $\mathbf{B}$ (edge-over-random prior baseline)
Initialize $(\forall i \in \{1, \cdots, n\}, \forall j \in \{1, \cdots, m\}, \forall l \in \{1, \cdots, k\})$:
$f_{0,j}(i, l) = 0$
$\mathbf{C}_{0,G}(i, l) = \mathbf{C}_{0,j}(i, l) = \begin{cases} 1 & \text{if } y_i \neq l \\ -(k-1) & \text{if } y_i = l \end{cases}$ where $\mathbf{C}_{0,G}$ is the global cost matrix
for $t = 1$ to $T$ do
    Train $WL$ using $\mathbf{C}_{t-1,1}, ..., \mathbf{C}_{t-1,m}$
    for $j = 1$ to $m$ do
        Get $h_{t,j}$ satisfying the edge condition on $\mathbf{B}$, and compute edge $\delta_{t,j}$ on $\mathbf{C}_{t-1,j}$,
        and $\alpha_{t,j} = \frac{1}{2} \ln \frac{1+\delta_{t,j}}{1-\delta_{t,j}}$
    end for
    Update cost matrices (for each view $j = 1 \cdots m$):
        $\mathbf{C}_{t,j}(i, l) = \begin{cases} \exp(f_{t,j}(i, l) - f_{t,j}(i, y_i)) & \text{if } l \neq y_i \\ -\sum_{p=1; p \neq y_i}^{k} \exp(f_{t,j}(i, p) - f_{t,j}(i, y_i)) & \text{if } l = y_i \end{cases}$
        where $f_{t,j}(i, l) = \sum_{z=1}^{t} \mathbb{1}[h_{z,j}(i) = l] \alpha_{z,j} d_{z,j}(i)$
        and $d_{z,j}(i) = \begin{cases} 1 & \text{if } h_{z,j}(i) = y_i \text{ or } \nexists q \in \{1, ..., m\}, h_{z,q}(i) = y_i \\ 0 & \text{else} \end{cases}$
    Choose $h_t = \underset{h_{t,j}}{\text{argmax}}(\text{edge } h_{t,j} \text{ on } \mathbf{C}_{t,G})$ and $\delta_t = \{\textit{edge of } h_t \textit{ on } \mathbf{C}_{t,G}\}$
    Compute $\alpha_t = \frac{1}{2} \ln \frac{1+\delta_t}{1-\delta_t}$
    Update $\mathbf{C}_{t,G}$ :
        $\mathbf{C}_{t,G}(i, l) = \begin{cases} \exp(f_{t,G}(i, l) - f_{t,G}(i, y_i)) & \text{if } l \neq y_i \\ -\sum_{j \neq y_i}^{k} \exp(f_{t,G}(i, j) - f_{t,G}(i, y_i)) & \text{if } l = y_i \end{cases}$
        where $f_{t,G}(i, l) = \sum_{z=1}^{t} \mathbb{1}[h_{z,m}(i) = l] \alpha_{z,m}$
end for
Output final hypothesis : $H(x) = \underset{l \in 1, \ldots, k}{\text{argmax}} f_T(x, l)$,  where  $f_T(i, l) = \sum_{t=1}^{T} \mathbb{1}[h_t(i) = l] \alpha_{t,m}$
```

In the last part of each round t, Mumbo chooses the classifier h_t among the m that minimizes the error on the global cost matrix. The confidence α_t is computed for h_t, based on its edge on the global cost matrix. Finally, the global cost matrix is updated, in a similar way that in the adaptive case of the OS algorithm [15].

The final hypothesis H is a weighted vote of the T selected weak classifiers h_t, and α_t is the weight assigned to h_t.

3 Properties of Mumbo

In this section, we present two properties of the Mumbo algorithm that together ensure it is a boosting algorithm. We first show that the update rules for the cost matrix of each view, as presented in the previous section, actually reduce the training error on this view. We then prove that the criterion for choosing the unique h_t at each step t, and eventually the update rule, allows Mumbo to be a safe boosting algorithm: the training error decreases with rounds. The most important property, a bound on the generalization error of Mumbo, is proved.

3.1 Bounding the Training Error on Each View

One property of Mumbo is that we can bound the empirical error made by the final classifiers of one view, when views are considered independently. We give a formal proof of this property, then we define a way of computing $\alpha_{t,m}$ in each round t for each view m in order to prove the decreasing of the empirical error of the final combination of weak classifiers.

Theorem 1 is an adaptation to Mumbo of the lemma presented in the supplement of [15].

Theorem 1 (bounding the empirical error in view m**).** *For a given view* m*, suppose the cost matrix* $\boldsymbol{C}_{t,m}$ *is chosen as in the algorithm 1, and the returned classifier* $h_{t,m}$ *satisfies the edge condition for the baseline* $\boldsymbol{U}_{\gamma_m}$ *and cost matrix* $\boldsymbol{C}_{t,m}$*, i.e.* $\boldsymbol{C}_{t,m} \cdot \boldsymbol{1}_{h_{t,m}} \leq \boldsymbol{C}_{t,m} \cdot \boldsymbol{U}_{\gamma_m}$.

Then choosing a weight $\alpha_{t,m} > 0$ *for* $h_{t,m}$ *makes the error* $\epsilon_{t,m} = \sum_{i=1}^{n} \sum_{l \neq y_i} \exp(f_{t,m}(i,l) - f_{t,m}(i,y_i))$*, at most a factor*

$$\tau_{t,m} = 1 - \frac{1}{2}\left(\exp(\alpha_{t,m}) - \exp(-\alpha_{t,m})\right)\delta_{t,m} + \frac{1}{2}\left(\exp(\alpha_{t,m}) + \exp(-\alpha_{t,m}) - 2\right)$$

of the loss before choosing $(\alpha_{t,m})$*, where* $\delta_{t,m}$ *is the edge of* $h_{t,m}$*,* $\delta_{t,m} = \boldsymbol{C}_{t,m} \cdot \boldsymbol{U}_{\gamma_m} - \boldsymbol{C}_{t,m} \cdot \boldsymbol{1}_{h_{t,m}}$.

Proof

Let S_+ be the set of the examples correctly classified by $h_{t,m}$, S_- the set of the examples misclassified by all the m classifiers returned by WL, and S_{-+} the set of the examples misclassified by $h_{t,m}$ and correctly classified by at least one of the other $h_{t,j}, j \neq m$.

In order to simplify the reading of the proof, we introduce the quantities: $L_{t,m}(i) = \sum_{l \neq y_i} \exp(f_{t,m}(i,l) - f_{t,m}(i,y_i))$, and $\zeta_{t,m}(i,l) = f_{t,m}(i,l) - f_{t,m}(i,y_i)$.

Using the edge condition we have :

$$\mathbf{C}_{t,m} \cdot \mathbf{1}_{h_{t,m}} \leq \mathbf{C}_{t,m} \cdot \mathbf{U}_{\delta_{t,m}} \tag{1}$$

The left and right sides of equation 1 can be rewritten as:

$$\mathbf{C}_{t,m} \cdot \mathbf{1}_{h_{t,m}} = -\sum_{i \in S_+} L_{t-1,m}(i) + \sum_{i \in S_-} \exp(\zeta_{t-1,m}(i, h_{t,m}(x_i))) + \sum_{i \in S_{-+}} \exp(\zeta_{t-1,m}(i, h_{t,m}(x_i)))$$

$$\mathbf{C}_{t,m} \cdot \mathbf{U}_{\delta_{t,m}} = \sum_{i=1}^{N} \left(-L_{t-1,m}(i)(\tfrac{1-\delta_{t,m}}{k} + \delta_{t,m}) \ + L_{t-1,m}(i)(\tfrac{1-\delta_{t,m}}{k}) \right) = -\delta_{t,m} \sum_{i} L_{t-1,m}(i)$$

So, using the edge condition 1 we obtain:

$$-\sum_{i \in S_+} L_{t-1,m}(i) + \sum_{i \in S_-} \exp(\zeta_{t-1,m}(i, h_{t,m}(x_i))) + \sum_{i \in S_{-+}} \exp(\zeta_{t-1,m}(i, h_{t,m}(x_i))) \leq -\delta_{t,m} \sum_{i \in S} L_{t-1,m}(i)$$

hence:

$$\sum_{i \in S_+} L_{t-1,m}(i) - \sum_{i \in S_- \cup S_{-+}} \exp(\zeta_{t-1,m}(i, h_{t,m}(x_i))) \geq \delta_{t,m} \sum_{i \in S} L_{t-1,m}(i) \qquad (2)$$

In order to compute the drop in loss after choosing $h_{t,m}$ with weight $\alpha_{t,m}$, let us consider three cases:

1. For $i \in S_+$:
   We have $f_{t,m}(i,l) - f_{t,m}(i,y_i) = f_{t-1,m}(i,l) - (f_{t-1,m}(i,y_i) + \alpha_{t,m})$, then:

$$\begin{aligned} \Delta_+ &= \sum_{i \in S_+} - L_{t,m}(i) - \sum_{i \in S_+} - L_{t-1,m}(i) = \sum_{i \in S_+} - \exp(-\alpha_{t,m}) L_{t-1,m}(i) - \sum_{i \in S_+} - L_{t-1,m}(i) \\ &= (1 - \exp(-\alpha_{t,m})) \sum_{i \in S_+} L_{t-1,m}(i) \end{aligned}$$

2. For $i \in S_-$:

$$\begin{aligned} \Delta_- &= \sum_{i \in S_-} \exp(f_{t,m}(i, h_{t,m}(i)) - f_{t,m}(i, y_i)) - \sum_{i \in S_-} \exp(f_{t,m}(i-1, h_{t,m}(i)) - f_{t-1,m}(i, y_i)) \\ &= \sum_{i \in S_-} \exp(f_{t-1,m}(i, h_{t,m}(i)) + \alpha_{t,m} - f_{t-1,m}(i, y_i)) - \sum_{i \in S_-} \exp(f_{t-1,m}(i, h_{t,m}(i)) - f_{t-1,m}(i, y_i)) \\ &= (\exp(\alpha_{t,m}) - 1) \sum_{i \in S_-} \exp(f_{t-1,m}(i, h_{t,m}(i)) - f_{t-1,m}(i, y_i)) \\ &= (\exp(\alpha_{t,m}) - 1) \sum_{i \in S_-} \exp(\zeta_{t-1,m}(i, h_{t,m}(x_i))) \end{aligned}$$

3. For $i \in S_{-+}$:

$$\begin{aligned} \Delta_{-+} &= \sum_{i \in S_-} \exp(f_{t,m}(i, h_{t,m}(i)) - f_{t,m}(i, y_i)) - \sum_{i \in S_-} \exp(f_{t-1,m}(i, h_{t,m}(i)) - f_{t-1,m}(i, y_i)) \\ &= \sum_{i \in S_-} \exp(\zeta_{t,m}(i, h_{t,m}(x_i))) - \sum_{i \in S_-} \exp(\zeta_{t-1,m}(i, h_{t,m}(x_i))) \\ &= 0 \text{ since the value of } f_{t,m} \text{ does not change for these examples} \end{aligned}$$

So, the drop in loss $\varDelta = \varDelta_+ - \varDelta_- - \varDelta_{-+}$ is:

$$\begin{aligned}
&= (1 - \exp(-\alpha_{t,m})) \sum_{i \in S_+} L_{t-1,m}(i) - (\exp(\alpha_{t,m}) - 1) \sum_{i \in S_-} \exp(\zeta_{t-1,m}(i, h_{t,m}(i))) \\
&= \left(\tfrac{\exp(\alpha_{t,m}) - \exp(-\alpha_{t,m})}{2}\right) \left(\sum_{i \in S_+} L_{t-1,m}(i) - \sum_{i \in S_-} \exp(\zeta_{t-1,m}(i, h_{t,m}(i)))\right) \\
&\quad - \left(\tfrac{\exp(\alpha_{t,m}) + \exp(-\alpha_{t,m}) - 2}{2}\right) \left(\sum_{i \in S_+} L_{t-1,m}(i) + \sum_{i \in S_-} \exp(\zeta_{t-1,m}(i, h_{t,m}(i)))\right) \\
&\geq \left(\tfrac{\exp(\alpha_{t,m}) - \exp(-\alpha_{t,m})}{2}\right) \left(\sum_{i \in S_+} L_{t-1,m}(i) - \sum_{i \in S_- \cup S_{-+}} \exp(\zeta_{t-1,m}(i, h_{t,m}(i)))\right) \\
&\quad - \left(\tfrac{\exp(\alpha_{t,m}) + \exp(-\alpha_{t,m}) - 2}{2}\right) \left(\sum_{i \in S_+} L_{t-1,m}(i) + \sum_{i \in S_- \cup S_{-+}} \exp(\zeta_{t-1,m}(i, h_{t,m}(i)))\right)
\end{aligned}$$

Using the result we obtained in equation 2 and the fact that $\exp(\zeta_{t-1,m}(i, h_{t,m}(i))) \leq L_{t-1,m}(i)$, we can give a lower bound of the loss drop:

$$\begin{aligned}
\varDelta &\geq \left(\tfrac{\exp(\alpha_{t,m}) - \exp(-\alpha_{t,m})}{2}\right) \delta_{t,m} \sum_i L_{t-1,m}(i) \\
&\quad - \left(\tfrac{\exp(\alpha_{t,m}) + \exp(-\alpha_{t,m}) - 2}{2}\right) \left(\sum_{i \in S_+} L_{t-1,m}(i) + \sum_{i \in S_- \cup S_{-+}} L_{t-1,m}(i)\right) \\
&\geq \left(\tfrac{\exp(\alpha_{t,m}) - \exp(-\alpha_{t,m})}{2}\right) \delta_{t,m} \sum_i L_{t-1,m}(i) - \left(\tfrac{\exp(\alpha_{t,m}) + \exp(-\alpha_{t,m}) - 2}{2}\right) \sum_i L_{t-1,m}(i) \\
&\geq \left(\tfrac{\exp(\alpha_{t,m}) - \exp(-\alpha_{t,m})}{2} \delta_{t,m} - \tfrac{\exp(\alpha_{t,m}) + \exp(-\alpha_{t,m}) - 2}{2}\right) \sum_i L_{t-1,m}(i)
\end{aligned}$$

Hence the loss $1 - \varDelta$ at round t is at most a factor $1 - \frac{1}{2}(\exp(\alpha_{t,m}) - \exp(-\alpha_{t,m}))\delta_{t,m} + \frac{1}{2}(\exp(\alpha_{t,m}) + \exp(-\alpha_{t,m}) - 2)$ of the loss in round $t - 1$. □

We proved that, in each view, the training error (cost) decreases. Based on theorem 1, and tuning $\alpha_{t,m}$ to $\frac{1}{2} \ln \frac{1+\delta_{t,m}}{1-\delta_{t,m}}$, we get the following bound on the empirical error of the classifier H_m obtained by the weighted combination of weak classifiers learned in view m after T iterations:

$$\epsilon_{T,m} \leq (k-1) \prod_{t=1}^{T} \sqrt{1 - \delta_{t,m}} \leq (k-1) \exp\left\{-\frac{1}{2} \sum_{t=1}^{T} \delta_{t,m}^2\right\} \qquad (3)$$

This result shows that Mumbo is a boosting algorithm even when the selected weak classifier always comes from the same view m for all steps. This might occur when the major view is far better than minor views for all training examples.

3.2 Bounding the Whole Empirical Error

At each step t of the algorithm 1, one classifier is selected among m weak classifiers, if m is the number of views, that is, the space of weak hypothesis $\mathcal{H}$ in this case is $\{h_{t,1}, \cdots, h_{t,m}\}$. This space is a particular case of the space of hypothesis used by the OS algorithm, thus we obtain the same bound on the empirical error as the OS algorithm, that is :

$$\epsilon_T \leq (k-1)\prod_{t=1}^{T}\sqrt{1-\delta_{t,m}} \leq (k-1)\exp\left\{-\frac{1}{2}\sum_{t=1}^{T}\delta_{t,m}^2\right\} \tag{4}$$

In practice, one may observe that the edges of the classifiers at step t are all negative. In such a case, since each weak classifier of view v is trained on a subset of the learning samples randomly drawn from the current distribution of v, iterating the learning step until γ_v is positive allows the algorithm to fulfill the conditions.

3.3 Results in Generalization

We show here that the generalization error of the final hypothesis learned by Mumbo after T iterations can be bound, and this bound converges towards 0 with the number of iterations.

The generalization error of a classifier is defined as the probability to misclassify any new example. For multiclass algorithms such as AdaBoost.MR, [19] shows that the generalization error of the final hypothesis can be bound and that it is related to the margins of the learning examples. We thus first recall the definitions of the bound on the generalization error, then we extend existing results to Mumbo.

Generalization Error for Multiclass Problems. The final hypothesis of Mumbo is a multi class classifier, thus its output space can be defined as $Y = \{1, 2, ..., k\}$. In this section, the weak classifiers $h \in \mathcal{H}$ are defined as mappings from $X \times Y$ to $\{0, 1\}$, where X is some description space. The label y is predicted as a potential label for x_i if $h(x, y) = 1$. Note that these classifiers are equivalent to $\mathbb{1}[h_t(x) = l]$, the weak classifiers described in the algorithm.

Let $\mathcal{C}$ denote the convex hull of $\mathcal{H}$, that is :

$$\mathcal{C} = \left\{ f : (x,y) \rightarrow \sum_{h\in\mathcal{H}} \alpha_h h(x,y) | \alpha_h \geq 0 \ and \sum_{h} \alpha_h = 1 \right\}$$

For a given example x and a label y, a classifier f in $\mathcal{C}$ predicts y as the class of x if $\underset{l\in Y}{argmax} f(x,l) = y$. The margin of an example is then defined as :

$$margin(f,x,y) = f(x,y) - \underset{l\neq y}{max} f(x,l)$$

The function f misclassifies an example x if the margin given by f on the couple (x, y) is negative or zero.

Using the previous definitions, Schapire et al. give a proof of theorem 2 [19]:

Theorem 2 (Schapire *et al.*, [19]). *Let $\mathcal{D}$ be a distribution over $X \times Y$, and let S be a sample of n examples chosen independently at random according to*

$\mathcal{D}$. Assume that the base-classifier[2] *space $\mathcal{H}$ is finite, and let $\delta > 0$. Then with probability at least $1 - \delta$ over the random choice of the training set S, every function $f \in \mathcal{C}$ satisfies the following bound for all $\theta > 0$:*

$$\mathbf{P}_{\mathcal{D}}[margin(f,x,y) \leq 0] \leq \mathbf{P}_S[margin(f,x,y) \leq \theta] + O\left(\frac{1}{\sqrt{n}} + \left(\frac{\log(nk)\log(|\mathcal{H}|)}{\theta^2} + log(1/\delta)\right)^{1/2}\right)$$

More generally, for finite or infinite $\mathcal{H}$ with VC-dimension d, the following bound holds as well, assuming that $n \leq d \leq 1$:

$$\mathbf{P}_{\mathcal{D}}[margin(f,x,y) \leq 0] \leq \mathbf{P}_S[margin(f,x,y) \leq \theta] + O\left(\frac{1}{\sqrt{n}} + \left(\frac{d\ log^2(nk/d)}{\theta^2} + log(1/\delta)\right)^{1/2}\right)$$

In theorem 2, the term $\mathbf{P}_{\mathcal{D}}[margin(f,x,y) \leq 0]$ is the generalization error of the function f. The term $\mathbf{P}_S[margin(f,x,y) \leq \theta]$ is the empirical margin error of f on the sample S, that is, the proportion of examples of S which are misclassified, or which are correctly classified but with a margin smaller than θ. In the following section, we use $\epsilon_\theta(f,S)$ instead of $\mathbf{P}_S[margin(f,x,y) \leq \theta]$.

The second term in the theorem is a complexity penalization cost.

Mumbo. Theorem 2 holds for every voting method using multiclass classifiers as weak classifiers; it thus also holds for Mumbo since his final hypothesis is $H_T(x) = \underset{l \in 1,2,\ldots,k}{argmax}\ f_T(x,l)$, where :

$$f_T(x,l) = \left(\sum_{t=1}^{T} h_t(x,l)\alpha_t\right) / \sum_{t=1}^{T} \alpha_t$$

The weak classifier h_t chosen at each iteration is selected from a set of classifiers $\{h_{t,1}, ..., h_{t,m}\}$. These classifiers are selected from potentially different spaces of hypothesis, namely $\mathcal{H}_1, ..., \mathcal{H}_m$. Thus the space of hypothesis $\mathcal{H}$ from which h_t is selected is the union of $\mathcal{H}_1, ..., \mathcal{H}_m$. We deduce by the definition of the VC-dimension [20] that $d_{\mathcal{H}} = min\{d_{\mathcal{H}_1}, ..., d_{\mathcal{H}_m}\}$.

We still have to prove that the generalization error decreases with the number of iterations. To do so, it is sufficient to prove that the empirical margin error decreases, since the term $O\left(\frac{1}{\sqrt{n}} + \left(\frac{d\ log^2(nk/d)}{\theta^2} + log(1/\delta)\right)^{1/2}\right)$ is a constant.

We start with showing that we can find a bound for $\epsilon_\theta(f_T, S)$.

Lemma 1. *The empirical margin error of Mumbo after T iterations is bounded by:*

$$\epsilon_\theta(f_T, S) \leq \frac{(k-1)}{n}\left(\prod_{t=1}^{T}(1+\delta_t)^{\frac{1+\theta}{2}}(1-\delta_t)^{\frac{1-\theta}{2}}\right)$$

[2] Note : the base-classifiers are referred to as weak classifiers in our paper.

Proof

Let $l = \underset{y' \neq y}{argmax} f(x, y')$. For readability reasons, we may write $\sum\limits_t$ instead of $\sum\limits_{t=1}^{T}$.

By the definition of the margin and f, we get :

$$margin(f, x, y) = f(x, y) - f(x, l) = \frac{\sum\limits_t h_t(x, y)\alpha_t}{\sum\limits_t \alpha_t} - \frac{\sum\limits_t h_t(x, l)\alpha_t}{\sum\limits_t \alpha_t}$$

Hence,

$$\begin{aligned} margin(f, x, y) < \theta &\Leftrightarrow \frac{\sum\limits_t h_t(x,y)\alpha_t}{\sum\limits_t \alpha_t} - \frac{\sum\limits_t h_t(x,l)\alpha_t}{\sum\limits_t \alpha_t} \leq \theta \\ &\Leftrightarrow \theta \sum_t \alpha_t - \left(\sum_t h_t(x, y)\alpha_t - \sum_t h_t(x, l)\alpha_t \right) \geq 0 \end{aligned}$$

Let $A_i = -\left(\sum\limits_t \alpha_t h_t(x_i, y) - \sum\limits_t \alpha_t h_t(x_i, l) \right)$ and $B = \theta \sum\limits_t \alpha_t$. We deduce that $\mathbf{P}[margin(f, x_i, y) \leq \theta] = 1 \Leftrightarrow A_i + B \geq 0$, that is, $\exp(A_i + B) \geq \mathbf{P}[margin(f, x, y) \leq \theta]$. Thus, $\epsilon_\theta(f_T, S) \leq \frac{1}{n} \sum\limits_{i=1}^{n} \exp(A_i) \exp(B)$.

$$\begin{aligned} \epsilon_\theta(f_T, S) &\leq \tfrac{1}{n} \sum_{i=1}^{n} \exp(A_i) \exp(B) \\ &\leq \tfrac{1}{n} \sum_{i=1}^{n} \exp\left(-(\sum_t \alpha_t h_t(x_i, y) - \sum_t \alpha_t h_t(x_i, l)) \right) \exp(\theta \sum_t \alpha_t) \\ &\leq \tfrac{1}{n} \sum_{i=1}^{n} \exp\left(-(f_T(x_i, y) - f_T(x_i, l)) \right) \exp(\theta \sum_t \alpha_t) \\ &\leq \tfrac{1}{n} \sum_{i=1}^{n} \sum_{y' \neq y} \exp\left(f_T(x_i, y') - f_T(x_i, y) \right) \exp(\theta \sum_t \alpha_t) \end{aligned}$$

Using the bound on the empirical error, we deduce :

$$\begin{aligned} \epsilon_\theta(f_T, S) &\leq \tfrac{1}{n} \exp(\theta \sum_t \alpha_t)(k-1) \prod_t \sqrt{1 - \delta_t^2} \leq \tfrac{1}{n} \exp(\theta \sum_t \tfrac{1}{2} \ln(\tfrac{1+\delta_t}{1-\delta_t}))(k-1) \prod_t \sqrt{1 - \delta_t^2} \\ &\leq \tfrac{1}{n} \prod_t (\tfrac{1+\delta_t}{1-\delta_t})^{\frac{\theta}{2}} (k-1) \prod_t \sqrt{1 - \delta_t^2} \leq \tfrac{(k-1)}{n} \left(\prod_t (1 + \delta_t)^{\frac{1+\theta}{2}} (1 - \delta_t)^{\frac{1-\theta}{2}} \right) \end{aligned} \qquad \square$$

The lemma 1 gives a bound on the empirical margin error. As it was shown in [19], if $\theta < \delta_t/2$, then $(1 + \delta_t)^{\frac{1+\theta}{2}} (1 - \delta_t)^{\frac{1-\theta}{2}} < 1$. We thus finally claim that the generalization error decreases with the number of iterations:

Theorem 3. *Let $\theta > 0$ be a fixed margin, then the empirical margin error $\epsilon_\theta(f_T, S)$ converges towards 0 with the number of iterations, if the edge of the weak hypothesis selected at each iteration is $> 2\theta$.*

Theorem 3 and the bound given in theorem 2 together prove that *the generalization error of the final hypothesis of Mumbo* decreases with the number of iterations. Indeed, the second term of the bound in theorem 2 is a constant, since all the parameters, including $d_{\mathcal{H}}$, are constant in a given problem, and theorem 3 proves that the first term of the bound decreases with the number of iterations.

4 Experiments on Mumbo

In order to empirically validate and illustrate this approach of multiview learning with boosting, we mainly used synthetic data that obey the underlying assumptions of Mumbo. After explaining the used protocol, this section presents and discusses the results of experiments.

4.1 Protocols

Data Generation. Data is generated within 3 views, and clustered in two classes $\{-1, +1\}$. In each view, the descriptions of examples are vectors of real values. Examples of each class y in view v are generated along a gaussian distribution $\mathcal{G}[m_{y,v}, \sigma_{y,v}]$. However, in order to generate weak views, two types of noise disrupt the sample:

- in each view, the distributions of classes may overlap: some examples are likely to belong to both classes[3].
- In each view, some examples are generated using a uniform distribution, the same for both classes. Let η be the rate of such a description noise (η_M is the noise rate of the major view, while η_m is the noise rate of minor views).

One major view is generated. The two minor views are generated with $\eta_m = \frac{3-2\eta_M}{4}$ in such a way that half of the noisy examples in view M are likely to be sane in minor views. Figure 1 pictures an example of a learning sample with $n = |S| = 20$ examples per class.

We can associate the disruption amount (distribution overlap and noise on descriptions) with the edge-over-random capabilities of weak-classifiers. The more disruption we have in a view, the more γ_v is low on that view. Such a sample generation process was designed in order to fit the assumptions that lead to the design of Mumbo: views are rather weak, and learning a classifier on the whole sample needs a cooperation between learners on each view, because information may be distributed among views.

Processing Experiments. Each weak classifier on view v is obtained by training a linear SVM on a subsample of examples randomly drawn from the current distribution (cost matrix) of v. We check that each weak classifier trained on the view v complies with the definition of weak classifiers in the theoretical scheme of [15], using **B**=**U**. Results are the mean of 10 experiments: one experiment

[3] For these examples x, $P(y = +1|x) = P(y = -1|x)$).

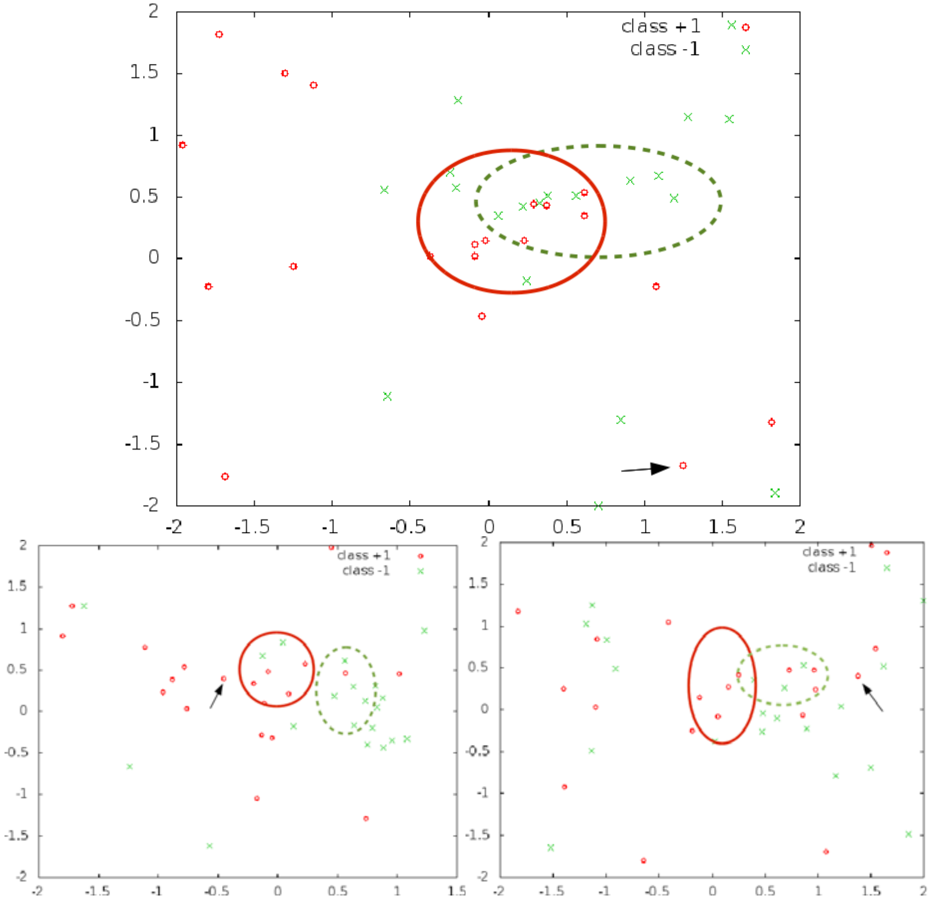

Fig. 1. Each example of the learning sample is represented under three views: the major view is on top, with $\eta_M = 0.38$; other views are minor (bottom), with $\eta_m = 0.56$. The ovals picture the parameters of the examples distribution within each class. The same example is pointed out in each view, in order to illustrate the distribution of information among views.

is made up of (1) the generation of learning and test samples, (2) the learning process, and (3) the evaluation process.

As said in the introduction, Mumbo was designed as an alternative way to fuse classifiers. We thus compare it with two basic methods of fusion, and with Adaboost: (1) late fusion SVM: one RBF SVM is trained on each view, and the final decision is a margin-weighted combination of their results; (2) early fusion SVM: descriptions of each example are concatenated, then a RBF SVM is trained on the single resulting view; and (3) early fusion Adaboost: descriptions of each example are concatenated, then Adaboost is trained on the single resulting view, with a RBF SVM on a subsample of examples as the weak learner.

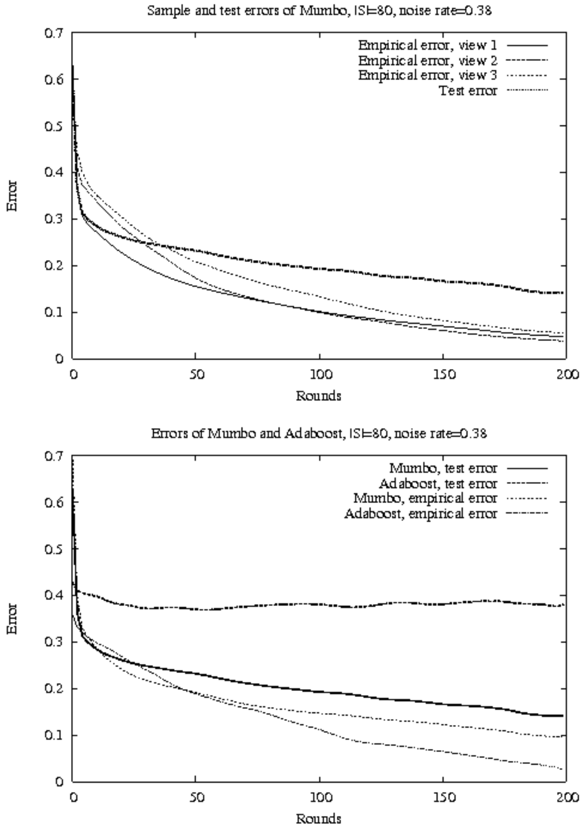

Fig. 2. Empirical and test errors of Mumbo (top), and Mumbo vs. Adaboost early fusion

Classifiers performances are computed using a testing sample drawn from the same setting that generated the learning sample, but twice bigger.

4.2 Results

We present here two kinds of results: an illustration of the behaviour of Mumbo, and a comparison of Mumbo with basic fusion approaches.

Illustration of Boosting Properties. Figure 2 reports, on the left, the boosting-like behaviour of Mumbo. As expected, the empirical costs on each view decrease with iterations, and the estimation of the generalization error also decreases. On the right, the figure pictures a first comparison of Mumbo with Adaboost (in an early fusion setting). We obtained this results with $n = |S| = 60$ and $\eta_M = 0.12$, but the same outlines of behaviours are observed whatever the parameters are ($|S|$ from 20 to 200, and η_M from 0 to 0.5).

Table 1. Comparison of Mumbo with early fusion and late fusion (base classifiers RBF SVM). Note that results of Adaboost are given after 200 iterations (like Mumbo): raw results show that Adaboost obtains slightly better results after about 50 iterations, then tends to over-fit.

$\|S\| = 80$, η_M	0.5	0.38	0.25	0.12	0
Early+SVM	0.390	0.410	0.437	0.396	0.389
SVM+Late	0.246	0.229	0.263	0.254	0.232
Early+Adaboost	0.415	0.420	0.403	0.364	0.358
Mumbo	0.148	0.152	0.168	0.174	0.164

$\|S\| = 120$, η_M	0.5	0.38	0.25	0.12	0
Early+SVM	0.367	0.382	0.396	0.389	0.343
SVM+Late	0.198	0.225	0.240	0.208	0.279
Early+Adaboost	0.425	0.415	0.466	0.411	0.389
Mumbo	0.02	0.036	0.012	0.026	0.020

The bad results of Adaboost are not surprising. Indeed, it processes examples on only one view that concatenates the smaller views. Since data was generated such that half of the disrupted examples on the major view are not disrupted in the minor views, the concatenation of descriptions leads to about 75% of noisy data. Adaboost is well-known to be sensitive to the noise, so one cannot expect better results, despite the true convergence of its empirical error.

In addition, which is not reported here, we observed that, whatever η_M is (always under 0.5), weak classifiers on minor views are selected in some rounds, in addition to the weak classifiers of the major view which are the most often selected. First rounds tend to only select the classifiers of the major view, then the minor views are alternatively selected with the major view. Besides, this behaviour can be observed on the first rounds on figure 2. It empirically shows that Mumbo actually encourages views to cooperate.

Comparison with Other Approaches. Table 1 compares Mumbo with basic early and late fusion approaches, with various values of η_M and various sizes of S. Late SVM is the best fusion approach with this type of data, which is not surprising since data is partially noisy (either in description or because distributions overlaps). Yet Mumbo is better for it processes the cooperation among views, leading each view to focus on the examples disrupted in other views.

However, the learning time of Mumbo is T times longer than the learning time of Late SVM. The collaboration slightly improves the results when the major view is disrupted. This is quite obvious with smaller learning samples (when $|S|$=15 or 30).

4.3 Discussion

As expected theoretically, the boosting usual behaviour is observed throughout the experiments, and the results of Mumbo are very good on synthetic data. These results validate the relevance of the Mumbo algorithm when cooperation among views is mandatory for obtaining a strong classifier. Results on empirical and generalization bounds of section 3 are also observed.

In further works, we should test Mumbo on UCI benchmarks. However, these benchmarks are not designed for multiview learning. We plan to select relevant views on these benchmarks (one major and several minor) using PCA or Canonical Component Analysis tools.

5 Related Works and Discussion

5.1 Related Works

So far, in the supervised setting, there is no multiview machine learning algorithm that considers the representation spaces as complementary. Early and late fusion-based approaches are only empirical ways to process the whole useful information available on samples.

The Multiple Kernel Learning (MKL) approaches [21], which may be used to process multiview samples, is then a costing way to rank the views. But yet, MKL does not promote the cooperation between views: it is much like a way to select the strongest views.

The closest approaches to Mumbo are co-training [5] and 2-boost [18]. The former is a multiview approach in the *semi-supervised setting*, where views iteratively cooperate for producing classifiers that converge to the same final hypothesis. The latter is a multiview boosting algorithm. However, Mumbo is different from co-training, first because it works in the supervised setting, and second because it does not assume that the classifiers actually must agree on the same examples. Indeed, Mumbo exploits the disagreements between views. Mumbo is thus closer to 2-Boost, although the motivations are not the same. 2-Boost is designed for dealing with one specific weak learner per view, in order to manage homogeneous views. Then, 2-Boost maintains only one global distribution of examples, whereas Mumbo maintains as many distributions as views in order to process cooperation.

Mumbo is an algorithm that may be categorized as an *ensemble of classifiers*, for the final classifier is a combination of other classifiers. In the literature, it was proved that without diversity between combined classifiers, the resulting classifier can not be better than the best of the combined classifiers. Many measures of diversity have then been studied so far [22]. We think that the hypothesis underlying Mumbo promote such a diversity. In that sense, we aim at obtaining some theoretical results between some diversity measures and classification accuracy of Mumbo.

5.2 Discussion and Improvements

In algorithm 1, the function $d_z(\cdot)$ indicates whether the update of the cost matrix is possible or not. It is a discrete $0-1$ function, which allows the update of the cost matrix only when some conditions are met (section 2.3). However, such an update rule might be too drastic for promoting the collaboration between views. We think that smoothing it, by changing its range to $[0,1]$, could improve the efficiency of the cooperation. Hence, in addition of having one or several views, each specialized in some parts of the description space, the minor views could be used more efficiently to increase the accuracy of the final predictions.

One of the main ideas of Mumbo is to enhance as much collaboration as possible between the views. We believe that it is possible to achieve a better cooperation also by changing the decision rule for h_t. Indeed, in algorithm 1, h_t is the best classifier among the m chosen classifiers at round t, *i.e.* the one that guarantees the best edge on the general cost matrix $G_{t,m}$. Many other ways to choose h_t could be studied, namely a combination of a subset of weak classifiers, or the choice between the weak classifier of the major view and a combination of the classifiers on minor views, etc. Hence, many alternate selections deserve to be studied, both theoretically (for example, in the PAC-Bayes framework [23]) and empirically.

The most urging work on Mumbo is its study on benchmark and real data. In some domains, such as image indexing, the views are quite natural: there exists dozens of image descriptors, either global or local, that could be considered as complementary views (texture vs. color, etc.). In many other domains, though, we must select the views according to the hypothesis underlying Mumbo (one major still weak view, and many minor views). We wish to adapt statistical tools for view selection, such as Principal Component Analysis as the simplest one.

6 Conclusion and Future Works

Mumbo is a boosting-like algorithm in the setting of multiview learning, where views are of different strenghts with regard to a classification task. The idea underlying Mumbo is to promote the cooperation between stronger and weaker views. To implement this idea, the originality of Mumbo is to maintain one distribution of examples per view, and to proceed to distribution updates that allow some views to focus on examples that are hard to classify in other views.

Mumbo is proved to be a boosting algorithm, within the new theoretical framework of [15]: the empirical error decreases with iterations, globally and within each view. Then, the generalization error of Mumbo is proved to be bounded. Finally, the experimental results on dedicated synthetic data give credits to the relevance of Mumbo for encouraging the cooperation among complementary views.

For now, Mumbo is a first attempt to tackle the problem of unbalanced views on data, and we expect to improve it both theoretically and through experiments on benchmarks and real data.

References

1. Masulli, F., Mitra, S.: Natural computing methods in bioinformatics: A survey. Information Fusion 10(3), 211–216 (2009)
2. Mansoorizadeh, M., Charkari, N.M.: Multimodal information fusion application to human emotion recognition from face and speech. Multimedia Tools and Applications 49(2), 277–297 (2010)
3. Culp, M., Michailidis, G., Johnson, K.: On multi-view learning with additive models. Annals of Applied Statistics 3, 292–318 (2009)

4. Sridharan, K., Kakade, S.M.: An information theoretic framework for multi-view learning. In: Annual Conference on Computational Learning Theory, pp. 403–414 (2008)
5. Blum, A.B., Mitchell, T.M.: Combining labeled and unlabeled data with co-training. In: 11th Annual Conference on Computational Learning Theory, pp. 92–100 (1998)
6. Christoudias, M.C., Urtasun, R., Kapoor, A., Darrell, T.: Co-training with noisy perceptual observations. In: Computer Vision and Pattern Recognition, pp. 2844–2851 (2009)
7. Sindhwani, V., Rosenberg, D.S.: An rkhs for multi-view learning and manifold co-regularization. In: International Conference on Machine Learning, pp. 976–983 (2008)
8. Christoudias, M.C., Urtasun, R., Darrell, T.: Multi-view learning in the presence of view disagreement. In: 24th Conference on Uncertainty in Artificial Intelligence, pp. 88–96 (2008)
9. Ambrym-Maillard, O., Vayatis, N.: Complexity versus agreement for many views: Co-regularization for multi-view semi-supervised learning. In: 20th International Conference on Algorithmic Learning Theory, pp. 232–246 (2009)
10. Saffari, A., Leistner, C., Godec, M., Bischof, H.: Robust multi-view boosting with priors. In: Proceedings of the 11th European conference on computer vision conference on Computer vision: Part III, ECCV 2010, Heraklion, Crete, Greece, pp. 776–789. Springer-Verlag, Berlin, Heidelberg (2010)
11. Kludas, J., Bruno, E., Marchand-Maillet, S.: Information fusion in multimedia information retrieval. In: Boujemaa, N., Detyniecki, M., Nürnberger, A. (eds.) AMR 2007. LNCS, vol. 4918, pp. 147–159. Springer, Heidelberg (2008)
12. Wozniak, M., Jackowski, K.: Some remarks on chosen methods of classifier fusion based on weighted voting. In: Corchado, E., Wu, X., Oja, E., Herrero, Á., Baruque, B. (eds.) HAIS 2009. LNCS, vol. 5572, pp. 541–548. Springer, Heidelberg (2009)
13. Snoek, C., Worring, M., Smeulders, A.: Early versus late fusion in semantic video analysis. In: Proceedings of the 13th Annual ACM International Conference on Multimedia, MULTIMEDIA 2005, pp. 399–402. ACM, New York (2005)
14. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning, pp. 148–156 (1996)
15. Mukherjee, I., Schapire, R.E.: A theory of multiclass boosting. In: Lafferty, J., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (eds.) Advances in Neural Information Processing Systems, vol. 23, pp. 1714–1722 (2010)
16. Goldwater, S., Jurafsky, D., Manning, C.D.: Which words are hard to recognize? prosodic, lexical, and disfluency factors that increase speech recognition error rates. Speech Communication 52, 181–200 (2010)
17. Hasegawa-Johnson, M., Chen, K., Cole, J., Borys, S., Kim, S.-S., Cohen, A., Zhang, T., Choi, J.-Y., Kim, H., Yoon, T., Chavara, S.: Simultaneous recognition of words and prosody in the boston university radio speech corpus. Speech Communication 46, 418–439 (2005)
18. Janodet, J.-C., Sebban, M., Suchier, H.-M.: Boosting classifiers built from different subsets of features. Fundam. Inf. 96, 89–109 (2009)
19. Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods. The annals of statistics 26(5), 1651–1686 (1998)
20. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience, Hoboken (1998)

21. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., El Ghaoui, L., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. J. Mach. Learn. Res. 5, 27–72 (2004)
22. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning 51(2), 181–207 (2003)
23. Germain, P., Lacasse, A., Laviolette, F., Marchand, M.: Pac-bayesian learning of linear classifiers. In: Proceedings of the 26th Annual International Conference of Machine Learning, pp. 353–360 (2009)