Université d'Aix-Marseille
Laboratoire d'Informatique Fondamentale de Marseille

# Thèse

présentée en première version en vu d'obtenir le grade de Docteur, spécialité "Informatique Fondamentale"

par

Sokol Koço

# Tackling the uneven views problem with cooperation based ensemble learning methods

Thèse à soutenir le 16 Décembre 2013 devant le jury composé de :

| | | | |
|---|---|---|---|
| M. | Marc Sebban | Université Jean Monnet de Saint-Etienne | (Rapporteur) |
| M. | François Yvon | Université Paris Sud | (Rapporteur) |
| M$^{me}$ | Géraldine Damnati | Orage Labs | (Examinateur) |
| M | Liva Ralaivola | Aix-Marseille Université | (Examinateur) |
| M | Nicolas Usunier | Université Technologique de Compiègne | (Examinateur) |
| M | Frédéric Béchet | Aix-Marseille Université | (Directeur) |
| M$^{me}$ | Cécile Capponi | Aix-Marseille Université | (Co-Directeur) |

*à Roanne / Roanës*

# Contents

# List of Figures

# List of Tables

# Introduction

> (Machine learning is the) Field of study that gives computers the ability to learn without being explicitly programmed.
>
> Arthur Lee Samuel, 1959

Artificial Intelligence (AI) is a field of computer science related to the development and study of machines and softwares capable of performing intelligent/humanlike actions. The branch of AI focused on the development, study and deployment of methods that can *learn* from data is called Machine Learning (ML). One of the major problems encountered in machine learning is that of *concept learning*, which consists in finding rules that divide the examples in two groups: object associated to a given concept and object not associated to that concept. Objects in the former case are called positives (+), since they belong to the concept, while the other examples are called negatives (-). In order to infer the rules, the learning agent is presented with both positive and negative examples. For example, in the well-known *SPAM* problem, the goal is to predict if a given e-mail is associated with the concept of *being a SPAM* [1]. Positive examples include e-mails containing ads, newsletters, etc, while the other e-mails (mail from the family, boss, university etc) are the negative ones.

Concept learning revolves around the idea of separating the objects in two groups: positives and negatives. In some cases, the negative instances belong to another common concept, that is, the goal is to distinguish between the two concepts classes (think of the case where the problem is to distinguish between apples and oranges). This is also known as binary learning and, when the number of concepts classes is greater than 2, we refer to it as multi-class learning. The space containing all the possible classes is called the output space, while the space containing all the object is called the input space. In order to be used in a learning procedure, objects are represented by sets of attributes or features, which are also called the descriptions or representations of the objects. For instance, in the SPAM problem, the descriptions consist of all the words contained in the e-mail, the hyperlinks, the mail addresses, various technical informations, and so on. We refer to the couple object and its label as an example, a sample or an instance. The process used by a learning agent in which a set of examples (also called a training set/sample) is used in order to derive a classification rule, is called supervised learning, since the agent can supervise its learning by checking if it gets the right classes for the training examples. In general, the training set consists only of a small part of the input space and the training agent is asked to produce rules that can be generalized to unseen objects: these rules are expected to correctly predict, with high probability, whether any instance is associated to a given concept or not. The rules can also be in the form of functions (mappings from the input space to the output space) and in this case, they are called hypothesis, classifiers or predictors.

1. Stupid Person's Annoying Mail

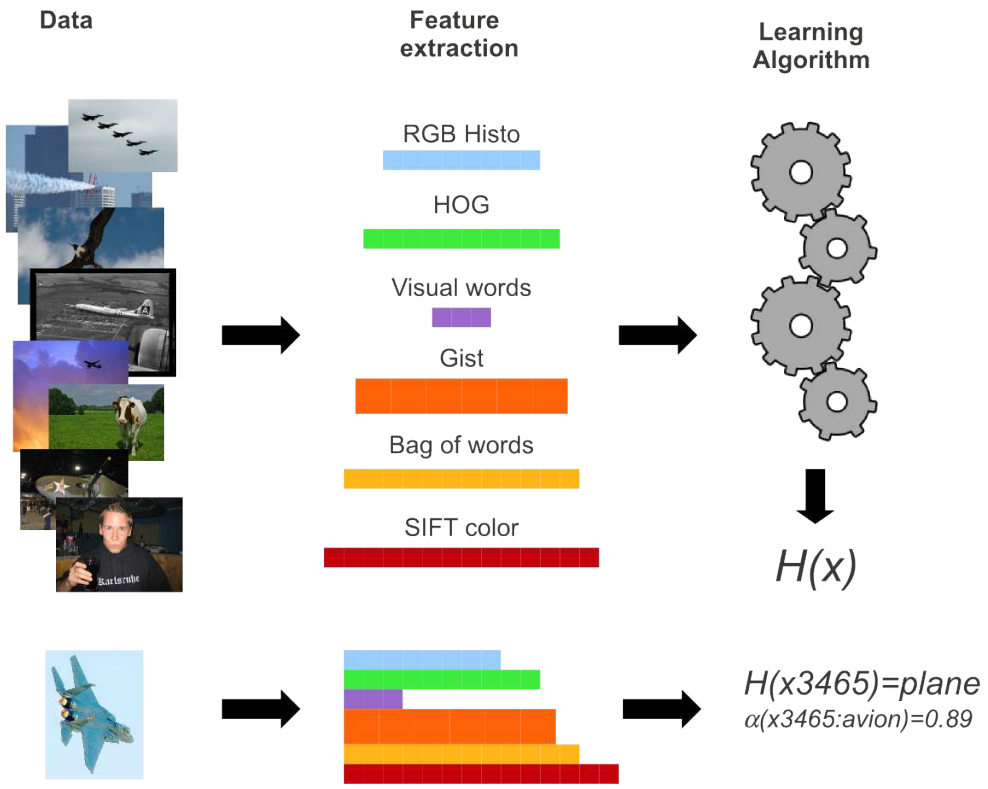



Figure 1 – *An example of machine learning problems: image classification.*

## A classification example

Let us consider the following problem (figure 1): the object are various images of persons, animals (cats, dogs, bears, etc), objects (tables, chairs, cars, etc), which make up the concepts, and, for simplicity sake, suppose that one image represents only one concept (either a person, or an animal, or an object, and so on). The goal consists in learning rules that correctly associate new unseen images to their corresponding concepts.

Now, there are different ways to describe an image (that is to define the sets of attributes/features) and we present here four of them. The simplest one is the *color histogram*, which corresponds to the distribution of the colors in the image, that is, for each color, we count the number of its occurrences in the image. The resulting description is a vector, whose size is the number of retained colors and the entries correspond to the occurrences of the colors. A second way to describe an image is by using the words/tags associated to it. Think, for example, of images obtained from various web pages; then the words contained in the page can be a good descriptor of what the image is about. This representation is also called *bag of words*. The third possible description is fairly similar to the bag of words, except that instead of words, we use patterns to describe an image. For instance, tigers have orange and black stripes, so images of tigers can be described using small images containing stripes. Finally, the last possible description we consider is the Histogram of Oriented Gradients (HOG), which counts occurrences of gradient orientation in localized portions of an image. This technique has been used with success for the purpose of object detection, since its output is the shape and outline of the objects present in an image.

Once the descriptions are available, they are fed to a learning algorithm, which

produces a classifier. The obtained classifier is then used to predicted the classes for new images. In figure 1, the new image is classified as *plane* with a probability of 89%.

The four aforementioned descriptions define four ways to *view* the examples. Although they are *views* of the same examples, they are by no means equivalent and do not contain the same information. The first (color histogram) and the second view (bag of word) contain global informations on the images: the former contains information on the distribution of the colors on the whole image, while for the latter, it is safe to assume that the words or tags refer more to the image as a whole, rather than to some isolated part. On the other hand, the third (bag of images) and the forth views (HOG) contain rather localized informations: the images in the bag of images are only related to a small part of the whole image, same for the gradients for HOG which are computed is small parts of the image. As such, the first two views can be quite informative if a class is identified by the global information of the image, such as a color proper to a class (white+polar bear), words describing exactly one class (desert+camel), etc. The other views are more informative in other cases, such as the aforementioned tiger example, where finding an orange and black stripes pattern may be sufficient; or using HOG in order to compare the shapes contained in an image, etc. These latter views concern localized patterns of the images.

Aside from the local versus global information, each view is more appropriate only for some tasks. If the task consists of predicting if an image represent a camel or a tuna fish, then the first view is well suited for this task since it is enough to check if the dominating color in the color histogram is the orange-yellow (camel) or the blue (tuna). At the same time, the other views can also be used to build good predictors for other concepts. The fourth view comes in handy when the task consists in distinguishing gazelles from lions, since their shapes are quite different. In this last example, the first view would not be so good, since in both cases the colors are nearly identical; same for the second and third view, since the same words/image patterns can be used to describe the images of both classes. These observations imply that each view can be used to build predictors that are capable of handling certain tasks, while miserably failing at others. Learning to efficiently combine the several classifiers learnt on the views is known as *multi-view learning*.

Building from the observations made on the image classification problem, the next part of this introductory chapter delves into the motivations behind the works presented in this thesis.

## Motivations

**The multi-view problem** Learning from multiple views has attracted a lot of attention in the last years due to both the number of increasing data and the number of methods employed to represent these data. In the image classification problem, the views defined for describing the images were primarily based on image processing techniques. Nowadays images are everywhere on the internet, from social media to news sites and passing through various web pages. Each of these mediums treats the images in a different way: social media regroup the images in albums, news sites use them to illustrate the articles, and so on. It follows that these representations of the images can be used as descriptions for the examples and thus increasing the number of available views. However, as previously mentioned, not all the views can be used to learn predictors that correctly recognize all the classes. This last remark rises up an important research problem:

How should the various classifiers be combined so that each *example* can be processed by the most appropriate view?

**The imbalanced problem** When dealing with problems, such as image classification, it is not unusual for some classes to be more represented than others. For example, it is easier to have images of cat and dogs, while it is harder to have images of whales or lions. In this cases, we say that the training set is *imbalanced*. The consequence of an *imbalanced* training set in the learning procedure is that if the method is not adapted for such cases, then it ends up training classifiers that correctly recognize the most represented classes, while shoving aside the least represented ones. It is thus necessary to come up with methods that take into account a possible imbalance in the representation of the different classes in the training sample. Which brings us to the main question for these methods:

What *training criterion* should the learning agent use in order to learn good predictors?

**The imbalanced multi-view problem** Looking back at the image classification example, the various views defined over the training sample are appropriate for different tasks, and more importantly they could be used to correctly recognize some of the classes. For instance, HOG can be used to correctly recognize classes such as giraffes, elephants, flamingos, etc, while the color histogram is more appropriate for polar bears, camels, etc. Another problem that may come up is that for certain examples only some of the views are available. Think of the case of automatic medical diagnostic, where for some patients only the results for a few tests are available. In such cases, it is possible for the learning sample to become imbalanced for a given view: some of the classes become more represented than the others. No matter the case, the views falling in this category bring up the same issues as in the imbalanced classes problem. The research problem associated to these methods is the following:

How should the various classifiers be combined so that each *class* can be processed by the most appropriate view?

## Outline of the thesis

The works presented in this thesis fall under the category of statistical learning theory. In order to facilitate the reading of this thesis and offer the reader a self-contained work, Chapter 1 formally introduces the notions of statistical learning and the tools and framework on which rely the contributions of this thesis. For each framework, a brief (non exhaustive) review of the state-of-the-art is proposed, including methods that have inspired this thesis. A first contribution in Chapter 1 concerns the definition of the *strength* of a view, which regroups the first and third question posed in the motivations into a single one: that of learning under uneven views.

In Chapter 2, we propose a multi-view method based on the boosting framework, called MuMBo, which attempts to address the first question given in the motivations. MuMBo is based on the assumption that a good strategy to build a classifier with good generalization properties would be to encourage some sort of cooperation between the views during the training process, so that each view can focus on what it does best and leave the other cases to more appropriate views.

Dealing with imbalanced classes in not a new problem in machine learning, and several methods have been proposed for tackling such problems. The main ap-

proaches consist in rebalancing the training set and modifying the learning methods in order to take into consideration the imbalance, (cf. Section 1.4). However, to the best of our knowledge, most imbalanced methods are based on heuristics and a theoretical approach is lacking. The aim of Chapter 3 is to propose a theoretical framework for the imbalanced problems through the study of the norm of the confusion matrix as an alternative optimization criterion, based on recent works by Ralaivola (2012) and Morvant et al. (2012). We show how to bound the norm of the confusion matrix and by doing so, we obtain a boosting method for the imbalanced problem, called CoMBo.

To address the third question posed in the motivations, we combine the imbalanced framework of Chapter 3 with the notion of cooperation between views of Chapter 2. The result is a series of optimization problems leading to several methods whose aim is to deal with imbalanced views. Chapter 4 summarizes the various approaches and introduces a multi-view version of CoMBo.

Throughout this thesis, the various methods are tested in two different settings: first they are tested on either artificial or UCI (Frank and Asuncion (2010)) datasets; then we give the empirical results obtained for the automatic classification of phone calls. This latter setting, obtained from the DECODA dataset (presented in Section 1.5), gives an insight on how the methods proposed in this thesis behave in the case of a real-world multi-view and imbalanced problem.

# Notations

| | |
|---|---|
| $S$ | training sample |
| $X$ | input (instance) space |
| $X_j$ | input (instance) space corresponding to view $j$ |
| $Y$ | output (classes) space |
| $\mathcal{D}$ | unknown distribution on $X \times Y$ |
| $(x, y)$ | a training example |
| $x$ | the description of a training example, $x \in X$ |
| $y$ | the class of the a training example, $y \in Y$ |
| $m$ | number of training samples |
| $m_l$ | number of training samples for a given class $l$ |
| $v$ | number of views |
| $K$ | number of classes (size of $Y$) |
| $X_1 \times X_2$ | Cartesian product of $X_1$ and $X_2$ |
| $\mathcal{H}$ | an ensemble of classifiers |
| $H(\cdot)$ | majority vote classifier, $H \in \mathcal{H}$ |
| $h(\cdot)$ | a classifier, $h \in \mathcal{H}$, $h : X \to Y$ |
| $\mathbf{A}$ | real-valued matrix, $\mathbf{A} \in \mathbb{R}^{p \times q}$, for some given $p, q \in \mathbb{N}$ |
| $\mathbf{A}(r)$ | the $l$-th row of matrix $\mathbf{A}$ |
| $\mathbf{A}(r, c)$, $\mathbf{a}_{r,c}$ | the entry of the $r$-th row and $c$-th coloumn of matrix $\mathbf{A}$ |
| $\lambda_{max}(\mathbf{A})$ | the largest eigenvalue of matrix $\mathbf{A}$ |
| $Tr(\mathbf{A})$ | the trace of matrix $\mathbf{A}$ |
| $\mathbf{A}^*$ | the conjugate transpose of matrix $\mathbf{A}$ |
| $\mathbf{A} \cdot \mathbf{B}$ | Frobenius (entry-wise) inner product of matrices $\mathbf{A}$ and $\mathbf{B}$ |
| $\mathbf{AB}$ | product of matrices $\mathbf{A}$ and $\mathbf{B}$ |
| $\mathbf{C}$ | the confusion matrix for a given classifier, $\mathbf{C} \in \mathbb{R}^{K \times K}$ |
| $\mathbf{D}$ | (in the boosting setting) the cost matrix, $\mathbf{D} \in \mathbb{R}^{m \times K}$ |
| $\mathbf{a}$ | real-valued vector, $\mathbf{a} \in \mathbb{R}^p$, for some $p \in \mathbb{R}$ |
| $\|\cdot\|_1$ | the $l1$-norm, both for vectors and matrices |
| $\|\cdot\|_2$ | the $l2$-norm, both for vectors and matrices |
| $\|\cdot\|$ | the operator norm, matrices only |
| $\|\cdot\|_F$ | the entry-wise Frobenius norm, matrices only |
| $\mathbb{I}(\cdot)$ | the indicator function |

# Preliminaries

# 1



In this introductory chapter we present the machine learning fields — and when needed, the frameworks — mostly related to the works presented in this thesis. The goal of the first part of this chapter is to give the reader an intuitive review of machine learning and to introduce the setting on which the present thesis focuses.

In the second part, we propose a short review of the boosting framework, an ensemble method which combines several *weak* classifiers into a *strong* one. For both the binary case and the multi-class case, several methods are presented, such as the AdaBoost family.

In the next part, we present a key framework for most of the methods proposed in this work: multi-view learning. Multi-view methods make use of several informations given by the different descriptions of the samples in order to learn better classifiers. As previously, a short review of multi-view learning methods is presented, ranging from fusion based ones to boosting based.

Finally, the last part concerns one of the most frequent problems in ML: the imbalanced classes problem, where one or a few classes, are more represented than the rest. The methods presented in this part are separated in two group: resampling methods, which increment (resp. decrement) the number of example of the minority classes (resp. majority classes), and re-weighting methods, which assign weight to the samples so that the majority class samples have small weight, while minority ones have larger weights.

## 1.1 A brief introduction to machine learning

The field of machine learning is commonly presented as an intersection of artificial intelligence, statistics and computer science. As opposed to the broader field of artificial intelligence, where the goal is to mimic human intelligent behavior, machine learning focuses on inferring rules from observed data that allow computers (and machines in general) to perform a certain task. Think for instance, of how humans can distinguish between "apples" from "oranges" only by observing some example of "apples" and "oranges". Amongst the different branches of machine learning, the most well-known is the supervised learning coupled with the problem of classification, which will also be the focus of this thesis.

In the supervised setting, the learning agent is presented with data (examples) that come in the form of input-output pairs. The space $X$ containing all the possible inputs is called the input (or instance) space; the ensemble of outputs $Y$ forms the classes space. In the case of binary classification, $Y$ contains two classes, usually noted $\{-1, 1\}$, while in multi-class classification, which is the case of this thesis, the size of $Y$ is $K \geq 2$, $Y = \{1, \ldots, K\}$. The training dataset, $S = \{(x_i, y_i)\}_{i=1}^{m}$, is made up of $m \in \mathbb{N}$ realizations of a pair of random variables

$$(x_i, y_i) \in X \times Y$$

drawn independently from $\mathcal{D}$, a fixed unknown joint probability distribution on $X \times Y$. The goal of a learning algorithm is to use the training data $S$ in order to find a mapping:

$$h : X \to Y.$$

The mapping $h$ associates an element of the input space to a class in the output space; $h$ is also called a classifier.

There are multiple mappings from $X$ to $Y$, and the learning algorithm should choose only one of them. In the case of supervised learning, the algorithm has access to both the descriptions and the classes for the examples in the training sample $S$. Intuitively the learning process can supervise the choice of the mapping by guiding it towards the one that makes the least mistakes on $S$. In order to quantify the *goodness* of a classifier, the notion of *loss functions* (denoted by $\ell$) was introduced, which measures how the prediction of a classifier fits the true class for a given example. The simplest loss function is the misclassification error, also known as the 0-1-loss, defined as follows:

$$\ell_{0-1}(y, h(x)) = \begin{cases} 1 & \text{if } h(x) \neq y, \\ 0 & \text{otherwise,} \end{cases}$$

where, $x \in X, y \in Y$, and $h$ is a mapping. Other possible losses can be defined, such as the exponential loss (Eq. 1.1), the logistic loss (Eq. 1.2), the hinge loss (Eq. 1.3), etc.

For simplicity sake, we give here the definitions for these losses in the case of binary classification.

$$\ell_{\exp}(y, h(x)) = \exp(-yh(x)) \tag{1.1}$$

$$\ell_{\log}(y, h(x)) = \log\Big(1 + \exp(-yh(x))\Big) \tag{1.2}$$

$$\ell_{hinge}(y, h(x)) = \max\{0, 1 - yh(x)\} \tag{1.3}$$

The previous losses are computed only for one example. In order to measure the goodness of a classifier on a training same, we define the *empirical risk* (also called empirical error) as follows:

$$\hat{\epsilon}(h, S) = \frac{1}{m}\sum_{i=1}^{m} \ell(y_i, h(x_i)).$$

The *true risk* of the classifier, measuring the probability of misclassifying an example by the classifier, is computed as the expectation of the loss:

$$\epsilon(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\ell(y, h(x)).$$

It follows that the goal of the learner is to find the mapping with the lowest risk; the risk for the *perfect classifier* is 0. That is, the best classifier that a supervised learning algorithm can output is the one verifying the following condition:

$$h^* = \operatorname*{argmin}_{h\in Y^X} \epsilon(h).$$

However, in general, the learning algorithm does not have access to all the possible mappings from $X$ to $Y$, but rather to a smaller subset $\mathcal{H} \subseteq Y^X$. The new goal of the learning process is to find the best possible classifier in $\mathcal{H}$ whose risk is minimal:

$$h^*_{\mathcal{H}} = \operatorname*{argmin}_{h\in\mathcal{H}} \epsilon(h).$$

Recall that $\epsilon(h)$ depends on $\mathcal{D}$ a quantity that is unknown, hence, finding $h^*_{\mathcal{H}}$ is difficult. To our rescue comes the *consistency* property, which states that the empirical risk $\hat{\epsilon}(h, S)$ converges to the true risk $\epsilon(h)$ with the number of instances in S, that is, the more training examples are available, the better the returned classifier is:

$$\lim_{m\to\infty} \hat{\epsilon}(h, S) - \epsilon(h) = 0.$$

This principle is also known as the Empirical Risk Minimization and it was introduced in the machine learning community by Valiant (1984) in the Probably Approximately Correct framework.

It follows that the goal of the learning procedure reduces to finding the classifier that minimizes the empirical risk, computed on the training set:

$$\hat{h} = \operatorname*{argmin}_{h\in\mathcal{H}} \hat{\epsilon}(h, S).$$

Multiple and various methods have been proposed for effectively minimizing the empirical risk, ranging from decision trees (Quinlan (1986)) favored for their interpretability, to neural networks and their mimic of the human brain, and passing through the success story that are the Support Vector Machines (Cortes and Vapnik (1995)). In the following section, we present the boosting framework, yet another approach for minimizing the empirical error, which on its own has reshaped the machine learning landscape. At the same time, this framework constitutes the foundations for most of the works in this thesis.

## 1.2 Boosting

The term *ensemble methods* denotes all learning methods that build predictive models from a set (ensemble) of classifiers. They have received a lot of interest in the past years and several frameworks and algorithms have been proposed. The main difference between these frameworks is related to how the set of classifiers is used during the learning phase. Some of the most well-known methods include late fusion (detailed in Section 1.3), random forests Breiman (2001), bagging Breiman (1996) and boosting. In this section, and throughout this thesis, we are only interested in the last framework: *boosting*.

### 1.2.1 A brief review of Boosting

**Definition 1** (Strong and weak learnability Schapire (1990)) *A concept class is learnable (or strongly learnable) if, given access to a source of examples of the unknown concept, the learner with high probability is able to output an hypothesis that is correct on all but an arbitrarily small fraction of the instances.*

*The concept class is weakly learnable if the learner can produce an hypothesis that performs only slightly better than random guessing.*

*Boosting* was first introduced in the statistical learning community in the form of a philosophical question related to *weak* and *strong* learnability. Kearns and Valiant (1989) posed as an open question whether there was a link — an equivalence, maybe — between *weak learnability* and *strong learnability*, and more precisely, if it was possible for weak learnability to achieve the same performances as strong learnability. The underlying idea was to establish whether it existed a way to *boost* a weak learning method into an arbitrary strong one. This question was also dubbed as the *hypothesis boosting problem*.

The answer to this question — which is *yes* — was first formally given in Schapire (1990). The proof consisted of a learning method for directly converting a weak learning method into a strong one, whose error is arbitrarily small. As a starting point for the method was the probably approximately correct (PAC) model introduced in Valiant (1984). In this model, the instances are chosen according to a fixed, but unknown, distribution and the task is to build a classification model that predicts, with high probability, the correct label for a given instance. The PAC model is also known as *distribution-free* model, since the distribution is unknown and the results given for this model do not depend on it. Schapire (1990) make this the cornerstone for their model: the distribution of the examples is modified in such a way that the weak learning algorithm is focused on the hard-to-classify examples. Depending on the distribution, some of the original training examples may be discarded, so that only the most informative ones are conserved. This *filtering* procedure, coupled with the recursive nature of the method, forces the weak learner to explore nearly the whole distribution. In order to take advantage of this exploration, the final classifier is an unweighted majority vote of all the classifiers learnt in the training phase, which is virtually limited to three.

A second proof of *boostability* was given in Freund (1995) and, once again, it was in the form of an *iterative* learning procedure, named *boost by majority* (BBM). Similarly to Schapire (1990), BBM manipulates the distribution of the examples in order to force the weak-learner to correctly classify all the examples, including the most difficult ones. However, instead of selecting only some of the examples, BBM assigns weights

**Algorithm 1:** AdaBoost Freund and Schapire (1996)

**Given**
— S = $\{(x_i, y_i)\}_{i=1}^m$ where $x_i \in X$, $y_i \in \{-1, +1\}$
— Weak learning algorithm $WL$
— $T$ the number of iterations
— $\forall i \in \{1, ..., m\}$ $\mathbf{D}_1(i) = \frac{1}{m}$
**for** $i = 1$ **to** $T$ **do**
Train $WL$ using $\mathbf{D}_t$ and get $h_t : X \to \{-1, +1\}$
Compute: $\epsilon_t = \sum_{i=1}^m \mathbf{D}_t(i)\mathbb{I}(h_t(i) = y_i)$, the empirical error
Compute *importance* coefficient: $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$
Update $\mathbf{D}_t$ :

$$\mathbf{D}_{t+1}(i) = \frac{\mathbf{D}_t(i) \exp(-\alpha_t h_t(i) y_i)}{Z_t}$$

where $Z_t = \sum_{i=1}^m \mathbf{D}_{t+1}(i)$ is a normalization coefficient.
**end for**
Output final hypothesis :

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

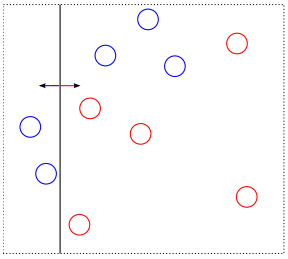
to all the examples, the harder an example is to classify, the higher the assigned weight. Freund (1995) showed also that BBM is optimal under some conditions.

### 1.2.2 An adaptive approach to boosting

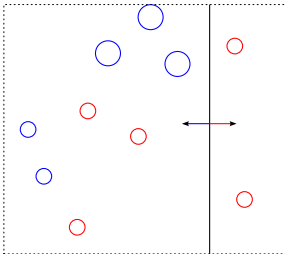
Both boosting methods presented in the previous section have been difficult to use in practice: they depend on hyper-parameters, thus making the learning procedure quite long. One of the major breakthroughs in boosting was the paper by Freund and Schapire (1995) and in particular the method they introduced, named AdaBoost. Following in the same path as the method proposed in Schapire (1990), AdaBoost manipulates the distribution over the training sample in order to identify hard-to-classify examples. This forces the weak learner to focus on those examples, thus promoting them during the training process and, hopefully, learning to correctly classify them.

Similarly to BBM, AdaBoost uses a *filtering* procedure, where a distribution over the training samples is maintained. However, contrary to BBM, which may discard some of the examples, AdaBoost updates the distribution so that the weight of the misclassified examples is increased, while the correctly classified ones have their weights decreased. The main advantage is that, at each iteration, all the examples are used in the learning phase, and the filtering effect is simulated by the weights assigned to the examples.

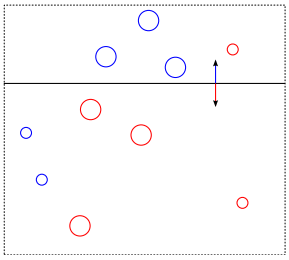
Let $S = \{(x_i, y_i)\}_{i=1}^m$ be a training sample, where $x_i \in X$ is the description and $y_i \in \{-1, +1\}$ is the class of the example. AdaBoost works as follows: first it assigns equal weights to all the training sample, since no prior knowledge on the distribution of the sample is supposed. Next a weak learning algorithm is called, where the inputs consist of the training sample and the weights associated to the examples. A weak learner $h : X \to \{-1, +1\}$ is returned and the only condition on $h$ is that it should



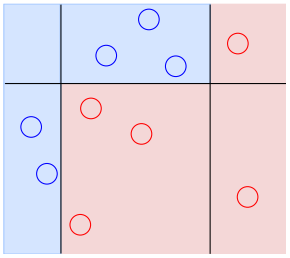
(a) $\epsilon_1 = 0.3, \alpha_1 = 0.42$



(b) $\epsilon_2 = 0.21, \alpha_2 = 0.66$



(c) $\epsilon_3 = 0.14, \alpha_3 = 0.91$



(d) *The final hypothesis*

Figure 1.1 – *An example of the run of AdaBoost on a dataset containing ten examples, five of class blue and five of class red. Three runs of the algorithm — corresponding to the top images — are presented and for each, the weighted error rate is given, as well as the importance coefficient α. The bottom image corresponds to the combined classifier, and the blue (resp. red) zones correspond to the regions were the examples are classified as blue (resp. red).*

perform better than random guessing. Since AdaBoost is a binary learning algorithm — there are only two classes, -1 and +1 —, the weighted empirical error of $h$ needs to be lower than 0.5.

Not all the classifiers returned by the weak learner are equal, some may have an error slightly lower than 0.5, while others may have a much lower error. In order to quantify this, a coefficient is computed for the classifier return by the weak learning algorithm; the better $h$ performs on the training sample, the higher its assigned coefficient. Finally, using the results of $h$ and the *importance* coefficient, the weights of the examples are updated: those of the misclassified examples are increased, while they are decreased for the correctly classified ones. As previously mentioned, this sort-of-filtering promotes the harder examples, thus forcing the weak learning algorithm to focus on them. Such a process is repeated for T rounds, and the final model is obtained by weighted majority voting of the T weak learners, where the weights correspond to the *importance* coefficients obtained during the training phase. The pseudo code of the algorithm is given in Algorithm 1.

An example[1] of how AdaBoost works is given in Figure 1.1, illustrating the effect of the update rule and the weighted majority vote. Since AdaBoost is a binary classification algorithm, we only consider two classes: blue class versus red class. The training sample, represented by circles, is made up of ten examples, five of each class. As for the weak learning algorithm, we only allow linear classifiers parallel to the axes.

Figure 1.1a corresponds to the first run of AdaBoost. As such, the weights of the examples, corresponding to the size of the circles, are all equal. The first classifier returned by the weak learning algorithm achieves an error of 0.3 — since it misclassifies three blue examples — and its *importance* coefficient is 0.42. Using these results, the

1. The example given here is taken from Schapire and Freund (2012).

weights of the examples are updated (as shown in Figure 1.1b): the three misclassified blue examples have their weights increased, while the weights of the other examples are decreased.

The second run of AdaBoost is given in Figure 1.1b. One may notice that the three misclassified examples in step 1 have been correctly classified, since the previous update of the weights gave more importance to these examples. As previously, the weighted error ($\epsilon = 0.21$) and the *importance* coefficient ($\alpha = 0.66$) are computed and the distribution in updated accordingly. A third run of AdaBoost (Figure 1.1c) gives a classifier with a weighted error of 0.14 and an *importance* coefficient of 0.91.

After the third run, the learning process is halted and the final hypothesis is computed as the weighted combination of the three classifiers, shown in algorithm 1. The colored zones in Figure 1.1d correspond to the predictions of the final hypothesis: the blue zones (resp. red zones) corresponds to where the final hypothesis predicts *blue* (resp. *red*). It is interesting to notice that the final hypothesis achieves a perfect score on our toy dataset, despite the fact that all the weak classifiers made mistakes during the training phase.

AdaBoost made a big impact in the learning community and it even made its way into the top 10 most influential algorithms in the data mining community Wu et al. (2007). Unfortunately, it is not the perfect learning algorithms and empirical evaluation quickly showed its shortcomings, especially when dealing with noisy data (for example Dietterich (2000)). Freund (2001) proposed BrownBoost, an adaptive version of BBM inspired from AdaBoost, which was more effective in dealing with noisy examples.

### 1.2.3 From binary to multi-class boosting

The version presented in Freund and Schapire (1995) was a binary learning algorithm and an extension for the multi-class setting — where the number of classes is more than two — was needed. The first version of multi-class AdaBoost, called AdaBoost.M1, was given in Freund and Schapire (1996) (and further polished in Freund and Schapire (1997)). AdaBoost.M1 is the most straightforward extension of AdaBoost to the multi-class framework, since the algorithm is pretty much the same. The main differences consisted in replacing AdaBoost's binary weak learners by multi-class learners and using the argmax operator instead of the sign one in the final hypothesis. The advantage of AdaBoost.M1 is that it inherits all of the good properties of AdaBoost. On the other hand, the main drawback resides in the fact that it uses the same weak learning condition, that is, it requires the weak classifiers to correctly classify more than 50% of the training set. In binary classification, this condition in closely linked to the error of random guessing. However the error of random guessing in the multi-class setting is not 0.5, and as such, this condition on the weak learner is restrictive. Freund and Schapire (1996) show that simple classifiers such as decision stumps, which often fail to meet this condition, can be effectively combined in stronger classifiers producing highly accurate predictions.

In order to overcome this difficulty, Freund and Schapire (1996) proposed a second version of AdaBoost for the multi-class setting, dubbed AdaBoost.M2 or AdaBoost.MR. Instead of using simple multi-class base classifiers, AdaBoost.M2 uses base classifiers that return a score in $[0, 1]$ for each couple example/class. That is, for each example, $K$ scores are returned, where $K$ is the number of classes, and the label predicted for the example is the one having the biggest score. Note that this is similar

to ranking (hence the R in AdaBoost.MR) the labels based on the returned scores and taking the top label as the predicted class [2].

Computing a score per couple example/label allows a better communication between the boosting procedure and the weak learner. In AdaBoost.M2, this is implemented in the construction of the training sample, where a new example is created for each couple example/label (different from the true class). So, if the original training sample contains $m$ examples, the new training sample is made up of $m(K-1)$ examples. The advantage of this extension of the training sample is that the (implicit) weak learning condition for AdaBoost.M2 is weaker than the one for AdaBoost.M1.

Another popular multi-class formulation of AdaBoost is AdaBoost.MH [3] proposed by Schapire and Singer (1999), which transforms the multi-class learning problem to binary decision problems using the one-versus-all approach. This way instead of learning one multi-class classifier, the goal is to train one classifier per class, which only predicts whether or not a class is associated to an example. The main advantage of this formulation is that it allows AdaBoost.MH to deal not only with multi-class problem, but also with multi-label ones. As for the weak learning condition, each of the per class weak classifiers need to perform better than random guessing, since they deal with binary classification problems.

Most of the previous multi-class algorithms suffer from the same problem: their weak learning condition is either too strong or not quite well defined. In order to overcome this and to unify the different algorithms, Mukherjee and Schapire (2011) proposed a framework suited for multi-class boosting. The main interest of this framework is the extension of the weak learning condition used in the binary case to the multi-class one, by replacing the distribution over the examples with a cost matrix. Here we recall the definitions of the cost matrix and the edge-over-random (eor) baseline, which can be interpreted as the random guessing in the multi-class setting.

**Definition 2** (cost matrix and edge-over-random baseline) *Mukherjee and Schapire (2011) The cost matrix* $\mathbf{D}$ *is a real-valued matrix of size* $m \times K$*, where each row* $\mathbf{D}(i)$ *corresponds to the example* $i$ *of the learning sample, and each entry* $\mathbf{D}(i,l)$ *corresponds to the cost of predicting class* $l$ *for the example* $i$*.*

*The edge-over-random cost matrix* $\mathbf{D} \in \mathbb{R}^{m \times K}$ *puts the least cost on the correct label, i.e. the rows of the cost matrix come from the set* $\left\{\mathbf{d} \in \mathbb{R}^K : \forall l, \mathbf{d}(y_i) \leq \mathbf{d}(l)\right\}$*.* $\mathcal{D}^{eor}$ *denotes the set containing all such matrices.*

*Let* $\gamma \geq 0$*. The edge-over-random baseline* $B \in \mathbb{R}^{m \times K}$ *is* $\gamma$ *more likely to predict the correct label 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* $l$*.* $\mathcal{B}_\gamma^{eor}$ *denotes the set containing all the edge-over-random baselines parametrized by* $\gamma$*.*

In the previous definition, if we consider $B$ as a classifier, then the parameter $\gamma$ defines how much better than random guessing $B$ performs. It plays the same role as the edge in the binary case AdaBoost (see Schapire (2002)). Therefore, from now on, we'll refer to it as the *edge*.

With the notations introduced in Definition 2, we can now state the weakest form of the weak learning condition as given in Mukherjee and Schapire (2011).

2. In the case of multi-label learning, it suffices to take the $k$ top label, where $k$ is the number of desired labels.

3. The H in AdaBoost.MH comes from the Hamming loss, which constitutes the minimization criterion for AdaBoost.MH.

**Definition 3** (edge-condition) *Mukherjee and Schapire (2011) Let $\mathcal{H}$ be a classifier space. Then "$\mathcal{H}$ is boostable" is equivalent to:*

$$\forall \mathbf{D} \in \mathcal{D}^{eor}, \exists h \in \mathcal{H} : \mathbf{D} \cdot \mathbf{1}_h \leq \max_{\mathbf{B} \in \mathcal{B}_\gamma^{eor}} \mathbf{D} \cdot \mathbf{B}, \tag{1.4}$$

*for some edge $\gamma > 0$ and where $\mathbf{1}_h$ is the prediction matrix defined as $\mathbf{1}_h(i,l) = 1$ if $h(i) = l$ and $\mathbf{1}_h(i,l) = 0$ if $h(i) \neq l$.*

Definition 3 implies that for every cost matrix in $\mathcal{D}^{eor}$, there exists a classifier in $\mathcal{H}$ that performs better than the *worst* baseline in $\mathcal{B}_\gamma^{eor}$. It ensures that the classifiers in $\mathcal{H}$ perform better than random guessing, without requiring them to be too strong.

This multi-class boosting framework is the base of some of the works presented in this thesis and the edge condition is a central part of most of the proofs. However, since most of the time we deal with *classifiers* instead of *classifier spaces*, we need to reformulate the edge condition (Equation 1.4) applied to a single classifier. This is done for convenience sake and in order to simplify future references to the edge condition.

**Definition 4** (simplified edge-condition) *Let $\mathbf{D} \subseteq \mathbb{R}^{m \times Q}$ and matrix $\mathbf{B} \in \mathcal{B}_\gamma^{eor}$, an eor-baseline, then a weak classifier $h$ satisfies the edge condition if :*

$$\mathbf{D} \cdot \mathbf{1}_h \leq \mathbf{D} \cdot \mathbf{B}, \tag{1.5}$$

*where $\mathbf{1}_h$ is the prediction matrix defined as $\mathbf{1}_h(i,l) = 1$ if $h(i) = l$ and $\mathbf{1}_h(i,l) = 0$ if $h(i) \neq l$.*

In the proofs provided throughout this thesis, we use a special case of edge-over-random baseline, which is defined as follows:

**Definition 5** *The uniform edge-over-random baseline is a cost matrix $\mathbf{U}_\gamma$ so that: $\mathbf{U}_\gamma(i,l) = 1/2 + \gamma/2$ if $l = y_i$ and $\mathbf{U}_\gamma(i,l) = 1/2 - \gamma/2$ if $l \neq y_i$.*

Theorem 20 of Mukherjee and Schapire (2011) shows that under some specific conditions — one of them related to the choice of the exponential loss as a minimization criterion —, the edge condition given in Equation 1.4 is equivalent to the same condition when the worst baseline is replaced by the uniform edge-over-random baseline. This implies that these two conditions can be used interchangeably, that is, it is possible to use the fixed uniform edge-over-random baseline instead of the (variable) worst baseline. A fixed matrix is easier to deal with, and as such, when needed, we'll be using the uniform edge-over-random baseline instead of the worst baseline.

This boosting framework is a well founded multi-class framework. Its main advantage is that it allows the use of classifiers weaker than the ones required by the other methods. Suppose that, under some conditions, only a (small) subset of the original training set is *interesting* or contains interesting informations; suppose also (for simplification sake) that these examples are located in an isolated part of the input space: then using weaker base learners allows the boosting procedure to process these examples earlier that using a strong learner, since in the latter case the weak learner is required to correctly classify more examples, thus lowering the chances of taking in account the isolated examples. This is the main motivation behind the choice of this multi-class boosting framework as the base for most of the works presented in this paper.

### 1.2.4 Other boosting methods

Even though most of this section was dedicated to the AdaBoost family methods, *boosting* is not limited only to adaptive boosting. Several boosting methods have been proposed for dealing with noisy data, which was one of the shortcomings of AdaBoost. Some of these methods include the aforementioned by BrownBoost Freund (2001), GentleBoost by Friedman et al. (2000) and LogitBoost by Friedman et al. (2000), the method presented by Nock and Lefaucheur (2002) and a recent method by Freund (2009). A meta algorithm, called AnyBoost, regrouping all boosting methods based on the optimization of convex loss functions, was proposed by Mason et al. (1999).

Other works have been focused on applying boosting to other settings, such as GradientBoost Friedman (2001) for dealing with regression problems, RankBoost Freund et al. (2003) for ranking problems, CoBoosting Collins and Singer (1999) for the semi-supervised setting, and so on. Linear Programming Boosting Demiriz et al. (2002), also called LPBoost, tackles the boosting problem in a different way: instead of iteratively learning the base classifiers, LPBoost uses linear programming in order to find the best combination for a given set of classifiers.

In the multi-class setting — aside from the aforementioned extensions of AdaBoost —one of the most interesting approaches is SAMME Zhu et al. (2009), which is similar to AdaBoost.M1. The advantage of SAMME is that it requires the weak learners to perform slightly better than $1/K$, while the classifiers of AdaBoost.M1 needed to perform better than 0.5. However, Mukherjee and Schapire (2011) showed that in their framework, even this condition was too strong and that it is possible to build *boostable* classifiers that do not meet the condition.

## 1.3 Learning with multiple views

The term *multi-view learning* refers to all machine learning methods that make use of several descriptions of the data in order to build a prediction model or a classifier. Generally, these methods are built on the hypothesis that each description of the data contains information that are not embedded in the other descriptions. The common goal of multi-view methods is use all the available information in the different descriptions in order to improve as much as possible the performances of the classifier.

This section proposes a simple review of multi-view methods. The first part introduces the notions of *view*, *strong* and *weak views*, as used throughout this thesis. Next we present the fusion-based approaches, which are perhaps the most common approaches in multi-view learning. The final part of this section is dedicated to multi-view methods which are not necessarily related to the fusion-based ones.

### 1.3.1 General notions on the views

In the literature, the term *multi-view learning* is used to denote:

a) methods that make use of several representations of the training examples in order to learn a predictive model;
b) the fact of observing an object from several viewpoints.

The latter is more frequently encountered in face detection/identification problems, where multiple views of the face are available, such as the frontal view, the lateral

ones, and so on (see, for example, Jones and Viola (2003)). Likewise, the description of an example is called *view* (Janodet et al. (2009)) or *modality* de Sa and Ballard (1998).

In this thesis, the term *view* denotes a *description* or *representation* of an example, that is, a set of attributes representing an example. For instance, the RGB histogram of an image defines a view, same as the SIFT vector Lowe (1999). Building on this definition, the term *multi-view* is used to denote all learning methods that take as inputs multiple views defined on the learning examples. That is, the input space $X$ is defined as the cartesian product of several input spaces $X = X_1 \times \ldots \times X_v$.

Since a view is defined as the description of an example, or rather a set of attributes representing the example, it follows that it embeds some information on the examples proper to the description. More importantly, the sort of information and its quantity may be different from one view to another, depending on the attributes that make up the views. For example, the RGB histogram view contains global information related to the colors contained in an image, while SIFT contains more local information related to the points of interest in an image. As a second example, in text categorization, the lexical view made of the different words of the text, is quite informative of the keywords, while the information contained in the syntactic view is closely related to the structure of the sentences. A direct consequence is that the classifiers obtained from the different views may not be equal performance wise, even though they are learnt on representations of the same examples.

Based on this observation, we distinguish three kinds of views: *weak views*, *strong views* and *imbalanced views*. These three types of views constitute key notions for this thesis, since they are part of the main hypothesis made by the methods presented here.

The notions of *weak* and *strong* view are closely related to the notions *weak* and *strong* classifier as defined in the PAC setting Valiant (1984). We define a strong view as an input space that allows to learn a classifier whose error is arbitrarily small. On the other hand, a weak view is defined as an input space that defines classifiers that perform only slightly better than random guessing. In other words, the notion of a *strong* view is associated with the possibility to learn a good classifier on that view, while the notion of *weak* view reflects the impossibility of learning such a classifier from the instance space defined by the view.

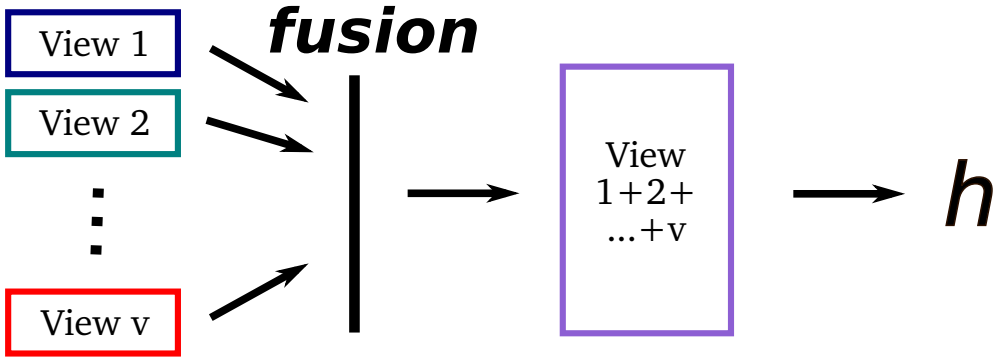
More formally, let $\mathcal{D}$ be a distribution defined on $X \times Y$. The best possible classifier that can be trained for samples drawn from $\mathcal{D}$ is the Bayes classifier, defined as:

$$h_{bayes}(x) = \underset{y \in Y}{argmin}\, \mathbb{P}(\mathcal{X} = y | \mathcal{Y} = x),$$

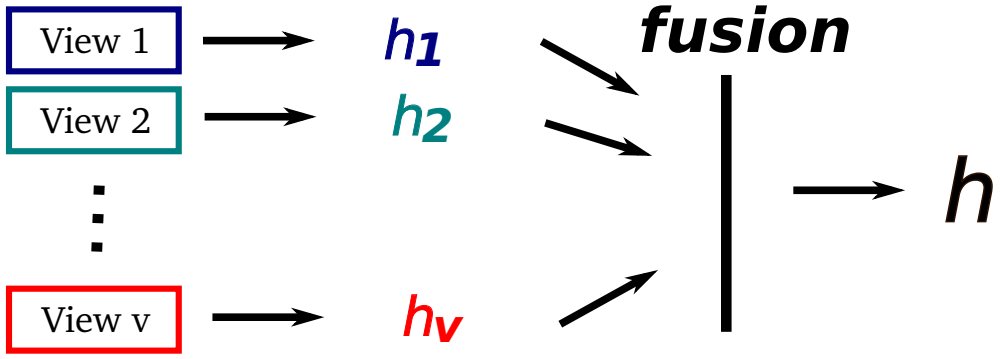
where $\mathcal{X}$ and $\mathcal{Y}$ are random variables taking values in $X$ and $Y$ respectively. Then, for a fixed $Y$, the input space $X$ defines a strong view if the Bayes classifier's error is small and $X$ defines a weak view if the error is close to the error of random guessing. $\gamma = 1 - \epsilon(h_{Bayes})$ gives a measure for the strength of the view defined by $X$.

Both notions are based on the performances of the classifiers learnt on the input space as a whole, independently of how it recognizes each of the classes that make up the output space.

The third type of view, *imbalanced* views, takes into consideration the impact that a classifier learnt on a view has on the different elements of the output space. In order to illustrate the idea behind this type of view, let us reconsider the image classification problem and the RGB histogram view constructed from the colors contained in an image. If the classification task consists in identifying if the image represents lions, gazelles or gorillas, then this view is informative mainly for the gorilla class, since it can be used to build classifiers that correctly identify that particular class.




(a) *Schema representing early fusion methods*




(b) *Schema representing late fusion methods*

Figure 1.2 – *Schemas for early and late fusion for two views.*

We define the notion of *imbalanced view* as the input space that allows to build classifiers that correctly recognize one class or a subset of classes. One may notice that this definition is related to the imbalanced classes problem, where classifiers fail to recognize all of the classes. This link will be investigate later on.

### 1.3.2 Multi-view learning through fusion-based methods

Fusion-based methods are amidst the first methods used when dealing with multi-view problems. They are usually divided in two major groups: *early fusion* methods and *late fusion* methods. The former regroups all the methods that perform some sort of fusion of the input spaces and then train a model on the unique modified input space. The latter often fall in the ensemble methods group, that is, they usually train as many classifiers as there are views (one classifier per view) and then perform some sort of fusion on the output space (the predictions of the classifiers). We now detail both approaches.

**Early fusion methods** The principal idea behind early fusion methods is to *combine* the different views in an unique view, which is then used to train a classifier. More formally, let $X_1, \ldots, X_v$ be $v$ views. Then the goal of early fusion methods is to construct/define a fusion operator $\odot$ so that $\odot : X_1, \ldots, X_v \rightarrow X$, that is, it builds an unique description space from the several description spaces it takes as parameters. The second step consist of learning a classifier $h : \odot(X_1, ..., X_v) \rightarrow Y$, where $Y$ is the ensemble of classes. A simple schema representing the early fusion where $v$ views are considered is given in Figure 1.2a. As such, these methods differ one from the other form *how* they combine the views. The three main types of *combinations* or *fusions* are: concatenation of the views, dimensionality reduction and feature selection.

As the name suggests, the *concatenation of the views* consists in aligning all the available descriptions one after the other. In this case, the fusion operator is defined as $\odot : X_1, \ldots, X_v \rightarrow X_1 \times \ldots \times X_v$. The advantage of the concatenation approach is that by augmenting the dimension of the description space, the examples might become separable or easier to separate than in each one of the views. However there are several inconveniences in this approach and we'll only list three of them. First, if the views contain noisy attributes or attributes with little information, then the concatenation might promote those attributes and thus implying little to no improvement in the performances of the classifier $h$. Secondly, if the views are represented by real-valued vectors, then they need to be normalized, otherwise views with higher values will take the upper-hand on the others. Last but not least, increasing the size of the description space, might heavily impact the learning time for $h$, thus making the training process long and tedious.

*Dimensionality reduction* refers to all those methods that modify the input space — usually made up of the concatenation of all the available views — in order to reduce the number of attributes in the final input space. The latter is not necessarily a subset of the concatenation of the views. That is, the goal is to combine several attributes into a single one, or replace whole subsets of attributes with one meta attribute. The fusion operator is defined as $\odot : X_1, \ldots, X_v \rightarrow X$, where $X$ is an input space proper to the method. This group of methods includes all methods that perform dimensionality reduction such as Principal Component Analysis (PCA) (performed on the concatenation of the views)[4], Fischer Discriminant Analysis (FDA) Diethe et al. (2008), Canonical Correlation Analysis (CCA) Foster et al. (2008), etc. The main advantage of these methods is that similar attributes or noisy attributes are regrouped and replaced after the *reduction* phase, thus minimizing their impact in the training process. However, since attributes contained in different views are potentially grouped together, the structures of the views are altered, which may result in loss of information proper to the views. A second drawback for these methods, is that usually it is not easy to associate a meaning to the newly created attributes, since they may represent heterogeneous and unreadable sets of attributes.

The third type of *fusion*, *feature selection*, is a particular case of *dimensionality reduction*, where a set of attributes is replaced by an attribute contained in one of the views. The goal of these methods is to select, from all available attributes, the minimal set of attributes (features) that keeps, or improves, the information embedded in the original set. The fusion operator is defined as $\odot : X_1, \ldots, X_m \rightarrow X \subseteq X_1 \cup \ldots \cup X_m$. Feature selection has been widely studied in different domains, such as text categorization Yang and Pedersen (1997) (for mono-view cases), image classification Feng et al. (2013), social media Tang et al. (2013), etc. The main advantage is that is speeds up the training time, while being more interpretable than *dimensionality reduction* methods. The main inconvenience of feature selection is that it might loose information, since the final set of attributes is just an approximation of the original.

It is interesting to notice that early fusion methods have been (implicitly) used in learning processes even when the multi-view aspect of the data was not the main concern. Let us take as an example the *arrhythmia*[5] dataset from the UCI Machine Learning Repository Frank and Asuncion (2010). The goal for this particular dataset is to predict if a patient is affected from one type of arrhythmia among 16. The dataset contains 452 examples, each one described by at most 279 attributes (some of the val-

4. A comparative review for dimensionality reduction methods in the mono-view case is given in van der Maaten et al. (2008).

5. http://archive.ics.uci.edu/ml/datasets/Arrhythmia

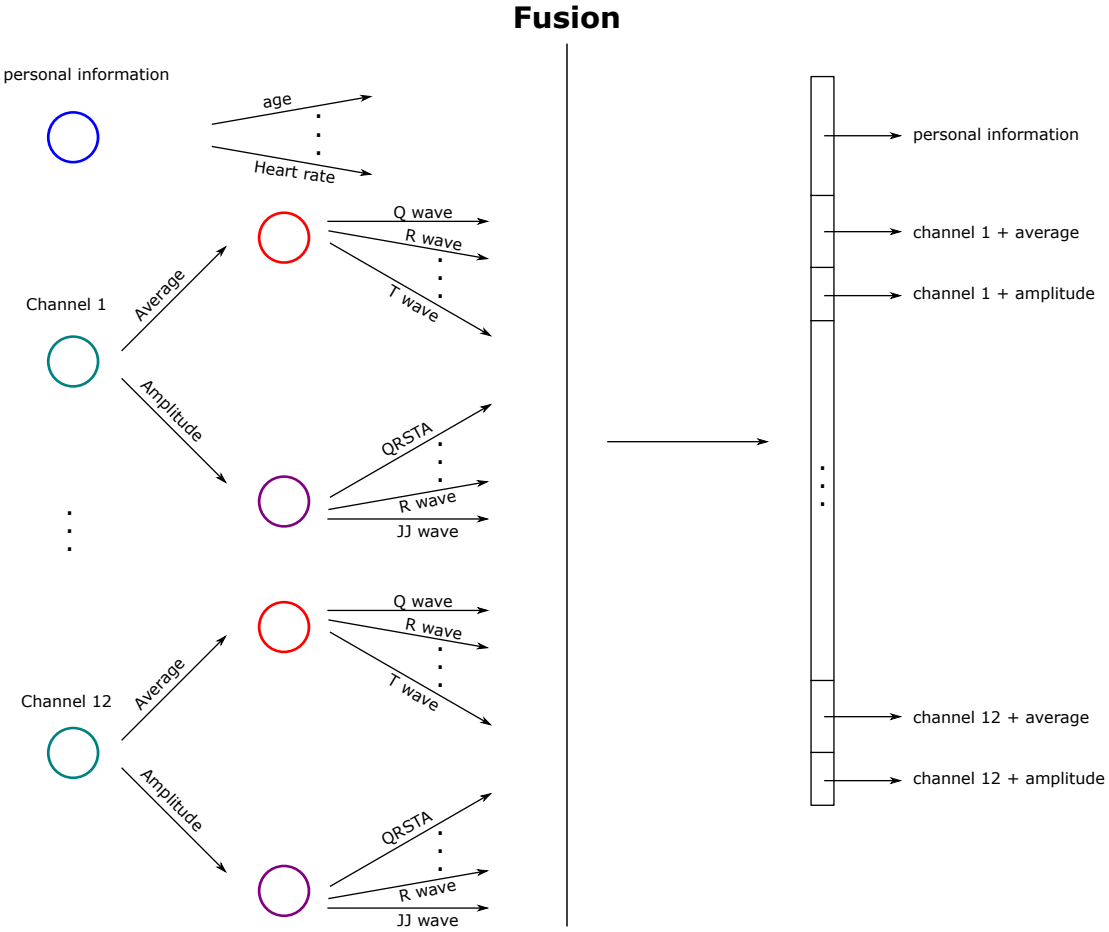



Figure 1.3 – *Description of the Arrhythmia dataset.*

ues are unknown for some examples).Figure 1.3 gives an insight on how the various attributes are obtained.

The first 15 attributes contain personal information on the patients, such as age, sex, weight and height. The other 264 attributes are averages and amplitudes of various types of waves computed from 12 different channels. Based on our definition of *view* (given in Section 1.3.1), the attributes can be split in several views. For instance, if we consider that each pair channel-measure defines a view, then arrhythmia is made up of 25 views: 1 containing the personal information, 12 representing the couples (channel, average) and 12 representing the (channel, amplitude) pairs. We can also regroup the attributes in 3 views (personal information, all averages, all means) and 13 views (personal information, channel 1 to 12). Whatever the number of views, *arrhythmia* can be seen as a multi-view sample obtained through an early fusion process — in this case, it is a simple concatenation of all the views — applied on the several views

**Late fusion methods** Late fusion methods apply the *fusion* on the output space, rather than on the input space, which is the case of early fusion methods. More formally, let $X_1, \ldots, X_v$ be $v$ views and, for the sake of simplicity and without loss of generality, we consider one classifier $h_j : X_j \to Y$ for each view $j \in \{1, \ldots, v\}$, for a total of $v$ classifiers. The goal of late fusion methods is to construct/define a fusion operator $\circledast$ so that $\circledast : Y, \ldots, Y \to Y$, returns a class in $Y$ from the combination of several predictions. For the $h_{j \in \{1,\ldots,v\}}$ and an example $x = (x_1, \ldots, x_v)$ described

by $v$ views, the operator $\circledast(h_1(x_1), \ldots, h_m(x_m)) \in Y$ defines the final prediction for $x$ through the combination of the predictions returned each of the classifiers $h_j$. A schema for the late fusion is given in Figure 1.2b where $v$ views are taken into consideration.

Generally, late fusion methods are better adapted for exploiting the individuality of each view, than early fusion methods. Indeed, since these methods train one classifier per view, then adapting the classifier to the corresponding view seems as a good strategy in order to make the best of the information embedded in the view. This means that different kinds of training procedures can be applied as long as the output spaces of the returned classifiers stay the same. While this is certainly the first advantage of late fusion methods, it constitutes at the same time the first question to be answered by all such methods: *how to train the various classifier on their corresponding views?* or simply put *what criterion should be minimized by the learning methods?* The most usual criterion is the empirical risk: for a given view, the chosen classifier should be the one that minimizes the empirical error, computed on a given training set. This does seem a convenient choice since most methods' goal is to minimize the risk of misclassifying the examples. Thus achieving a small risk for each of the classifiers will hopefully guarantee a small risk for the combination.

A second advantage of these methods is the fusion operator which can be used to embed different prior information, such as giving the highest priority to *strong* views, or minimizing the role played by the classifiers learnt of the *weak(er)* views. Due to their nature, late fusion methods usually fall in the category of ensemble learning methods (ELM). As such most approaches developed for the combination of the classifiers in ELM can be applied to these methods. Perhaps the most well-known fusion approach is the weighted majority, where a weight is associated to each classifier and the final prediction is the class that obtains the highest score. In this case the fusion operator $\circledast$ associated to the classifiers $h_1, \ldots, h_m$ and applied to an example $x = (x_1, \ldots, x_m)$ is defined as:

$$\circledast(h_1(x_1), \ldots, h_m(x_m)) = \underset{y \in Y}{\operatorname{argmax}} \sum_{j=1}^{m} \mathbb{I}(h_j(x_j) = y)\alpha_j,$$

where $\alpha_j$ is the weight associated to the classifier $h_j$ and $Y$ is the output space containing the classes. The simple majority vote is obtained when the weights $\alpha_j$ have the same value for all $j \in \{1, \ldots, m\}$ (usually this value is set to 1). A review on the different combining rules is given in Kittler et al. (1998).

This brings us to the second question *late fusion* methods should answer: *how to compute the weighting coefficients?*. The easiest way is to consider the simple majority vote, as previously defined, but it might not be the ideal choice since it gives the same importance to all the classifiers. Learning the coefficients can be performed through different way: through an optimization problem where the solution defines the optimal value for the considered criterion Awais et al. (2011); using a PAC-Bayesian approach Laviolette et al. (2011) (in the mono-view case) or learning them in an adaptive way Opitz and Shavlik (1996).

Answering both question can be quite challenging, since they both depend on different criterions and have different objectives.

The most representative candidate for the class of late fusion methods is the Multiple Kernel Learning (MKL) approach Lanckriet et al. (2004). MKL propose to

learn one kernel per view — and thus one classifier per view — while at the same time learning the optimal combination for the kernels, according to the objective of the classifiers. A kernel can be seen as a projection of the training examples from their input space into the one defined by the kernel. Hence, the idea behind learning a kernel per view is to make the most out of all the information contained in a view, so that an input space where the example are separable, can be found. This can be viewed as an answer to the first question. The second step, which consists of learning the weights for the kernels, is performed through an optimization procedure. The resulting solution minimizes the optimization criterion which is the minimization of the empirical error, while at the same time being an answer to the second question. However, MKL's learning procedure can be quite long since both steps are time consuming.

Early and late fusion methods propose a solution to the multi-view learning problem. Both of them have advantages and disadvantages, and an empirical comparison of these methods is given in Snoek et al. (2005).

### 1.3.3 Other multi-view methods

The methods presented in Section 1.3.2 were mainly given for the supervised setting — albeit some early fusion methods can be used for unsupervised learning. However multi-view learning became a topic of interest mainly thanks to the work by Blum and Mitchell (1998) in the semi-supervised setting. In their paper, Blum and Mitchell proposed a method, named co-training, which makes use of the *cooperation* between the different views in order to learn good classifiers for classifying unlabeled data. Co-training is based on the hypothesis that the examples are described by two views and both views are *sufficient*, that is, given a sufficient number of examples, each of the views can be used to learn a good classifier[6]. It was the first method to rely upon the cooperation between the views in order to improve the performances of the classifier. However the number of views was limited to two and it was based on the far too strong *sufficiency* hypothesis. Improvements and variations of co-training have been proposed, such as the works by Balcan et al. (2005), Yu et al. (2011), Tang et al. (2007), Goldman and Zhou (2000), Sindhwani and Niyogi (2005), etc.

The interesting idea of the cooperation between the views was also used in the active learning setting. Similar to semi-supervised learning, in the active learning setting the goal is to label a (large) set of unlabeled example, however here we allow the learning algorithm to interact with a human factor or oracle, which is asked to label *interesting* examples. Muslea and Knoblock (2006) proposed to use the cooperation between (classifiers learnt on different) views in order to minimize the human intervention. Contrary to co-training, Muslea and Knoblock (2006) used the disagreement between the views in order to detect the *interesting* example. The intuition is that the more two views disagree on an example, the more information it contains and the more useful it would be to include this example in the labeled set. Other works related to multi-view active learning include Wang and Zhou (2008), Wang and hua Zhou (2010), Zhang et al. (2009), etc.

In the supervised setting — which is the setting of the works presented in this thesis —, the cooperation between the views was first (implicitly) used by Multiple

6. Note that this condition means that the considered views are *strong* views as defined in Section 1.3.1

Kernel Methods (MKL), presented in Section 1.3.2. An extensive review of MKL Algorithms is given in Gönen and Alpaydın (2011).

The main inconvenience of MKL is that the training procedure can be quite costly and, when the views are heterogeneous, the kernels should be proper to each view. One way to speed this up is to train simpler classifiers on each of the views and to combine them into a strong/performing one. These are the main motivations behind 2-Boost Janodet et al. (2009), a boosting method, based on AdaBoost, using several heterogeneous views. At each iteration, one classifier is learnt per view and the update of the weights depends on the predictions of all the classifiers. This way the information contained in each (heterogeneous) view are taken into consideration during the training process. The final hypothesis is the weighted majority vote of all the trained classifiers, which ensures that the different information from all the views are involved in the predictions.

A survey of multi-view methods in machine learning was recently proposed in Sun (2013).

Usually in the supervised setting, the cooperation between the views is mainly implemented in the combination of the various classifiers learnt on the views. The closer to promoting some sort of cooperation are the MKL, even though it is not explicit in the training process, since MKL build a *good* kernel per view. The cooperation between the views has also been used in the semi-supervised setting (co-training) and the active setting (co-testing) in order to learn better classifiers and minimizing the role of human agents in the training process (active learning). We think that encouraging the views to communicate between them during the training process, in the supervised setting, can be a good strategy in order to enhance the performances of the final classifier. This can be particularly interesting in cases when each view (correctly) recognizes only a small region of the input space (think of the case when the training data are quite noisy and only some of them contain valuable information). So, the communication between the views would allow each view to focus on the examples it recognizes best and vice versa in such a way that each example would be processed by the most appropriate view. This is one of the main motivations behind the works described in Chapter 2 and Chapter 4.

## 1.4 Dealing with imbalanced classes

When dealing with *real-world* data, often some of the classes are less represented than the others, that is, only a few examples from those classes are available. Think, for instance, of the automatic disease diagnostic problem, where the number of patients infected by a disease is smaller than the number of sane patients. Such learning problems, where classes are not represented equally in the training set, fall in the so called *imbalanced learning* problem. For simplicity's sake, we will refer as *minority classes* to the least represented classes and as *majority classes* to the most represented ones. The *imbalanced ratio* refers to the ration between the number of instances in biggest majority class and those in smallest minority class.

Using traditional learning methods — that is, methods that do not take into consideration this disproportion between the classes — is not a good strategy, since they often over-focuson the majority classes. Henceforth methods that deal with imbalanced learning have been proposed in the past years and they mainly fall in two categories:

a) methods that modify the training sample in order to have the same number of examples for each class;
b) methods that assign penalization costs to each class so that they have roughly the same weighted error.

We detail both cases in the following.

### 1.4.1 Methods that modify the training sample

A popular way to deal with the imbalanced classes problem is to make use of resampling methods in order to even the misbalance between the classes. The goal of these methods is to modify the distribution of the training examples, so that all the classes contain roughly the same number of examples. We distinguish two cases of resampling: *oversampling* and *subsampling*.

Oversampling methods increment the size of minority classes by duplicating some of the minority instances contained in the training set. The duplication process continues till some stopping criterion is attained. On the other hand, subsampling methods operate on majority classes by selecting only a few of the majority instances, in such a way that the new training set becomes balanced. The selection process can be random or based on bootstrapping.

The main advantage of resampling methods is that they use only examples contained in the original training set, thus minimizing the risk of introducing *artificial* noise to the final training set. Indeed, oversampling keeps the information contained in the majority instances, while aiming to capitalize on the information of the minority classes, while subsampling puts the minority classes on the same ground as the majority classes, with the same number of instances per class. However, on the downside, oversampling can lead to the duplication, thus promotion, of noisy examples or not informative ones, while subsampling may result in information loss for the majority classes, since only a part of the original examples is kept.

It is interesting to notice that resampling methods, as presented here, can be seen as special cases of assigning weights to the training instances (which fall in the group of cost sensitive methods, described in Section 1.4.2). Indeed, the oversampling methods duplicate the existing instances, as such the weight assigned to each example correspond to the number of duplicates it has in the training set(+1). On the other hand, since the subsampling methods remove instances from the training set, the weights correspond to boolean values, 1 if the instance is present in the final set and 0 otherwise.

Manipulating the training set is not limited to resampling methods and researchers have been interested in how to augment the size of the minority classes by forging artificial instances. The interest of artificial instances is that they allow to better define the minority instances. Think of the case where the instances of a minority class follow a Gaussian distribution. Then, by computing the mean and standard deviation of the instances in the training set — which makes a good estimation for the true mean and standard deviation —, it is possible to forge new artificial data which follow the same distribution, that is, enhance the informations related to the minority class, while at the same time minimizing the risk of introducing noise in the final training set.

A popular method when it comes to data generation is the synthetic minority oversampling technique (SMOTE) by Chawla et al. (2002). It generates artificial data by selecting two instances, one of a minority class and the second in the neighborhood

of the first, and computing the values of the new instance's attributes as (roughly) the mean of the values of the selected instances' attributes. SMOTE has received great attention and has been used in different methods Wang and Japkowicz (2004), Batista et al. (2004). Improvement of SMOTE have also been proposed such as Borderline-SMOTE Han et al. (2005) and Adaptive Synthetic Sampling He et al. (2008), in order to patch some of SMOTE's drawbacks (see Wang and Japkowicz (2004) for further details).

Other methods related to resampling methods for the imbalanced problem include the works by Stefanowski and Wilk (2008), Estabrooks et al. (2004), Guo and Viktor (2004), etc.

### 1.4.2 Cost sensitive methods

Resampling methods, presented in Section 1.4.1, can be quite effective since they tend to make use of the information embedded in the training sample. Alas they are not the miracle solution. Consider, for instance, the case where the number of classes is high and only a couple of them are majority classes (the rest are minority ones, and the imbalance ratio is high). Using oversampling methods would result in increasing the size of the training sample (by a lot), thus slowing down the training process. Likewise, subsampling is not a viable choice, since it would require multiple runs in order to have statistically significant results. This is where *cost sensitive* methods come into play.

The term *cost sensitive* regroups all methods that make use of class-based loss functions during the *training* phase in order to build models that take into account the imbalanced rate of the training set. The main difference between resampling methods and cost sensitive ones, is that the former are mainly used as preprocessing routines before the actual training phase, while the latter are closely linked to the training phase. Cost sensitive methods differ one from the other from *how* they define the loss functions. A common approach is to consider per-couple-of-classes penalization terms, usually coming from a cost matrix. Let $K$ be the number of classes and $\mathbf{C} \in \mathbb{R}^{K \times K}$ a cost matrix, then $\mathbf{C}(p, q)$ — $p$ is a row and $q$ is a column — defines the cost of predicting class $q$ for examples of class $p$. During the training procedure, if the learning method misclassifies an example of class $p$ as of class $q$, then the error is weighted by $\mathbf{C}(p, q)$. These cost matrices are usually given prior to the training procedure: they are either obtained based on prior information on the training set, or computed during a pre-processing procedure.

As an example of methods using cost matrices, Ting (2000) proposed an extension of AdaBoost (algorithm 1) to the imbalanced setting by using per-class misclassification costs in the weight update formula. Three variants of AdaBoost, named SCBo-1-2, were proposed based on where the misclassification coefficient was introduced in the formula.

In Sun et al. (2007) a different definition for the cost matrix is used: instead of considering per-class misclassification costs, they used per-example misclassification cost. Their method is yet another extension of AdaBoost to the imbalanced classes framework similar to SCB, since the misclassification costs are included in the weight update formula.

Another interesting approach to imbalanced problems was given in Wang et al. (2010). Their method, named AdaBoost.NC, does not depend on pre-computed cost-matrices, but it maintains (and updates) a cost matrix parallel to the distribution. Each example is given a penalization cost and its update depends on how well the example

is recognized throughout the training process. The advantage of AdaBoost.NC is that it can be easily applied to multi-class problems, contrary to other methods that apply one-versus-one or one-versus-all approaches Abe (2004), Wang and Yao (2012).

Methods other than boosting based have also been proposed, such as works based on bagging Barandela et al. (2003) Wang and Yao (2009), decision trees Maloof (2003), Support Vector Machines Akbani et al. (2004), etc.

Extensive studies on defining a taxonomy for cost-sensitive methods have been conducted in recent years such as the works provided by Galar et al. (2012) and He and Garcia (2009).

### 1.4.3 Evaluations measure for the multi-class setting

Traditionally, the misclassification rate, given in Equation 1.6, has been one of the most use methods, when evaluating the performances of a classifier.

$$misclassification\ rate = \frac{\#\text{incorrectly classified examples}}{\#\text{examples}} \tag{1.6}$$

However, in the imbalanced setting, this measure is not the fair measure to use, since it does not take into consideration the rate of misclassification is the different classes. Take for example, a training set made of 99% of examples of one class, and 1% of an other class. Then the majority vote classifier, which constantly predicts the first class, has a misclassification rate of 1%, although it is not a good predictor at all (if the first class if non deadly mushrooms and the second one deadly mushrooms, the consequences of this classifier would be quite severe). It is obvious that other evaluation measures need to be considered for this particular case.

Since most learning approaches to the imbalanced problem have been developed for the binary classification setting, it follows that most the evaluation methods have been proposed for that setting. Most of these methods are easily extendable for the multi-class setting, but, for simplicity's sake, we'll present them only in their binary form. Same as in page 1, we note the two classes as the positive one and the negative one.

First, let us define the confusion matrix for a given classifier $h$ as follows:

| | $predicted positives$ | $predicted negatives$ |
|---|---|---|
| $real positives$ | $TP$ | $FN$ |
| $real negatives$ | $FP$ | $TN$ |

In the confusion matrix, $TP$ (true positives) represents the number of positive example that are classified as positives, $FP$ (false positives) represents the number of negative examples misclassified as positives, $FN$ (false negatives) represents the number of positive examples classified as negatives and, lastly, $TN$ (true negatives) represents the number of correctly classified negative examples. The confusion matrix contains informations both on how the classifier recognizes each class (rate of true positives and rate of true negatives), and how it errs on each on them (rate of false positives and rate of false negatives):

$$TP_{rate} = \frac{TP}{TP+FN}, \quad TN_{rate} = \frac{TN}{TN+FP}, \quad FP_{rate} = \frac{FP}{TN+FP}, \quad FN_{rate} = \frac{FN}{TP+FN}.$$

$TP_{rate}$, the positive class accuracy, is also called *sensitivity*, while $TN_{rate}$, the negative class accuracy, is called *specificity*.

Some of the usual metric computed from the confusion matrix are the accuracy (which is also $1 - misclassification\ rate$), the precision and the recall:

$$Acc = \frac{TP+TN}{TP+FN+FP+TN}, \quad Precision_{neg} = \frac{TP}{TP+FP}, \quad Recall_{pos} = \frac{TP}{TP+FN}$$

These definitions of precision and recall are given for the positive class, and obviously, they are defined in a similar way for the negative class.

Two other measures, which make use of the previous definitions, are the $G - mean$ (the geometrical mean of the recalls) and $F - Measure$.

$$G - mean = \sqrt{Recall_{pos} * Recall_{neg}} = \sqrt{sensitivity * specificity}$$

$$F - Measure_{pos} = (1 + \beta^2)\frac{Precision_{pos} * Recallpos}{(\beta^2 Precision_{pos}) + Recall_{pos}}, \ \beta \in \mathbb{R}^+$$

$$F - Measure_{neg} = (1 + \beta^2)\frac{Precision_{neg} * Recallneg}{(\beta^2 Precision_{neg}) + Recall_{neg}}, \ \beta \in \mathbb{R}^+$$

Now, suppose that the classifier $h$ returns a score, instead of a class; in this case the example is classified as positive if the score returned by the classifier is greater than a certain threshold. Changing the threshold has a direct effect on the previous measures, since it changes the rates of correctly classified examples (both in the positive and negative class). Moreover, varying the threshold from the lowest value to the highest one and computing the values of $G - mean$ for the predictions of $h$, allows to draw a curve, also known as the receiver operating characteristic (ROC) curve. The size of the area under the ROC curve (AUC) is a good estimator of the performances of $h$, and as such it has been used as an evaluation measure for classifiers in the imbalanced setting Bradley (1997). Similar to AUC it is possible to estimate the area under the Recall-Precision curve, which is computed from the $F - Measure$ rather than the $G - mean$. A simple definition of AUC is the following (give for example in K. Tang and 2011. (2011)):

$$AUC = \frac{\sum_{x_1 \in positives, x_2 \in negatives} s(x_1, x_2)}{n_{pos} * n_{neg}},$$

where $n_{pos}$ is the number of positive examples, $n_{neg}$ the number of negative examples and $s$ is defined as follows:

$$s(x_1, x_2) = \begin{cases} 1 & \text{if } h(x_1) > h(x_2) \\ 0.5 & \text{if } h(x_1) = h(x_2) \\ 0 & \text{if } h(x_1) < h(x_2) \end{cases}$$

Since AUC is exclusively defined for the binary case, there are multiple extensions of it to the multi-class framework. We present here the one proposed in Hand and Till (2001):

$$MAUC = \frac{1}{K(K-1)} \sum_{i \neq j} AUC_{l,c},$$

where $K$ is the number of classes, $i$ and $j$ are classes and $AUC_{l,c}$ is the AUC between class $l$ and $c$.

### 1.4.4 Prospective evaluation measure: the norm of confusion matrix

In Chapter 3, we propose yet another measure for the goodness of a classifier, which is the norm of the *probabilistic*[7] confusion matrix. However, instead of considering the whole confusion matrix, we propose to keep only the non diagonal entries,

7. In the probabilistic version, $TP$ is replaced by $TP_{rate}$, $TN$ by $TN_{rate}$, and so on.

since they represent the *errors* of the classifier. For the binary case, this matrix is defined as follows:

| | $predicted positives$ | $predicted negatives$ |
|---|---|---|
| $real positives$ | 0 | $FN_{rate}$ |
| $real negatives$ | $FP_{rate}$ | 0 |

It is interesting to notice that the (operator) norm of this matrix, which will be defined later on, is equal to $\max\{FN_{rate}, FP_{rate}\}$. So, if the norm is used as an optimization criterion in a learning method, then the goal would be to minimize the maximum of the two quantities, thus implicitly aiming to achieve the same error on both classes. Further details are given in Chapter 3.

## 1.5 Introducing the DECODA project

The imbalanced classes problem, presented in Section 1.4, is a common occurrence in datasets coming form real-world applications. Such is the case for the DECODA project, which deals with phone calls classification and the goal is to predict the theme of the phone call. Aside from the imbalanced facet of the data (some themes are more occurring than others), phone calls present an interesting setting for multi-view learning. The audio signal of the calls is usually noisy and/or it contains various interferences, thus depending on the representation, some features could be more noisy than the others. Although in Section 1.3, the notion of *strength* of a view is linked to the error of the Bayes classifer, noisy features could also lead to views of different strength. The aim of describing the DECODA problem as a multi-view on is to study whether promoting the cooperation (both in the input and output spaces) between various views computed from noisy features could lead to better performances than fusion based approaches.

### 1.5.1 Paris from the RATP viewpoint

The DECODA project (*fr.* DEpouillement automatique de COnversations provenant de centres D'Appels) deals with data coming from the Paris transport authority (RATP, Régie autonome des transports parisiens). The data correspond to human-human conversations between the operators and the users of the public transport in Paris and the operators are asked to assign a theme to each conversation. Labeled conversations can be quite useful when it comes to receiving feedbacks from the users and planning strategies for improving the performances of the transport network. However manually labeling the conversations can be quite costly and time consuming. A viable solution is the automatically processing the data through machine learning methods.

The automatic processing of these conversations is quite challenging because of the spontaneous nature of the language used and the surrounding noise in most of the speech signals. For instance, in calls made from people on the street or a subway station, it is not uncommon for the surrounding noises (klaxons, loud voices, etc.) to overlap with the voice of the user. The various noises can lead to not-so-reliable features (and by extension, to not-so-reliable views), since phonetically similar words can be mistaken one for the other. This is its turn influences the strength of the views computed from the features, since for some view, characteristic features for one class may be replaced by features proper to other classes. Other potential *noises* come also

| | aapl | etfc | horr | itnr | nvgo | objt | pv | vgc | other |
|---|---|---|---|---|---|---|---|---|---|
| train | 20 | 117 | 27 | 94 | 68 | 75 | 30 | 24 | 53 |
| test | 15 | 41 | 5 | 21 | 22 | 12 | 10 | 11 | 15 |

(a) *Classes distribution in train and test*

| | train | | test | | test |
|---|---|---|---|---|---|
| speaker | #turn | #word | #turn | #word | WER |
| Operators | 12423 | 81422 | 4341 | 27408 | 56.3 |
| Users | 10867 | 75545 | 3632 | 23551 | 67.3 |

(b) *Dataset description*

Table 1.1 – *Description of the conversation corpus and WER obtained on the test corpus*

from misspelled words, the accent of the speaker or even unfinished sentences, all related to the spontaneous nature of the speech.

On the other hand, as most real-world problems, DECODA is also an imbalanced classes problem, where some of the conversation themes are more frequent than others. This is mainly related to the fact that phone calls on traffic issues and/or status are more frequent than calls on fines.

To sum up, DECODA presents an interesting setting, both for multi-view approaches and imbalanced classes ones. In this thesis, these different facets are dealt with in their corresponding chapters. Chapter 2 deals (only) with the multi-view aspect of DECODA (Section 2.5), while aiming to bring an answer to wether installing (and promoting) some cooperation in the input space may lead to more robust methods. In Chapter 3 we tackle DECODA's imbalanced classes aspect (Section 3.5). Finally, Chapter 4 regroups both the multi-view and imbalanced aspect of DECODA, while transposing the cooperation to the output space (Section 4.5).

### 1.5.2 The DECODA corpus

**Presenting the corpus** Throughout this thesis, we work on a subset of the DECODA corpus, made of 600 dialogs, split into 508 dialogs for training and 152 dialogs for testing. Nine different semantic categories are considered: *fares* (aapl), *timetable* (horr), *traffic info* (etfc), *pass navigo* (nvgo), *lost+found* (objt), *itinerary*(itnr), *fines*(pv), *frequent traveler* (vgc) and *other*. In order to test the methods presented in this thesis, we consider two different settings for the testing sample: in the first setting the transcriptions of the data are handmade, that is, human agents are asked to transcribe the phone calls; in the second setting the transcriptions are the outputs of an Automatic Speech Recognition (ASR) system. We'll refer to the first case as *gold* and to the second as *ASR*. As for the training sample, in both cases the same sample is used, made of handmade transcriptions of the conversations (*gold*).

Table 1.1 gives a summary of the considered corpus. The imbalanced nature of the classes in the DECODA project is clearly shown in Table 1.1a: *traffic info* is way more represented than *frequent traveler*. In Table 1.1b, the number of turns consists of both the number of turns of the operators and that of the users. Likewise the number of words represents the total of all the words in all the conversations. The last column, the WER (short for Word Error Rate), gives the rate of the words that are different between the ASR testing set and the gold testing one. In other words, it measures how often the ASR system errs on the words it predicts; the smaller the WER the better the ASR system performs. In our case, WER is more then 50, both for the users and to operators. Such an WER was to be expected since we deal with spontaneous

speech *and* noisy signals. In order to illustrate this, let us have a look at the following examples:

1. gold: j' aimerais connaître le prix d' un billet arcueil cachan orly
   ASR: aimerais connaître le prix d' un billet arcueil cachan orly ligne
2. gold: en fait voilà je vous appelle parce que j' ai perdu ...
   ASR: en tête voilà j' ai une dame elle est ce que j' ai perdu ...
3. gold: qui a eu un accident mardi soir ...
   ASR: signalé une maxi de temps en mardi trois ...

In the first example, the ASR achieves a nearly perfect score since all the words are recognized correctly. The last word (*ligne*) is added due to the quality of the signal, and it's safe to assume that the last syllable of *orly* plays an important role in this mistake. In the second example, the words *en fait* are mistakenly recognized as *en tête*, which is understandable since their pronunciation is nearly the same. Lastly, both in the second and third example, the ASR system outputs words that are phonetically close to a part of the original words in most cases, due to the quality of the signal and the spontaneous nature of the speech.

It is interesting to notice that there is more than a 10% absolute WER difference between the automatic transcriptions of the operators' turns and the users' one. This can be explained by the fact that most of the operators are both in the training and test corpus and therefore the acoustic models are adapted to them. Another explanation is that the users' signal is significantly noisier than the operators' one since they often phone from a street or a subway station. However, because the recording device in the call-center mixes the two channels, the noise affects the whole signal.

**Presenting the views** A conversation can be described in three different ways:

1. the *content* of the conversation (mainly its transcription),
2. the *prosody* represented by general statistics of the conversation,
3. the *interaction* between the speakers.

In order to process the DECODA corpus in a multi-view learning framework, we define five different views based on the three aspects of a conversation: three for the *content*, one for the *prosody* and one for the *interaction*.

Concerning the *content* of the conversation, we choose to represent it by the words of the operators (first view), the words of the users (second view) and the named entities (fifth view). The first and second views are quite informing on how the speakers approach the conversations and it usually contains keywords that make it easier to identify the theme of the conversation. For instance, in the following example, key words such as *grève pour demain* strongly suggest that the theme of this particular conversation is *traffic info*.

*op.:* *bonjour*
*user:* *bonjour madame*
*user:* *à massy palaiseau demain est ce que le r e r b sont encore en grève*
*op.:* *alors écoutez concernant donc la grève pour demain nous n' avons aucune information pour l' instant*
*user:* *d' accord*
*op.:* *fin d' après midi*

Table 1.1b suggests that both views are quite sensitive to WER and to the introduction of noise from ASR systems. Due to the nature of the data, the second view may

contain fewer informations compared to the first one, however Table 1.1b suggests that it is less sensitive to artificial noise introduced from the ASR system.

The fifth view is build on the notion of named entities, which are atomic elements that clearly identify one item from a set of other items that have similar attributes. Named entities include (but are not limited to) name of persons, locations, quantities, monetary values, etc. In DECODA, named entities are principally related to the names of the bus/train/subway stations, public places, subscriptions and so on. This view can be seen as a simplified version of the first two views and its goal is mainly to guide the learning procedure towards easy-to-recognize elements of the dialogs.

The *interactions* between the speakers is represented by a global structure based on the turn-taking aspect of a conversation. We propose to represent each turn by a letter, $O$ for operators and $U$ for users, and a number representing the duration of the turn normalized into 6 duration lengths. This is done to capture the fact that sometimes the users take more time (think for example when users need to explain a problem), and other times it's the operators that take more time (when replying to the users, providing informations on the services, etc.). A whole conversation is thus described by a sequence such as: $O1U4O2U5O4U1O1U1\ldots$

The prosody of a conversation, and the forth view, is mainly represented by general statistics computed from the conversation itself, such as the total duration of the call, the speech rate (in letters per second) and speech duration for each speaker.

Although these views might not be the best representatives for the different aspects of the conversations, they were fixed at the beginning of this thesis and did not change ever since.

# Promoting the cooperation between views through boosting

# 2



Nowadays, in fields such as bioinformatics or multimedia, data may be described by different sets of attributes, also called views (see Section 1.3). In Section 1.3.1, we defined three types of views based on the information embedded therein. The work presented in this chapter deals with the first two type of views, weak and strong views, where the information is contained in the input space and it can be either global of local. While global information can be captured by classifiers learnt on the whole input space, local information needs specialized classifiers. For instance, if a view contains useful information only for a subset of examples, then it would be preferable to learn a classifier that correctly classifies that particular set of examples (and forsaking the other examples).

The method proposed in this chapter, named MuMBo, is based on the multi-class boosting framework presented in Section 1.2.2. It relies on the weak classifiers to capture the local informations usually contained in the weaker views. This is implemented by defining and maintaining one distribution for each view. Compared to late fusion approaches presented in Section 1.3.2, where the classifiers are learnt separately, the originality of MuMBo consists in promoting the cooperation between the views, so that each view can focus on the examples it recognizes the best. MuMBo's aim is to capitalize on the information extracted from each view, so that each example is processed by the most appropriate view, and vice versa.

The motivations, principles (and hypothesis) and the algorithm itself are presented in Section 2.1 and Section 2.2. In Section 2.3, we give a theoretical study of the boosting properties of MuMBo, related to the empirical error and true risk, while Sections 2.4 and 2.5 relay the results of MuMBo on synthetic data and data from the DECODA corpus. Finally, in Section 2.6, we detail a general version of MuMBo and Section 2.7 offers a discussion on the implications and prospects of the work in this chapter.

The work presented in this chapter was first published in the Proceedings of the 11th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML-PKDD 2011, Athens, Greece, Koço and Capponi (2011). The work presented in Section 2.5 was first published in the Proceedings of the 13th Annual Conference of the International Speech Communication Association, Interspeech 2012, Portland, Oregon, Koço et al. (2012).

## 2.1 Introduction and Motivations

In many application domains of machine learning, such as bioinformatics or multimedia indexing, the descriptions of the data may come from several sources or views, for instance, in Masulli and Mitra (2009), Mansoorizadeh and Charkari (2010). When facing a classification or a regression task, the use of multiple views might be of great interest, since each view is supposed to carry information not embedded in other views (cf. Section 1.3). Many methods exploit these enclosed information in order to select the most informative sources, or set of features, and build models which either best discriminate data concepts or best describe one concept among others, Culp et al. (2009), Sridharan and Kakade (2008).

Most of these selective methods base their choices on the training set as a whole and, as such, they tend to discard localized or isolated information, which could be useful to compensate the lack of performance on some (group of) learning examples. Think for example of the case of noisy data, which is quite a common issue for data coming from real-life problems. In such cases, only a part of the training examples contains useful information, while the noisy part usually tends to mislead the training method and worsen the generalization capabilities of the learnt models. 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 experimentally, to be able to lessen the strain put on the generalization capabilities of classifiers. In such scenarios, the interest of considering multi-view learning methods — and multi-view approaches in general — resides in the fact that the localized lack of classifiers trained on one view, can be compensated by the performances of the classifiers trained on the other views. The main idea here consists in installing

some sort of cooperation between the views in order to obtain classifiers with better generalization properties..

In the review of multi-view methods, given in Section 1.3, we saw that in the supervised learning case, the most common multi-view methods were the fusion approaches, both early and late ones. The main drawback of fusion based methods resides in the fact that there is little to no communication between the views. In the early fusion case, this lack of communication may lead to increasing the number of noisy attributes, while in late fusion, since the classifiers are learnt independently on each view with no communication whatsoever, it may lead to a redundancy of the learnt classifiers (think of the case where classifiers learnt on different views recognize the same examples). This simple observation implies that fusion based methods perform better when the considered views are independent. Alas such (strong) condition can be quite difficult to achieve in practical cases, since the views often share mutual information.

We think that promoting the cooperation between the views during the training phase, can be an interesting way to establish a sort of communication between them during the learning phase. More precisely, when a classifier on a view fails to recognize a region of the the instance space, it entrusts the other views with the classification of the examples contained in that region. One of the major difficulties is then to delimit, for each view, the concerned subareas, without penalizing the generalization capabilities. Instead of locating precisely these subareas, we propose an algorithm based on boosting (a framework presented in Section 1.2), whose principle is to slightly remove the hard-to-classify examples from the learning space of one view, while encouraging other (more adapted) views to process these examples. This way, we expect the examples to be processed by the most appropriate views.

In order to implement this principle, we designed MuMBo as a boosting-like algorithm based on a multi-class version of AdaBoost and adapted for the multi-view framework. The interest of using a boosting framework similar to AdaBoost is two fold: the presence of a distribution and the iterative nature of the method. Maintaining a distribution over the training sample can be seen as a way to encode different information on the examples, which in our case are closely related to the capability of a view to recognize a certain part of the instance space. As such, each view is associated with a weak learning algorithm, and one distribution per view is maintained. On the other hand, the iterative nature is an excellent tool for the detection of the hardest examples. At each iteration, the distribution update takes into account not only the performances of that view in classifying learning examples, but it also embeds the performances of the other views. Hence, the capabilities of every view to process the examples are broadcasted to all the views.

## 2.2 A multi-view boosting method: MuMBo

### 2.2.1 Motivations

MuMBo is a multi-view boosting algorithm: each example of the learning sample $S$ is represented by several sets of features, which are called *views*. Each of these views can be used to train models, which are then used to classify other examples. Even though the models are learnt on different representations of the same examples, they are by no means equal, performance wise. This is especially true when the noise or the quantity of information contained in each view is different form one view to another, mostly due to the differences in the associated representation spaces. In this

case, the classifiers learnt on some of the views perform better than those learnt on other views. Based on the performances of the classifiers learnt on each view, we distinguish two different type of views: *strong* views and *weak* views. Roughly, the notion of a *strong* view is associated with the possibility to learn a good classifier on that view, while the notion of *weak* view reflects the impossibility of learning such a classifier from the instance space defined by the view. In Section 1.3.1, the notion of *strength* of a view was linked to the error of the Bayes classifier learnt on the view. The lower the error of the Bayes classifier, the stronger the view.

MuMBo was designed in order to learn a classifier in a multi-view setting, where views are supposed to be of different strength. Also, we assume that among the views, there exists at least one strong view, while the others are weak(er) views. In the case where there is only one strong view, we refer to it as the major view ($V$), while the others are called minor views ($v_1, \cdots, v_z$).

The main interest of placing such a hypothesis as the foundation of MuMBo, is mainly in order to avoid the cases where all the views are weak ones, hence making the learning process quite challenging. Think of the case where the noise rate in each view is such that it makes it impossible to learn a good classifier with good generalization properties. Another way to look at this hypothesis is that the learning process consists of using the few informations contained in the weaker views, in order to improve the performances of the major view (strong views). On a side note, this hypothesis is usually verified in most real-world problems, even though it may seem as too strong and/or restrictive. For instance, in the introductory example on image classification (cf. Figure 1), we argued that the four views defined on the images (histogram of colors, bag of words, bag of images and HOG) can be used to train good classifiers depending on the classification task. That is, for a classification problem, it is possible to find a view that allows to train a good classifier.

As pointed out in the introduction, the basic principle of MuMBo is to encourage each view to focus on the examples are hard to process in other views, while being easy for the view itself. 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.

### 2.2.2 The core of MuMBo

**Views, distributions and cost matrices** The proposed algorithm, called MuMBo (the simplified version in Algorithm 2 and the complete one in Algorithm 3), is an attempt to encourage the cooperation between views of different strengths.The goal of MuMBo is to enhance the performances of the classifiers usually learnt only on the stronger views, through the use of the informations contained in all the views, especially the weaker ones. This implies that throughout the learning process, each view should be able to keep (and eventually exploit) its individuality/typicality and not to be influenced (too much) by the other views. In the boosting frameworks presented in Section 1.2, information related to the capability of recognizing the training examples are embedded in a distribution maintained over the learning sample. As such, defining and maintaining a distribution per view seems to be a reasonable choice for keeping each view's individuality. In MuMBo, the updates of the distributions are primarily based on the results obtained from the classifiers learnt on their corresponding views. More precisely, the weights of the examples that are correctly classified are systematically updated, while for the misclassified examples, the update is closely related to the cooperation. By doing so, we expect a distribution to keep track of the

**Algorithm 2:** MuMBo: MUltiModal BOosting (in a nutshell)

Initialize each $\mathbf{D}_{1,j}$
**for** $t = 1$ **to** $T$ **do**
  **for** $j = 1$ **to** $m$ (for each view) **do**
    Train $h_{t,j}$ with learning algorithm $L_j$ on $\mathbf{D}_{t,j}$
    Compute edge $\delta_{t,j}$ and $\alpha_{t,j} = \frac{1}{2} \ln \frac{1+\delta_{t,j}}{1-\delta_{t,j}}$
  **end for**
  Update cost matrix $\mathbf{D}_{t,j}$ (for each view)
  Choose

$$\begin{cases} h_t = \underset{h_{t,j}}{\operatorname{argmax}}(\text{edge } h_{t,j} \text{ on } \mathbf{D}_{t,G}) \\ \delta_t = \{edge\ of\ h_t\ on\ \mathbf{D}_{t,G}\} \end{cases}$$

  Compute $\alpha_t = \frac{1}{2} \ln \frac{1+\delta_t}{1-\delta_t}$
  Update $\mathbf{D}_{t,G}$, the global cost matrix
**end for**
Output final hypothesis :

$$H(x) = \underset{l \in 1,\ldots,k}{\operatorname{argmax}} f_T(x, l)$$

performances of its view, in particular it should keep track of the examples that are easily recognized.

MuMBo is based on the multi-class boosting framework given in Mukherjee and Schapire (2011), where distribution are replaced by cost matrices. The framework was recalled in Section 1.2. In Algorithms 2 and 3, the cost matrices are denoted $\mathbf{D}_{t,j}$, where $t$ corresponds to the iteration and $j \in \{1 \ldots v\}$ corresponds to the view. The cost of classifying the example $(x_i, y_i)$ in class $l$ for the view $j$ at iteration $t$, is given by $\mathbf{D}_{t,j}(i, l)$. The *cost* of example $(x_i, y_i)$ is given by $\mathbf{D}_{t,j}(i, y_i)$ and a simple way to pass from the cost-matrix representation to a distribution is to affect the weight $-\mathbf{D}_{t,j}(i, y_i) / \sum_{e=1}^{m} \mathbf{D}_{t,j}(e, y_e)$ to the example $(x_i, y_i)$.

**Installing the cooperation in MuMBo** The cooperation between the views is implemented in two different parts of the learning process. Firstly, we promote the cooperation between the views by computing *cooperation coefficients* for each example in all the views. Secondly, a unique classifier is chosen at each iteration, thus finding the most adapted view for the sample (and the distribution).

The cooperation coefficients, noted $d$ in Algorithm 3, take binary values (0 or 1) and they are closely related to the performances of the classifiers learnt on all the views. Their role is to detect, for each view, the examples that need to be updated. We distinguish two cases when examples can be updated:

1. examples correctly classified in a view,
2. examples misclassified by all the views.

More precisely, for a given example $i$, view $j$ and iteration $t$, if the classifier learnt on $j$ correctly classifies $i$, then its coefficient $d_{t,j}(i)$ is 1, thus allowing the costs of $i$ to be updated in view $j$. If $i$ is misclassified in $j$ but it is correctly classified in at least one of the other views, that $d_{t,j}(i)$ is 0, that is, the costs for example $i$ are not updated in view $j$. By doing so, the view $j$ is less likely to focus on example $i$, at least in the short term, since it may not be informative in $j$ and, more importantly, it is recognized by

at least one other view. If $i$ is misclassified by all of the views, then its coefficient is 1 for all the views. This is done in order to promote the example $i$ in all the views, so that it can be correctly classified by at least one of them.

As a boosting method, MuMBo builds the final hypothesis by iteratively adding weak classifiers coming from the different views and selected based on some cooperation criterion. The second part where the cooperation between the views is implemented is the choice of a unique classifier at each iteration. In MuMBo, this classifier is chosen from the $v$ classifiers learnt on the views, and more precisely, it is the one that has the smallest error on the global cost matrix. The intuition behind this criterion is that the smaller the error, the more appropriate a view is for the considered cost matrix. This choice is closely related to the fact that boosting methods can be seen as gradient descent in the function space, cf. Mason et al. (1999). In our case, the function space is limited to the $v$ classifiers learnt on all the views. As shown in Mason et al. (1999), choosing the classifier that has the smallest error on the global cost matrix ensures that the drop in error of the combined classifier is maximal.

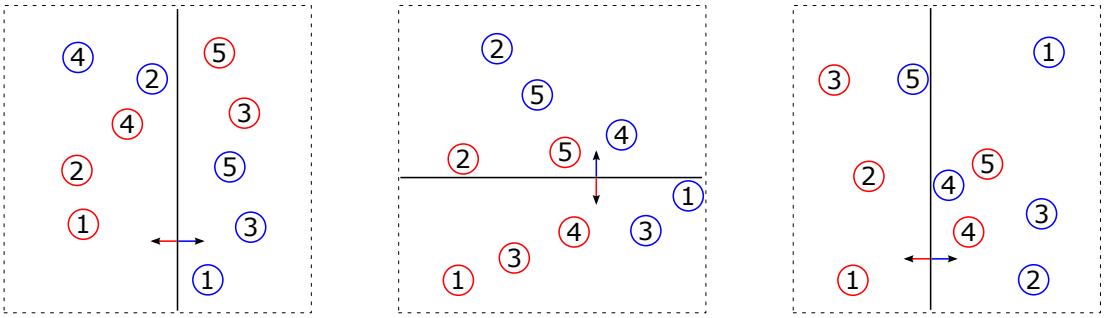
In order to keep track of the selected classifiers and to choose the most appropriate view, a global cost matrix is maintained, denoted $\mathbf{D}_{t,G}$ in Algorithms 2 and 3. The selected view is the one that better recognizes the training set weighted by the global cost matrix. The update of this cost matrix depends only on the performances of the selected classifiers, hence it embeds the different informations coming from the (selected) views. Including this global cost matrix, the number of cost matrices maintained at each iteration is $v + 1$, that is, $v$ local cost matrices maintained by the views and one global cost matrix maintained by all the views.

The pseudo-code of MuMBo is given in Algorithm 3. The output classifier is a weighted combination of the base classifiers chosen during the training phase and the predicted label is the one that has the biggest score, give by $f_T(x, l)$ in Algorithm 3 for an example $x$ and label $l$.

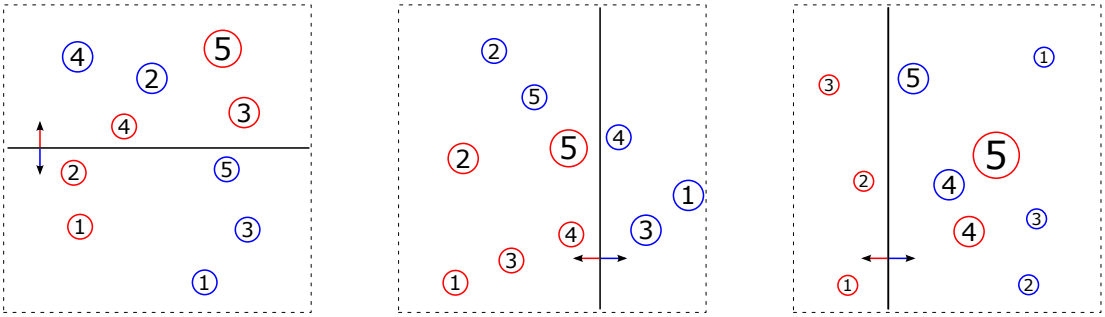
### 2.2.3 An example of MuMBo's update rule

An example of how this update process works and its impact on the learning process is shown in Figure 2.1. For simplicity reasons, we consider a binary classification problem, *red* class versus *blue* class. Each class contains five examples, numbered from 1 to 5, and each example is described by six attributes divided in three views of two attribute each. The weights of the examples are proportional to the cost put on the correct class. Figure 2.1a corresponds to the first iteration, and as such, the weights of all the examples are equal. The weak classifiers considered during the learning process are lines parallel to the axes. At each iteration, MuMBo trains one classifier per view, which correspond to the lines given in Figure 2.1a. The *red* (resp. *blue*) arrow defines the zone where the examples are predicted as belonging to class *red* (resp. *blue*).

In each view, some of the examples are correctly classified, while the others are misclassified. Let us take a closer look to the cooperation coefficients computed for the first view. In this first view the examples 1r, 2r, 4r, 1b, 3b and 5b are correctly classified, while 3r, 5r, 2b and 4b are misclassified. The cooperation coefficients for the correctly classified examples are automatically put to 1, as for the misclassified examples', they depend on the performances of the other views. Examples 3r, 2b and 4b are misclassified in the first view, however they are correctly classified in the second and third view, hence their coefficients are put to 0 for the first view. The last



(a) *The weights of the examples and the weak classifiers learnt on each view during the first iteration.*



(b) *The updated weights during the second iteration and the learnt classifiers.*

Figure 2.1 – *An example of the cooperation promoted between the views for MuMBo.*

example, 5r, is misclassified by all the views and, since we want all the examples to be processed by the most appropriate view, 5r is to be promoted in all the views, thus its cooperation coefficient is put to 1 for the all the views. Finally, for the first view we have: 1) the costs of the examples 1r, 2r, 4r, 1b, 3b and 5b will be decremented since they are recognized by the view; 2) the costs of the examples 3r, 2b and 4b are not updated; and 3) the cost of example 5r is incremented since all the views failed to recognize it.

The same process is repeated for all the views and the resulting costs (weights) are given in Figure 2.1b. Note that due to the performance of the classifier learnt on the third view, the weight of the example 5r is bigger in the third view than on the other views. Despite this, the third view still fails to recognize the example 5r during the second iteration. However, due to the increased weight of 5r in all the views, the views 1 and 2 correctly classify it in the second iteration. It is interesting to notice that views 1 and 2 are indeed more appropriate to process example 5r, since in view 3, it is nearly an outlier.

## 2.3 Properties of Mumbo

This section is dedicated to the theoretical properties of MuMBo. We start by showing that, for a given view $v$, if the weak classifiers learnt on $v$ verify the edge condition, given in Equation 1.5, then the total cost of the classifier made of the all the classifiers learnt on $v$ decreases after each iteration. This roughly means that for every view independently, a boosting process takes place. The second result consists in showing that the choice of the unique classifier at each iteration drives down the error of the majority vote, that is the classifier obtained from the combination of all the uniques classifiers.

**Algorithm 3:** MuMBo: MultiModal Boosting

**Given**

— S = $\{(x_i, y_i)\}_{i=1}^{m}$ where $x_i \in X_1 \times X_2 \times ... \times X_v$, $y_i \in \{1, ..., K\}$

— $v$ weak learning algorithms $WL$

— T the number of iterations

— $\forall i \in \{1, ..., m\}, \forall j \in \{1, ..., v, G\}, \forall l \in \{1, ..., K\}$ $f_{1,j}(i, l) = 0$

— $\forall j \in \{1, ..., v, G\}$ $\mathbf{D}_{1,j}(i, l) = \begin{cases} 1 & \text{if } y_i \neq l \\ -(K-1) & \text{if } y_i = l \end{cases}$

**for** $i = 1$ **to** $T$ **do**

  **for** $j = 1$ **to** $v$ **do**

    Train $WL$ using $\mathbf{D}_{t,j}$ and get $h_{t,j}$

    Compute:

$$\delta_{t,j} = \frac{-\sum_{i=1}^{m} \mathbf{D}_{t,j}(i, h_{t,j}(i))}{\sum_{i=1}^{m} \sum_{l \neq y_i} \mathbf{D}_{t,j}(i, l)} \text{ and } \alpha_{t,j} = \frac{1}{2} \ln \frac{1 + \delta_{t,j}}{1 - \delta_{t,j}}$$

  **end for**

  **for** $j = 1$ **to** $v$ **do**

    Update cost matrices:

$$\mathbf{D}_{t+1,j}(i, l) = \begin{cases} \exp(f_{t+1,j}(i, l) - f_{t+1,j}(i, y_i)) & \text{if } y_i \neq l \\ -\sum\limits_{p \neq y_i}^{K} \exp(f_{t+1,j}(i, p) - f_{t+1,j}(i, y_i)) & \text{if } y_i = l \end{cases}$$

$$\text{where } f_{t+1,j}(i, l) = \sum_{z=1}^{t} \mathbb{I}(h_{z,j}(i) = l) \alpha_{z,j} d_{z,j}(i)$$

$$\text{and } d_{z,j}(i) = \begin{cases} 1 & \text{if } h_{z,j}(i) = y_i \text{ or } \nexists q, h_{z,q}(i) = y_i \\ 0 & \text{otherwise} \end{cases}$$

  **end for**

  Choose $h_t = \underset{h_{t,j}}{\text{argmax}}(\delta_{t,h_{t,j}})$, where $\delta_{t,h_{t,j}} = \frac{-\sum_{i=1}^{m} \mathbf{D}_{t,G}(i, h_{t,j}(i))}{\sum_{i=1}^{m} \sum_{l \neq y_i} \mathbf{D}_{t,G}(i,l)}$

  Compute $\delta_t = \delta_{t,h_t}$ and $\alpha_t = \frac{1}{2} \ln \frac{1+\delta_t}{1-\delta_t}$

  Update $\mathbf{D}_{t,G}$ :

$$\mathbf{D}_{t+1,G}(i, l) = \begin{cases} \exp(f_{t+1,G}(i, l) - f_{t+1,G}(i, y_i)) & \text{if } y_i \neq l \\ -\sum\limits_{j \neq y_i}^{K} \exp(f_{t+1,G}(i, j) - f_{t+1,G}(i, y_i)) & \text{if } y_i = l \end{cases}$$

$$\text{where } f_{t+1,G}(i, l) = \sum_{z=1}^{t} \mathbb{I}(h_{z,m}(i) = l) \alpha_{z,m}$$

**end for**

Output final hypothesis :

$$H(x) = \underset{l \in \{1,...,K\}}{\text{argmax}} f_T(x, l), \text{ where } f_T(i, l) = \sum_{t=1}^{T} \mathbb{I}(h_t(i) = l) \alpha_t$$

The most important property – a bound on the generalization error of Mumbo – is proved in the last part of this section.

### 2.3.1 Bounding the training error on each view

The first result that we give in this section shows that, for a view $j$, the loss of the majority classifier

$$H_j(x) = \underset{l \in \{1 \cdots K\}}{\operatorname{argmax}} \sum_{t=1}^{T} \mathbb{I}(h_{t,j}(x) = l)\alpha_{t,j}$$

decreases at each iteration, provided that the weak classifier learnt on that view verifies the weak learning condition Eq. 1.5 on the cost matrix $\mathbf{D}_{t,j}$. Even though this result may not seem interesting at first, it implies that at each iteration the view $j$ focuses on the most informative examples for that view, which is the goal of MuMBo.

The following theorem is an adaptation to our Algorithm of Lemma 24 presented in Mukherjee and Schapire (2011).

**Theorem 1** (bounding the empirical loss in view $j$) *For a given view $j$, suppose the cost matrix $\mathbf{D}_{t,j}$ is chosen as in the Algorithm 3, and the returned classifier $h_{t,j}$ verifies the weak learning condition Eq. 1.5 for $\mathbf{D}_{t,j}$ and $\mathbf{U}_{\delta_{t,j}}$, i.e. $\mathbf{D}_{t,j} \cdot \mathbb{I}_{h_{t,j}} \leq \mathbf{C}_{t,j} \cdot \mathbf{U}_{\delta_{t,j}}$.*

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

$$\tau_{t,j} = 1 - \frac{1}{2}\left(e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}\right)\delta_{t,j} + \frac{1}{2}\left(e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2\right)$$

*of the loss before choosing $\alpha_{t,j}$, where $\delta_{t,j}$ is the edge of $h_{t,j}$.*

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

For readability reasons, we introduce the quantities:

$$L_{t,j}(i) = \sum_{l \neq y_i} \exp(f_{t,j}(i,l) - f_{t,j}(i,y_i)), \text{ and } \Delta_{t,j}(i,l) = f_{t,j}(i,l) - f_{t,j}(i,y_i).$$

Recall that the only hypothesis made in the theorem is that $h_{t,j}$ verifies the weak learning condition:

$$\mathbf{D}_{t,j} \cdot \mathbb{I}_{h_{t,j}} \leq \mathbf{D}_{t,j} \cdot \mathbf{U}_{\delta_{t,j}} \tag{2.1}$$

The left side of this equation can be written as:

$$\mathbf{D}_{t,j} \cdot \mathbb{I}_{h_{t,j}} = -\sum_{i \in S_+} L_{t-1,j}(i) + \sum_{i \in S_-} e^{\Delta_{t-1,j}(i,h_{t,j}(x_i))} + \sum_{i \in S_{-+}} e^{\Delta_{t-1,j}(i,h_{t,j}(x_i))}$$

and the right side can be written as:

$$\begin{aligned} \mathbf{D}_{t,j} \cdot \mathbf{U}_{\delta_{t,j}} &= \textstyle\sum_{i=1}^{m} \left(-L_{t-1,j}(i)\left(\frac{1-\delta_{t,j}}{K} + \delta_{t,j}\right) + L_{t-1,j}(i)\left(\frac{1-\delta_{t,j}}{K}\right)\right) \\ &= -\delta_{t,j} \textstyle\sum_i L_{t-1,j}(i) \end{aligned}$$

So, using the edge condition in Eq. 2.1, we have:

$$-\sum_{i\in S_+} L_{t-1,j}(i) + \sum_{i\in S_-} e^{\Delta_{t-1,j}(i,h_{t,j}(x_i))} + \sum_{i\in S_{-+}} e^{\Delta_{t-1,j}(i,h_{t,j}(x_i))} \leq -\delta_{t,j}\sum_{i\in S} L_{t-1,j}(i)$$

hence:

$$\sum_{i\in S_+} L_{t-1,j}(i) - \sum_{i\in S_-\cup S_{-+}} e^{\Delta_{t-1,j}(i,h_{t,j}(x_i))} \geq \delta_{t,j}\sum_{i\in S} L_{t-1,j}(i) \tag{2.2}$$

In order to compute the drop in loss $\sum_i L_{t-1,j}(i) - \sum_i L_{t,j}(i)$ after choosing $h_{t,j}$ with weight $\alpha_{t,j}$, let us consider three cases:

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

$$\begin{aligned}\mathcal{L}_+ &= \sum_{i\in S_+} L_{t-1,j}(i) - \sum_{i\in S_+} L_{t,j}(i)\\ &= \sum_{i\in S_+} L_{t-1,j}(i) - \sum_{i\in S_+} e^{-\alpha_{t,j}} L_{t-1,j}(i)\\ &= \left(1-e^{-\alpha_{t,j}}\right)\sum_{i\in S_+} L_{t-1,j}(i)\end{aligned}$$

2. For $i \in S_-$:

$$\begin{aligned}\mathcal{L}_- &= \sum_{i\in S_-} L_{t-1,j}(i) - \sum_{i\in S_-} L_{t,j}(i)\\ &= \sum_{i\in S_-} e^{f_{t-1,j}(i,h_{t,j}(i))-f_{t-1,j}(i,y_i)} - \sum_{i\in S_-} e^{f_{t,j}(i,h_{t,j}(i))-f_{t,j}(i,y_i)}\\ &= \sum_{i\in S_-} e^{f_{t-1,j}(i,h_{t,j}(i))-f_{t-1,j}(i,y_i)} - \sum_{i\in S_-} e^{f_{t-1,j}(i,h_{t,j}(i))+\alpha_{t,j}-f_{t-1,j}(i,y_i)}\\ &= -\left(e^{\alpha_{t,j}}-1\right)\sum_{i\in S_-} e^{f_{t-1,j}(i,h_{t,j}(i))-f_{t-1,j}(i,y_i)}\\ &= -\left(e^{\alpha_{t,j}}-1\right)\sum_{i\in S_-} e^{\Delta_{t-1,j}(i,h_{t,j}(x_i))}\end{aligned}$$

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

$$\begin{aligned}\mathcal{L}_{-+} &= \sum_{i\in S_-} e^{f_{t-1,j}(i,h_{t,j}(i))-f_{t-1,j}(i,y_i)} - \sum_{i\in S_-} e^{f_{t,j}(i,h_{t,j}(i))-f_{t,j}(i,y_i)}\\ &= \sum_{i\in S_-} e^{\Delta_{t-1,j}(i,h_{t,j}(x_i))} - \sum_{i\in S_-} e^{\Delta_{t,j}(i,h_{t,j}(x_i))}\\ &= 0 \text{ since the value of } f_{t,j} \text{ does not change for these examples.}\end{aligned}$$

So, the drop in loss $\mathcal{L} = \mathcal{L}_+ + \mathcal{L}_- + \mathcal{L}_{-+}$ is:

$$\begin{aligned}\mathcal{L} &= \left(1-e^{-\alpha_{t,j}}\right)\sum_{i\in S_+} L_{t-1,j}(i) - \left(e^{\alpha_{t,j}}-1\right)\sum_{i\in S_-} e^{\Delta_{t-1,j}(i,h_{t,j}(i))}\\ &= \left(\frac{e^{\alpha_{t,j}}-e^{-\alpha_{t,j}}}{2}\right)\left(\sum_{i\in S_+} L_{t-1,j}(i) - \sum_{i\in S_-} e^{\Delta_{t-1,j}(i,h_{t,j}(i))}\right)\\ &\quad - \left(\frac{e^{\alpha_{t,j}}+e^{-\alpha_{t,j}}-2}{2}\right)\left(\sum_{i\in S_+} L_{t-1,j}(i) + \sum_{i\in S_-} e^{\Delta_{t-1,j}(i,h_{t,j}(i))}\right)\\ &\geq \left(\frac{e^{\alpha_{t,j}}-e^{-\alpha_{t,j}}}{2}\right)\left(\sum_{i\in S_+} L_{t-1,j}(i) - \sum_{i\in S_-\cup S_{-+}} e^{\Delta_{t-1,j}(i,h_{t,j}(i))}\right)\\ &\quad - \left(\frac{e^{\alpha_{t,j}}+e^{-\alpha_{t,j}}-2}{2}\right)\left(\sum_{i\in S_+} L_{t-1,j}(i) + \sum_{i\in S_-\cup S_{-+}} e^{\Delta_{t-1,j}(i,h_{t,j}(i))}\right)\end{aligned}$$

Using the result we obtained in (Eq. 2.2) and the fact that $e^{\Delta_{t-1,j}(i,h_{t,j}(i))} \leq L_{t-1,j}(i)$, we obtain the following lower bound of the loss drop:

$$\begin{aligned}
\mathcal{L} &= \sum_i L_{t-1,j}(i) - \sum_i L_{t,j}(i) \\
&\geq \left(\frac{e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}}{2}\right) \delta_{t,j} \sum_i L_{t-1,j}(i) \\
&- \left(\frac{e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2}{2}\right) \left( \sum_{i \in S_+} L_{t-1,j}(i) + \sum_{i \in S_- \cup S_{-+}} L_{t-1,j}(i) \right) \\
&\geq \left(\frac{e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}}{2}\right) \delta_{t,j} \sum_i L_{t-1,j}(i) - \left(\frac{e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2}{2}\right) \sum_i L_{t-1,j}(i) \\
&\geq \left(\frac{e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}}{2} \delta_{t,j} - \frac{e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2}{2}\right) \sum_i L_{t-1,j}(i)
\end{aligned}$$

Hence the loss at round $t$, $\sum_i L_{t,j}(i)$, is at most a factor $1 - \frac{1}{2}\left(e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}\right)\delta_{t,j} + \frac{1}{2}\left(e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2\right)$ of the loss in round $t-1$.

□

We thus proved that, in each view $m$, the training error (cost) decreases. Using the result of the previous theorem and tuning $\alpha_{t,j}$ to $\frac{1}{2}\ln\frac{1+\delta_{t,j}}{1-\delta_{t,j}}$, we get the following bound on the loss of the classifier, $H_j$ defined as the weighted combination of weak classifiers learnt in view $j$ after $T$ iterations:

$$\epsilon_{T,j} \leq (K-1)\prod_{t=1}^{T}\sqrt{1-\delta_{t,j}^2} \leq (K-1)\exp\left\{-\frac{1}{2}\sum_{t=1}^{T}\delta_{t,j}^2\right\} \tag{2.3}$$

On the one hand, this results shows that, as long as the edges of the weak classifiers are positive, then the loss of the combined classifier decreases after each iteration. On the other, it means that, for a classifier $h_{t,j}$, its edge should be the highest value for which the weak learning condition Eq. 2.1 still holds. More precisely:

$$\begin{aligned}
\delta_{t,j} &= \sup\left\{\gamma : \mathbf{D}_{t,j} \cdot \mathbb{I}_{h_{t,j}} \leq \mathbf{D}_{t,j} \cdot \mathbf{U}_\gamma\right\} \\
&= \sup\left\{\gamma : \mathbf{D}_{t,j} \cdot \mathbb{I}_{h_{t,j}} \leq -\gamma \sum_{i=1}^{m}\sum_{l \neq y_i} e^{\Delta_{t-1,j}(i,l)}\right\}.
\end{aligned}$$

The last part suggests that the best value for $\delta_{t,j}$ is edge $\gamma$ is the one that verifies the equality, that is:

$$\gamma : \mathbf{D}_{t,j} \cdot \mathbb{I}_{h_{t,j}} = -\gamma \sum_{i=1}^{m}\sum_{l \neq y_i} e^{\Delta_{t-1,j}(i,l)} \Rightarrow \delta_{t,j} = \frac{-\sum_{i=1}^{m} \mathbf{D}_{t,j}(i, h_{t,j}(i))}{\sum_{i=1}^{m}\sum_{l \neq y_i} \mathbf{D}_{t,j}(i,l)}.$$

We have thus derived the expression given in Algorithm 3 for the computation of $\delta_{t,j}$. Note that the edge $\delta_{t,j}$ is positive only if the classifier $h_{t,j}$ verifies the weak condition.

### 2.3.2 Bounding the whole empirical loss

At each step $t$ of the Algorithm 3, one classifier is selected among $v$ weak classifiers, where $v$ is the number of views, that is, the space of weak hypothesis $\mathcal{H}$ is limited to $\{h_{t,1}, \cdots, h_{t,v}\}$. The chosen classifier $h_t$ is the one that achieves the largest edge on the global cost matrix $\mathbf{D}_t$, or simply put, the one that has the smallest error on the training sample weighted by the global cost matrix.

Bounding the drop of the loss after each iteration is done in a similar fashion as in Theorem 1. The main differences consist in

— replacing $h_{t,j}, \delta_{t,j}, \alpha_{t,j}$ and $\mathbf{D}_{t,j}$ by $h_t, \delta_t, \alpha_t$ and $\mathbf{D}_{t,G}$ in the theorem and in the proof;
— considering only two cases, $S_+$ the examples correctly classified by $h_t$ and $S_-$ the examples misclassified by $h_t$.

Moreover, the result given in Equation 2.3 (and the choice of $\delta_t$) remains valid:

$$\epsilon_T \leq (K-1) \prod_{t=1}^{T} \sqrt{1-\delta_t} \leq (K-1) \exp\left\{ -\frac{1}{2} \sum_{t=1}^{T} \delta_t^2 \right\} \tag{2.4}$$

where $\epsilon_T = \sum_{i=1}^{m} \sum_{l \neq y_i} \exp(f_t(i,l) - f_t(i,y_i))$.

Since the hypothesis space is quite limited, it is possible that none of the classifiers verify the weak learning condition on $\mathbf{D}_t$. In such cases, one may either chose to halt the learning process or, if the weak classifiers are trained by resampling, reiterate the learning process on the views till at least one of the classifiers achieves a sufficient edge.

### 2.3.3 Results in generalization

In this section we show that the generalization error of the final hypothesis learnt by Mumbo after $T$ iterations, can be bound and that this bound converges towards 0 with the number of iterations.

The generalization error of a classifier $f$ is defined as the probability to misclassify a new example. For multi-class algorithms such as AdaBoost.MR, Schapire et al. (1998) show how to bound the generalization error of the final hypothesis as a function depending on the margins of the learning examples. We first recall the definitions and the bound on the generalization error. Then we extend these results to Mumbo.

#### Generalization Error for Multi-class 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 (called also base 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 $\mathbb{I}[h_t(x) = l]$, the weak classifiers used in Algorithm 3.

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 an 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 proof of the following theorem in Schapire et al. (1998):

**Theorem 2** *(Schapire et al.[1998]) Let $\mathcal{D}$ be a distribution over $X \times Y$, and let $S$ be a sample of $m$ examples chosen independently at random according to $\mathcal{D}$. Assume that the base-classifier 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 $\theta > 0$ :*

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

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

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

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 $\theta$. In the last part of this section, we use $\epsilon_\theta(f,S)$ instead of $\mathbf{P}_S[margin(f,x,y) \leq \theta]$. Finally, the second term in the theorem is a complexity penalization cost.

### Mumbo

The previous theorem holds for every voting method using multi-class classifiers as weak classifiers, thus it holds also for Mumbo since his final hypothesis is $H_T(x) = \underset{l \in 1,2,...,K}{argmax}\ f_T(x,l)$, where :

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

Note that this new definition on $f_T$ is simply a normalized version of the original $f_T$ given in Algorithm 3.

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 given in Vapnik (1998) that $d_{\mathcal{H}} = min\{d_{\mathcal{H}_1},...,d_{\mathcal{H}_m}\}$.

We have yet to show 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 second term in Theorem 2 is a constant.

The following lemma, based on Lemma 4.1 from Janodet et al. (2009), gives an upper bound for $\epsilon_\theta(f_T,S)$.

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

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

*Proof.* Let $l = \underset{y' \neq y}{argmax} f(x, y')$. For readability reasons, we write $\sum_{t=1}^{T}$ as $\sum_t$ and $\prod_{t=1}^{T}$ as $\prod_t$.

By the definitions of the margin and $f$, we have:

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

Hence,

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

Let $A_i = -\left( \sum_t \alpha_t h_t(x_i, y) - \sum_t \alpha_t h_t(x_i, l) \right)$ and $B = \theta \sum_t \alpha_t$. 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, $\epsilon_\theta(f_T, S) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(A_i) \exp(B)$.

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

Using the bound on the empirical loss, we obtain:

$$\begin{aligned} \epsilon_\theta(f_T, S) &\leq \frac{1}{m} \exp(\theta \sum_t \alpha_t)(K-1) \prod_{t=1}^{T} \sqrt{1 - \delta_t^2} \\ &\leq \frac{1}{m} \exp(\theta \sum_t \tfrac{1}{2} \ln(\tfrac{1+\delta_t}{1-\delta_t}))(K-1) \prod_{t=1}^{T} \sqrt{1 - \delta_t^2} \\ &\leq \frac{1}{m} \prod_t (\tfrac{1+\delta_t}{1-\delta_t})^{\frac{\theta}{2}} (K-1) \prod_{t=1}^{T} \sqrt{1 - \delta_t^2} \\ &\leq \frac{(K-1)}{m} \left( \prod_t (1+\delta_t)^{\frac{1+\theta}{2}} (1-\delta_t)^{\frac{1-\theta}{2}} \right) \end{aligned}$$

□

The precedent lemma gives a bound on the empirical margin error. As it was shown in Schapire et al. (1998), if $\theta < \delta/2$, then $(1+\delta_t)^{\frac{1+\theta}{2}} (1-\delta_t)^{\frac{1-\theta}{2}} < 1$. Using this result and the precedent lemma, we 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 o with the number of iterations, if the edge of the weak hypothesis selected at each iteration is $> 2\theta$.*

This last theorem 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.

## 2.4 Experimental results on artificial data

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

### 2.4.1 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 $y$ are generated along a gaussian distribution $\mathcal{G}[m_{y,v}, \sigma_{y,v}]$, one per view $v$. In order to simulate weak views, we introduce some uncertainty in the description of the examples of both classes:

— In each view, the distributions of classes may overlap: some examples are likely to belong to both classes [1]. This allows to increase the error of the Bayes classifier, thus lowering the strength of the view.

— In each view, some examples are generated using a uniform distribution, the same for both classes. We denote by $\eta$ the rate of example generated by such a distribution.

One major view is generated, with various values of $\eta_M$. 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 2.2 presents an example of a learning sample with 20 examples per class. Each example of the learning sample is represented by three views: the major view is on top, with $\eta_M = 0.38$; other views are minor ones, with $\eta_m = 0.56$. The parameters of the examples' distribution within each class are represented by the red and green ellipses. The same example is pointed out in each view, in order to illustrate the distribution of information among views.

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 $v$, the more $\gamma_v = 1 - \epsilon(h_{Bayes})$ is low on that view. Such a sample generation process was designed in order to fit the hypothesis leading 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, since information is 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 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 Mukherjee and Schapire (2011). Results are the mean of 10 experiments: one experiment is made up of (1) the generation of learning and test samples, (2) the learning process, and (3) the evaluation process.

As said in introduction, Mumbo was designed as an alternative way to fuse classifiers. We thus compare it with two naive other methods of fusion, and with Adaboost:

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

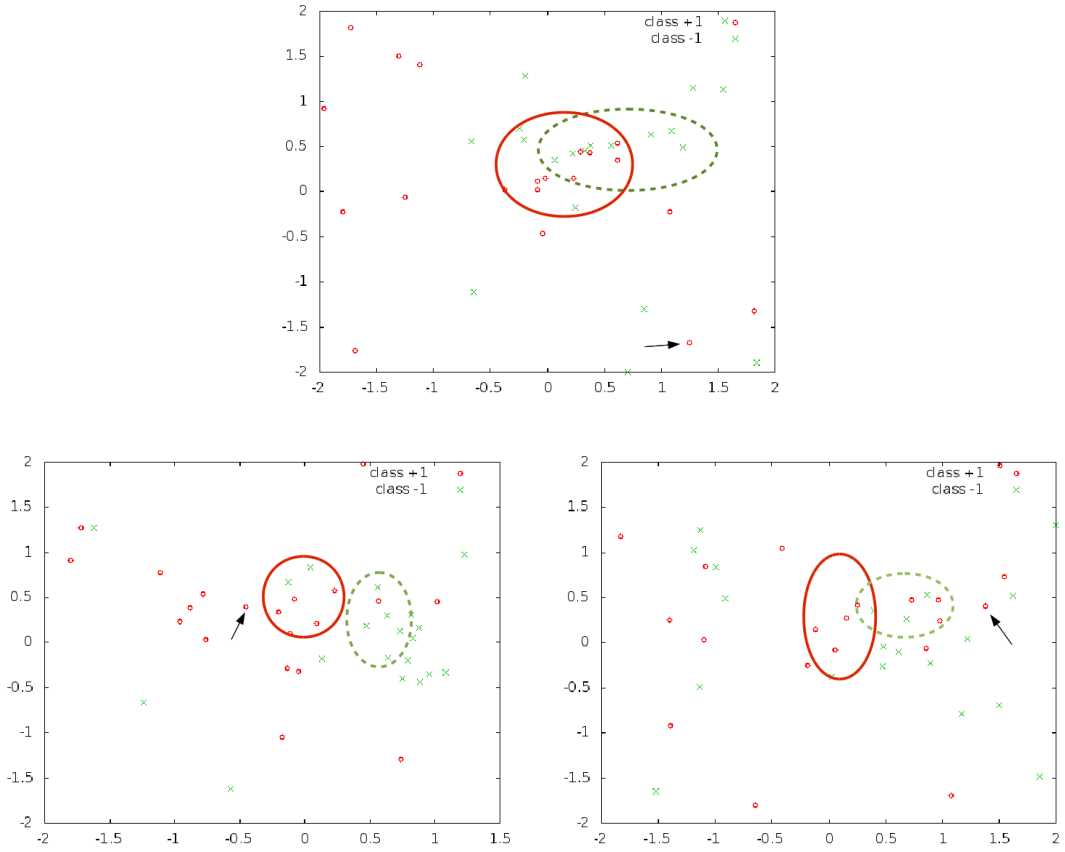



Figure 2.2 – *An example of the multi-view artificially generated data for MuMBo.*

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;
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.

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

### 2.4.2 Results

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

#### Illustration of boosting properties

Figure 2.3 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 $|S| = 80$ and $\eta_M = 0.38$, but the same outlines of behaviours are observed whatever the parameters are ($|S|$ from 20 to 200, and $\eta_M$ from 0 to 0.5).

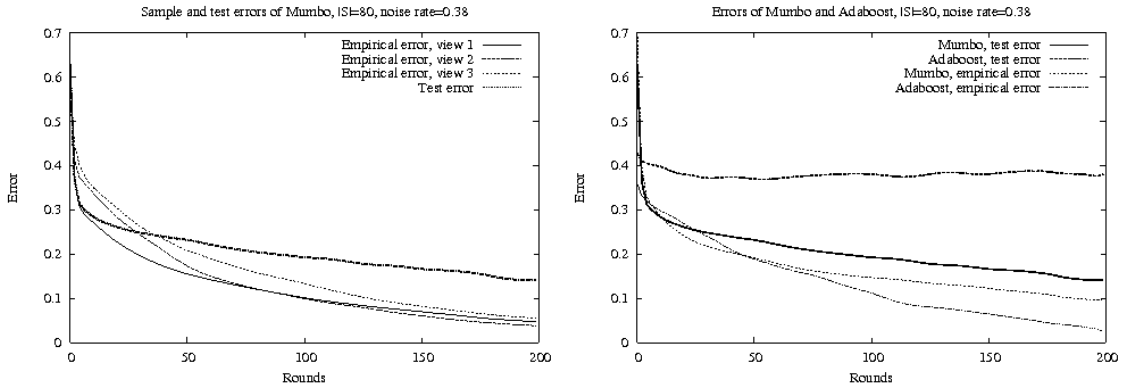



Figure 2.3 – *Empirical and test errors of Mumbo (left), and Mumbo vs. Adaboost early fusion*

The bad results of Adaboost are not surprising. Indeed, it processes examples on only one view that concatenates the three 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 $\eta_M$ 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 2.3. It empirically proves that Mumbo actually encourages views to cooperate.

### Comparison with other approaches

Table 2.1 compares Mumbo with basic early and late fusion approaches (where fused classifiers are RBF SVM), with different values of $\eta_M$ and different sizes of the learning sample. 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

| $\lvert S\rvert = 80$, $\eta_M$ | 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 |
| $\lvert S\rvert = 120$, $\eta_M$ | 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 |

Table 2.1 – *Comparison of Mumbo with early fusion and late fusion (where base classifiers are RBF SVM). Note that results of Adaboost are given after 200 iterations (like Mumbo): raw results show that it obtains a slightly better results after about 50 iterations, then tends to overfit.*

distributions overlaps). Yet Mumbo is better for it processes the collaboration among views, thus leading each view to focus on the examples disrupted in other views.

However, the learning time of Mumbo is many longer than the learning time of Late SVM (about 200 times slower). 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).

## 2.5 A multi-view approach to DECODA

The task we consider in the DECODA project is that of phone calls classification, as described in Section 2.2. In Section 1.5.2, we described the categorization task as a multi-view learning problem, with views based on the content of the content, its prosody and the interaction between the speakers. We recall here the five views:

1. word transcription of the user's turns,
2. word transcription of the operator's turns,
3. turn taking description between the operator and the users,
4. dialog and average turn duration, % of speech for each speaker, speech rate (in letters per second),
5. bag of named entities extracted from the word transcriptions.

Two experimental setting have been tested:

1. *test(gold)*: the classification performance of each method is evaluated on features obtained on the reference transcriptions (*gold*) of the test corpus
2. *test(ASR)*: in this case the features are obtained on the ASR transcriptions.

To compare the results of MuMBo with a standard classification approach, we use the *Icsiboost* Favre et al. (2007) implementation of AdaBoost. Firstly, to check the strength of each view, we present in Table 2.2 the classification error rate obtained on the two experimental settings with AdaBoost. As expected the two lexical views ($V1$ and $V2$) clearly outperform the other views.

| setting | error | $V1$ | $V2$ | $V3$ | $V4$ | $V5$ |
|---|---|---|---|---|---|---|
| *test(gold)* | # | 40 | 43 | 107 | 104 | 82 |
| | % | 26.3 | 28.3 | 70.4 | 68.4 | 53.9 |
| *test(ASR)* | # | 65 | 55 | 107 | 104 | 82 |
| | % | 42.7 | 36.2 | 70.4 | 68.4 | 53.9 |

Table 2.2 – *Classification errors with AdaBoost on the five views ($V1 \ldots 5$) without fusion (152 examples in the test sample).*

Three fusion algorithms between the views are compared in Table 2.3:
— AdaBoost with late fusion (fusion of the decisions taken independently by each view);
— AdaBoost with early fusion (all the views are merged together);
— and our multiview algorithm MuMBo.

The weak classifiers used in the three cases are decision stumps and the algorithms ran for 1000 iterations. We noticed that adding the *weak* views $V3$, $V4$, $V5$ to the strong ones improves the classification performance of all the algorithms, except in the case of early AdaBoost on *test(gold)*: it means that weaker views still carry some useful information. Late or early fusion don't have a strong impact on the AdaBoost performance in our experiments.

| Algo | Errors | *test(gold)* | *test(ASR)* |
|---|---|---|---|
| AdaBoost(early) | # | 37/152 | 49/152 |
| | % | 24.3 | 32.2 |
| AdaBoost(late) | # | 37/152 | 46/152 |
| | % | 24.3 | 30.3 |
| **MuMBo** | # | **32/152** | **38/152** |
| | % | **21.1** | **25.0** |

Table 2.3 – *Classification errors for MuMBo and Adaboost (late fusion and early fusion) on the two experimental settings*

As we can see *MuMBo* gives better results than AdaBoost on both settings, especially when the classification process is applied on the noisy ASR data. This is particularly encouraging as our main motivation for using MuMBo was to be able to spread the classification weights more equally on the different views in order to be more robust when one view fails, as this is the case in the setting *test(ASR)* where the main lexical views are affected by the high WER of the ASR transcriptions.

| **MuMBo** | view 1 | view 2 | view 3 | view 4 | view 5 |
|---|---|---|---|---|---|
| *test(gold)* | 393 | 429 | 16 | 92 | 70 |
| *test(ASR)* | 393 | 429 | 16 | 92 | 70 |
| **AdaBoost** | view 1 | view 2 | view 3 | view 4 | view 5 |
| *test(gold)* | 418 | 452 | 18 | 80 | 48 |
| *test(ASR)* | 418 | 452 | 18 | 80 | 48 |

Table 2.4 – *# of selection for each view with MuMBo and Adaboost on the DECODA corpus*

Table 2.4 shows that MuMBo encourages the cooperation between the different views as the number of selections of the stronger views ($V1$ and $V2$) is smaller for MuMBo than for AdaBoost. This cooperation allows the classification weights to be distributed more equally among the views.

## 2.6 Generalization of MuMBo

In this section we propose two possible improvements, or rather generalizations, of MuMBo. The first improvement is related to the computation of the cooperation coefficients, while the second one concerns the choice of the unique classifier at each iteration.

### 2.6.1 On more general cooperation coefficients

The cooperation coefficients, noted $d$ is Algorithm 3, are one of the tools used to enforce the collaboration between the views. Their goal is to promote for each view the best examples, that is, the examples that are easier to recognize within that view. Another way to see this, is to consider these coefficients as flags that designate for each view the examples that should be updated. In Algorithm 3, the cooperation coefficients take boolean values and they are computed as shown in Section 2.2.2. Recall that for a given view, the weight of an example is updated — that is, its cooperation coefficient for that view is set to 1 — only if the weak classifier correctly recognizes the example in this view, or if ti is not recognized in any view. This constitutes at the same time the main advantage behind the coefficients' choice: they are easy to interpret. On the other hand, the main inconvenience is that they are not flexible, since we

have the choice between only two values. Having a larger choice can be quite useful in some cases.

Suppose that multiple views have been defined for a given problem and that a small number of training examples — we'll note them $S_v$ — are recognized within only one view. Using the binary update rule, the weights of these examples change only on the view that recognizes them, while the other views will focus less on them. Now, suppose that for whatever reason, during the early learning phase, only the classifiers from the other views are selected as the unique classifiers. This means that the resulting classifier, made up of the combination of the unique classifiers, will poorly recognize the examples of $S_v$, and these examples will only be taken into consideration in later steps of the training procedure.

As a second example, take the case where, for a given view, an example is repeatedly misclassified, even though it might not be one of the hardest examples for that view. This may due to the fact that the learnt classifiers are weak ones, performing slightly better than random guessing. The binary update does not change the weight of that example for the considered view, hence it is taken into consideration only in later stages.

The two previous examples show that the binary update rule may not be the perfect choice. Indeed, in the first example, it would be preferable to slightly update the weight of the examples $S_v$ in all the views, so that a unique classifier recognizing these examples is chosen earlier in the learning phase. The same is valid for the second example also, since it would help the view to recognize that example earlier. In order to overcome these shortcomings, we propose to consider a more general form of the cooperation coefficients. Instead of considering only boolean values, these new coefficients take their values in $[0,1]$. The values of these coefficients are chosen in a similar way as shown in Section 2.2.2: if an example is correctly classified, or is misclassified by all the views, its coefficient is 1. The main difference is that in the case where an example is misclassified in a view, but correctly classified in (at least) one other view, its coefficient in that view might not be 0.

Some possible choices for the new coefficients are the following:

— $d_{t,j}(i) = \max\{\mathbb{I}\Big(h_{t,j}(i) = y_i\Big), (v-p)/v\}$

— $d_{t,j}(i) = \max\{\mathbb{I}\Big(h_{t,j}(i) = y_i\Big), \exp\Big((\mu - p/v)^2\Big)\}$, for some $\mu \in [0,1]$

— $d_{t,j}(i) = \max\{\mathbb{I}\Big(h_{t,j}(i) = y_i\Big), |0.5 - p/v|\}$

where $v$ is the number of views and $p = \Big|\{c : h_{t,c}(i) \neq y_i\}\Big|$, the number of classifiers that misclassify $x_i$.

The main interest of selecting coefficients that take values in $[0,1]$ is that Theorem 1 still holds. This implies that all the results shown in Section 2.3 which depend on that theorem, are valid even for the continuous coefficients. We give here its formulation for the continuous case.

**Corollary 1** (bounding the empirical loss in view $j$ in the continuous case) *For a given view $j$, suppose the cost matrix $\mathbf{D}_{t,j}$ is chosen as in the Algorithm 3, the returned classifier $h_{t,j}$ verifies the weak learning condition Eq. 1.5 for $\mathbf{D}_{t,j}$ and $\mathbf{U}_{\delta_{t,j}}$ and, that for each example, the cooperation coefficient takes its value in $[0,1]$.*

*Then choosing a weight $\alpha_{t,j} > 0$ for $h_{t,j}$ makes the error at time $t$, at most a factor*

$$\tau_{t,j} = 1 - \frac{1}{2}\left(e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}\right)\delta_{t,j} + \frac{1}{2}\left(e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2\right)$$

*of the loss before choosing $\alpha_{t,j}$, where $\delta_{t,j}$ is the edge of $h_{t,j}$.*

*Proof.* The proof of this corollary is the same as the one given in Theorem 1. We refer the reader to Theorem 1 for the definition of the various quantities utilized here. The main difference is the computation of the drop in loss for the examples in $S_{-+}$, examples misclassified in view $j$, but correctly classified in at least one of the other views.

$$\begin{aligned}
\mathcal{L}_{-+} &= \sum_{i \in S_{-+}} e^{f_{t-1,j}(i,h_{t,j}(i)) - f_{t-1,j}(i,y_i)} - \sum_{i \in S_{-+}} e^{f_{t,j}(i,h_{t,j}(i)) - f_{t,j}(i,y_i)} \\
&= \sum_{i \in S_{-+}} e^{f_{t-1,j}(i,h_{t,j}(i)) - f_{t-1,j}(i,y_i)} - \sum_{i \in S_{-+}} e^{f_{t-1,j}(i,h_{t,j}(i)) + \alpha_{t,j} d_{t,j}(i) - f_{t-1,j}(i,y_i)} \\
&\geq \sum_{i \in S_{-+}} e^{f_{t-1,j}(i,h_{t,j}(i)) - f_{t-1,j}(i,y_i)} - \sum_{i \in S_{-+}} e^{f_{t-1,j}(i,h_{t,j}(i)) + \alpha_{t,j} - f_{t-1,j}(i,y_i)} \\
&= -\left(e^{\alpha_{t,j}} - 1\right) \sum_{i \in S_{-+}} e^{f_{t-1,j}(i,h_{t,j}(i)) - f_{t-1,j}(i,y_i)} \\
&= -\left(e^{\alpha_{t,j}} - 1\right) \sum_{i \in S_{-+}} e^{\Delta_{t-1,j}(i,h_{t,j}(x_i))}
\end{aligned}$$

So, computing the drop in loss $\mathcal{L} = \mathcal{L}_+ + \mathcal{L}_- + \mathcal{L}_{-+}$, we have:

$$\begin{aligned}
\mathcal{L} &= \left(1 - e^{-\alpha_{t,j}}\right) \sum_{i \in S_+} L_{t-1,j}(i) - \left(e^{\alpha_{t,j}} - 1\right) \sum_{i \in S_- \cup S_{-+}} e^{\Delta_{t-1,j}(i,h_{t,j}(i))} \\
&= \left(\frac{e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}}{2}\right) \left(\sum_{i \in S_+} L_{t-1,j}(i) - \sum_{i \in S_- \cup S_{-+}} e^{\Delta_{t-1,j}(i,h_{t,j}(i))}\right) \\
&- \left(\frac{e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2}{2}\right) \left(\sum_{i \in S_+} L_{t-1,j}(i) + \sum_{i \in S_- \cup S_{-+}} e^{\Delta_{t-1,j}(i,h_{t,j}(i))}\right)
\end{aligned}$$

The rest of the proof is the same as in Theorem 1. □

Bottom line, choosing cooperation coefficients in $[0, 1]$ gives more flexibility to the algorithm, while at the same time keeping the theoretical properties.

### 2.6.2 On the choice of the unique classifier

The choice of the unique classifier at each iteration, given in Algorithm 3, depends on the cost obtained from the classifiers on the global cost matrix: the chosen classifier is the one that achieves the smallest cost, that is, the largest drop in loss. In Section 2.2, we argued that this choice is mainly due to the fact that the boosting process can be seen as a gradient descent in the function space. Here we propose several methods for improving the choice of the unique classifier made by MuMBo.

In Section 2.3.2, we slightly touched upon one of the weak points of MuMBo: the classifier space for the choice of the unique classifier is quite limited, since it contains only the classifiers learnt on each view. The direct consequence is that it might be necessary to re-train the classifiers on all the views, since none of the learnt classifiers might verify the weak learning condition Eq. 1.5 for the global cost matrix. It means that the weak learning algorithms are asked to train classifiers strong enough to satisfy the weak learning condition for two different cost matrices: the global cost matrix and the one of the view associated to the algorithm. An improvement of MuMBo

would then consist in choosing stronger learning algorithms; although one may argue that this kinda defeats the purpose of using a boosting procedure. Another possible improvement is that of considering a larger set/space of classifiers, built from the weak classifiers learnt on the views. Such a space is, for instance, the space containing all the possible linear combination of the weak classifiers. The underlying idea is that of choosing two or more weak classifiers at each iteration, compared to the unique classifier chosen by MuMBo:

$$h_t(x) = \text{argmax}_{l \in Y} \sum_{j=1}^{v} \mathbb{I}(h_{t,j}(x) = l)\beta_{t,j}.$$

The coefficients $\beta_{t,j}$ can be computed in various ways: using optimization problems, prior information on the views, heuristics, etc. However the most straightforward way would be to compute the coefficients as $\frac{1}{2} \ln \frac{1+\delta_{t,j}}{1-\delta_{t,j}}$, where $\delta_{t,j}$ is the edge of the classifier learnt on view $j$ on the global cost matrix. Note that in this case MuMBo becomes similar to 2-Boost Janodet et al. (2009), except for the fact that MuMBo trains its weak classifiers on different distributions.

Obviously other combinations can be considered, as long as at least one of the classifiers contained in the final classifier space satisfies the weak learning condition.

Other possible improvements of MuMBo are related to the *actual choice* of one unique classifier. In Algorithm 3, we proposed to select the classifier with the smallest error/loss on the cost matrix, a greedy criterion that ensures a small empirical error on the training set. However, other measures can be considered, which may hopefully make an impact on the generalization error. Kuncheva and Whitaker (2003) claim that the accuracy of an ensemble learning method is closely related to the diversity of the classifiers that make up the ensemble. A selection criterion would than be to choose at each iteration, the classifier that optimizes the diversity of the ensemble made up of the previously selected classifiers. However, the optimization of the diversity of an ensemble remains an open research problem: the "good" degree of diversity depends on the learning problem.

Selection criterions can also be based on prior informations on the views, such as, defining the proportion of weak classifiers selected from each view, or giving priority to view that correctly classify an *interesting* subspace of the input space. In the latter case, the subspace can also be the one defined by instances of the same class, that is, at each iteration, the selected view is the one that better recognizes one of the classes.

Similarly to the improvements proposed in Section 2.6.1, the advantage of these improvement is that all the theoretical properties proven in the previous sections remain valid, as long as the classifier selected at each iteration satisfies the weak learning condition.

## 2.7 Conclusion for this chapter

### 2.7.1 Related works

In Section 1.3, we presented several multi-view approaches, for different settings, such as semi-supervised learning, active learning, and so on. In the supervised setting, early and late fusion-based approaches are only empirical ways to process the

whole useful information available on samples. On the other hand, the Multiple Kernel Learning (MKL) approaches Lanckriet et al. (2004) (also presented in Section 1.3.2), used to process multi-view samples, is then a costly way to rank the views. As a supervised method, it could be compared to MuMBo, but MKL does not promote the cooperation between views: it is more a way to select the strongest views.

The closest approaches to MuMBo are co-training Blum and Mitchell (1998) and GBoost Saffari et al. (2010), in the semi-supervised setting, and 2-Boost Janodet et al. (2009) and ShareBoost Peng et al. (2011), in the supervised one. Co-training is a multi-view approach in the *semi-supervised setting*, where views iteratively cooperate for producing classifiers that converge to the same final hypothesis. GBoost is a generalization of co-training, where the learning process is based on the boosting framework and the unlabeled data are also used in the training process in order to improve the performances of the views and making them less sensitive to label noise. However MuMBo is different from both these methods, since it works in the supervised setting and it does not assume that the classifiers agree on the same examples. Indeed, MuMBo relies on the disagreement between the views.

MuMBo is closer to 2-Boost and ShareBoost, not only because they are supervised methods, but also because the three methods rely on the boosting framework in order to take advantage of the multi-view nature of the problems. However, the motivations behind MuMBo and the two other methods are not the same. 2-Boost is designed to use 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. Theoretically speaking, though, the generalization error bound of MuMBo is proven in a similar way than that of 2-Boost, integrating the several hypothesis spaces. Also, in Section 2.6, we proposed, as an amelioration of MuMBo, to define the unique classifier as a combination of all the classifiers, which is the case of 2-Boost.

ShareBoost can be seen as a particular case of 2-Boost. They both maintain a single distribution over the training sample and, during the training procedure, trains as many weak classifiers as there are views, but ShareBoost only keeps the best one (whose weighted error is minimal), similarly to MuMBo. The motivation behind this choice is to implement a co-training-like procedure in the supervised setting: during the training process, each view is encouraged to deal with the examples it recognizes best, thus, hopefully, minimizing the number of hard-to-classify examples for the other view. Although this might seem as a similarity with MuMBo, there is no guarantee that the cooperation does take place, since the classifiers learnt on the views are weak ones. Other versions of ShareBoost have been proposed, such as rShareBoost Peng et al. (2011), which randomly chooses the unique classifier at each iteration. rShareBoost tries to find for a given problem, the most appropriate views (think of this as a ranking of the view), thus minimizing the cooperation between them.

Recently, in Kadri et al. (2013) we proposed to tackle the multi-view learning problem through the multi-task perspective. The idea behind the framework proposed therein is to reformulate the multi-view learning problem as a multi-task one, where the goal is to learn a classifier that depends on a multi-task kernel defined over the input space. In the case of multiple views, the input space is made up of all the views input spaces and Kadri et al. (2013) define the kernel matrix as a block kernel matrix, where each block correspond to the kernel matrix between two views. This formulation takes into consideration the within- and between- views information, which is close to the notion of cooperation between views, used in MuMBo.

### 2.7.2 Conclusion

We presented the MuMBo algorithm (Section 2.2), which is a boosting-like method in the setting of multi-view learning, where views are of different strength with regard to a classification task. The idea underlying MuMBo is to encourage the cooperation between stronger and weaker views. To implement this idea, the originality of MuMBo, as a boosting algorithm, 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.

Theoretical results for the boosting properties of MuMBo, within the theoretical framework of Mukherjee and Schapire (2011), were given in Section 2.3: first we showed that the empirical error decreases with the iterations, then using this result, we proved a bound on the generalization error of MuMBo, thus showing that the true risk decreases with the iterations. An interesting property of MuMBo was also proved in 2.3: the empirical error of the combination of all the classifiers in one view decreases with the number of iterations. This means that a boosting process takes place for each view, thus allowing it to focus on the most interesting examples.

As expected theoretically, the boosting usual behavior is observed throughout the experiments in Section 2.4, and the results of MuMBo are very good on synthetic data, generated to simulate real-life data, with noisy data and weak views.These results validate the relevance of the MuMBo algorithm when cooperation among weak views is mandatory for obtaining a strong classifier. Results on empirical and generalization bounds, as proved in Section 2.3, are also observed.

Several improvements of MuMBo were proposed in Section 2.6, both on the choice of the unique classifier at each iteration and the choice of the cooperation coefficients. As a result of these improvements, MuMBo can be seen as a family of multi-view boosting methods, regrouping all methods respecting the conditions given in 2.6.

The works presented here on MuMBo open up different questions leading to future works, both in the theoretical and practical aspect. When presenting the possible improvements in Section 2.6, the bound given in Corollary 1 is obtained by using a lower bound for the case $S_{-+}$, contrary to the bound in Theorem 1, which is more appropriate for the choice of the cooperation coefficients. Future works will be focused on finding better per view bounds depending on these coefficients, and incorporating them in the bound on the empirical error computed from the global cost matrix. This will allow to carry the idea of cooperation from the cost matrix to the empirical error bound, and ultimately to the generalization error bound. However, concerning the latter bound, we think it would be more interesting to find other bounds directly based on the multi-view framework, related to the complementary of the views, or the definitions of strong and weak views, as given in Section 1.3.1.

MuMBo is an algorithm that may be categorized as an *ensemble of 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 Kuncheva and Whitaker (2003). We think that the hypothesis underlying MuMBo promote such a diversity. In that sense, future work will aim at obtaining some theoretical results between diversity and classification accuracy of MuMBo.

Last but not least, based on the success of MKL, future works will center on implementing the concept of selecting appropriate learners for each view in MuMBo. This includes both theoretical and empirical aspects.

# Tackling the imbalanced classes problem through the confusion matrix norm

3



Dealing with the imbalanced classes problem consists in learning a classification model form data, where the distribution of examples among classes is skewed. In such cases, the simple (empirical) error rate is not the right measure to use for quantifying the goodness of a classifier, since it would favor major classes over minor ones. Section 1.4 introduced several measures that are mode adapted for these methods, such as the MAUC, F-measure and cost-weighted error. Before tackling the imbalanced views problem, this chapter proposes a novel framework for (single view) imbalanced learning methods.

In Section 1.4.4, we briefly argued that an interesting error measure would be the norm of the confusion matrix. Based on recent works on the confusion matrix as an error measure by Morvant et al. (2012), Ralaivola (2012), we use the confusion matrix as a starting point for learning imbalanced methods in the supervised setting. A first contribution in this chapter consists in showing in Section 3.2.1 that minimizing the norm of the confusion matrix leads to a common base for most existing imbalanced methods. This base comes in the form of an optimization problem, which depends on loss functions computed over two classes and representing the loss of mistaking one class for the other.

As a second contribution, in Section 3.2.2 and Section 3.2.3,we adapt the aforementioned optimization problem to the boosting framework and derive an extension of AdaBoost.MM (Mukherjee and Schapire (2011)) for the imbalanced classes problem. The booting properties of the new method, called CoMBo, are proven in Section 3.2.4, while Section 3.4 gives empirical results of CoMBo on various UCI datasets. Section 3.5 presents the performances of CoMBo for the imbalanced aspect of DECODA. Finally Section 3.6, wrap it all up with a discussion on the contributions of this chapter and the future works.

The work presented in this chapter was first published in the Proceedings of the 5th Asian Conference on Machine Learning, ACML 2013, Canberra, Australia, Koço and Capponi (2013). The work presented in Section 3.3 was first published in the Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, United Kingdom, Morvant et al. (2012).

## 3.1 The confusion matrix as an error measure

This section deals with the imbalanced classes problem and gives a brief overview of the existing methods, both in the binary multi-class problem. This is followed by the motivations of choosing the confusion matrix as an error measure. We conclude with the formal definition of the confusion matrix and the optimization objective.

### 3.1.1 Introduction and Motivations

In this chapter we tackle a learning problem that is a common occurrence in data coming from real-world scenarios: the imbalanced classes (data) problem. Learning from imbalanced data concerns theory and algorithms that process a relevant learning task whenever data is not uniformly distributed among classes. When facing imbalanced classes, the classification accuracy is not the fair measure to be optimized, as advocated in Fawcett (2006). Accuracy can be quite high in case of extreme imbalanced data: majority classes are promoted, while minority classes are not recognized. Prediction is biased toward the classes with the highest priors. Such a bias gets stronger within the multi-class setting.

As an example of where this problem in encountered and its effects, let us consider two classification problems: automatic diagnosis of a disease and mushrooms categorization. In the first case, the classification problem consists in detecting if a patient is infected with a disease or if he is sane. Obviously (and thankfully) whatever the considered disease may be, it is fairly easier to collect data from sane patients than from infected ones. A simple classifier learnt from the data is the majority vote, which constantly predicts the majority class — here, that a patient is not infected. This classifier has an excellent accuracy (think of the cases where the sane patients

make up 95-99% of the training sample), however the consequences of always predicting *sane* can be quite heavy, especially if the considered disease is life-threatening. In the second case, the task consists of labeling mushrooms as *safe*, *deadly* or *sick* (as in, they make a person ill, without being life-threatening). The distribution over the mushrooms shows that 18% of the mushrooms are safe, 81% make a person sick and only a mere 1% are deadly. Two majority classifiers can be built from this distribution: the first predicts *sick*, with an accuracy of 81%, while the second predicts *not deadly*, with an accuracy of 99%. However much as in the first case, the side-effects of these predictions can be quite catastrophic.

Generalizing from these examples, the goals and motivations of imbalanced classes methods revolve around designing classifiers that better take into account the probability distribution of each class, both when these are known and unknown.

In the binary setting, learning from imbalanced data has been quite studied over the past years, leading to many algorithms and theoretical results He and Garcia (2009). It is mostly achieved by either resampling methods for rebalancing the data over classes (for example Estabrooks et al. (2004)), or/and by dealing with cost-sensitive methods (for example Ting (2000)), which take advantage of prior informations — potentially on the class distributions — in order to penalize the errors made on the training set, or with additional assumptions such as active learning within kernel-based methods (for example: Bordes et al. (2005)). Whatever the approach, the goal stays the same: force the learning method to achieve similar error rates on both classes, by promoting the examples of the minority class.

Learning with imbalanced data within a multi-class or multi-label setting is still an open research problem, which is sometimes adressed through the study of some alternate measures of interest. Most of times, generalizing the binary setting to the multi-class setting is based on the one-vs-all (or one-vs-one) usual trade-off Abe (2004). Recently, Zhou and Liu (2010) proposed a well-founded general rescaling approach of multi-class cost-sensitive learning, that pointed out the need of separating misclassification costs from imbalance rates during the learning phase.

It is worth noticing that specific learning tasks other than classification have been ad- dressed through the optimization of relevant measures within the multi-class imbalanced setting Chapelle and Chang. (2011), Yue et al. (2007), Wang et al. (2012), K. Tang and 2011. (2011). Although related to accuracy, these measures are intended to better model what one would expect to be a a relevant performance measure in such a setting. Meanwhile, the correlations between these alternative measures and accuracy have been partly studied Cortes and Mohri (2004) without any theoretical result so far He and Garcia (2009).

Cost-sensitive methods are usually based on a cost matrix which embeds misclassification costs carrying various meanings. These methods weight the error measure according to the class-based cost of each misclassification, for instance computed from the confusion matrix results Elkan (2001). Indeed, the confusion matrix is one of the most informative performance measures a multi-class learning system can rely on. Among other information, it contains how much the classifier is accurate on one class, and the way it tends to mistake each class for other ones (confusion among classes).

In the binary classification setting, cost-sensitive approaches provide algorithms that somehow optimize a metric computed from the raw confusion matrix where the entries of a row sum up to the number of examples of the class corresponding to the row. Basically, a raw confusion matrix is a square matrix that represents the count of a classifier's class predictions with respect to the actual outcome on some labeled

learning set. Computed from the raw matrix, the *probabilistic confusion matrix* (Section 3.1.2) exhibits an interesting property: the entries of a row sum up to 1, independently from the actual number of examples of the class corresponding to the row. With such property, the confusion matrix constitutes a great equalizer tool that can be used to overcome the class-imbalance problem. Moreover, if one only considers the non-diagonal elements in the matrix, then summing over a row gets quite informing about how the corresponding class is correctly handled by the predictor at hand. This section explores one way to capitalize on this property: we advocate that directly minimizing the norm of the confusion matrix is helpful for smoothing the accuracy among imbalanced classes, so that minority classes are considered as important as majority classes.

The contributions of this section are mainly motivated — and based — by previous works on the confusion matrix Morvant et al. (2012), Ralaivola (2012), where the novelty resides in the use of the norm of the confusion matrix as an error measure. Building upon these motivations, we sketch up a boosting algorithm, based on the multi-class setting proposed in Mukherjee and Schapire (2011), that is ensured to minimize the norm of the confusion matrix, hence minimizing the classification error as proven in Section 3.2. Albeit the proposed methods can be seen as a simple extension of AdaBoost.MM to the imbalanced classes setting, the main contribution of this chapter is the framework given in Section 3.2.1, which is a common base for most cost-sensitive learning methods.

Before tackling the main part of this chapter, we need to give the formal definitions of the confusion matrices and the optimization problem.

### 3.1.2 The confusion matrix as an error measure

When building predictors, most Empirical Risk Minimization based methods optimize loss functions defined over a training sample and depending on the number of misclassified data. This reveals to be a poor strategy when tackling problems with class-imbalance, since minimizing the estimated error may result in overfitting [1] the majority classes.

As previously argued, a common tool used for estimating the goodness of a classifier in the imbalanced classes scenario is the confusion matrix. We give here the probabilistic definitions of the *true confusion matrix* and the *empirical confusion matrix*.

**Definition 6** (True and empirical confusion matrices) *The true confusion matrix* $\mathbf{A} = (a_{l,j}) \in \mathbb{R}^{K\times K}$ *of a classifier* $h : X \leftrightarrow \{1,\dots,K\}$ *over a distribution* $\mathcal{D}$ *is defined as follows:*

$$\forall l, j \in \{1,...,K\},\ \mathbf{a}_{l,j} \overset{def}{=} \mathbb{E}_{\mathbf{x}|y=l}\mathbb{I}\big(h(\mathbf{x}) = j\big) = \mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}(h(\mathbf{x}) = j|y = l).$$

*For a given classifier* $h$ *and a sample* $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \sim \mathcal{D}$*, the empirical confusion matrix* $\mathbf{A}_S = (\hat{a}_{l,j}) \in \mathbb{R}^{K\times K}$ *of* $h$ *is defined as :*

$$\forall l, j \in \{1,...,K\},\ \hat{\mathbf{a}}_{l,j} \overset{def}{=} \sum_{i=1}^{m} \frac{1}{m_l}\mathbb{I}(h(\mathbf{x}_i) = j)\mathbb{I}(y_i = l).$$

An interesting property of these matrices is that the entries of a row sum up to 1, independently from the number of examples contained in the corresponding class. Moreover, the diagonal entries represent the capability of a classifier $h$ to recognize

1. Once again, we abuse the language and use the term *overfit* to refer to methods and/or classifiers that recognize only a few classes.

the different classes, while the non-diagonal entries represent the mistakes of the classifier.

In this chapter, and in the following sections, we propose a framework which makes use of the confusion matrix as an error measure in order to learn predictors for the imbalanced classes case. As such, we need to redefine the confusion matrices, so that only the mistakes of the classifier are taken into consideration. This is done by keeping the non-diagonal entries and zeroing the diagonal ones:

**Definition 7** *For a given classifier $h$, we define the empirical and true confusion matrices of $h$ by respectively $\mathbf{C}_S = (\hat{\mathbf{c}}_{l,j})_{1\leq l,j\leq K}$ and $\mathbf{C} = (\mathbf{c}_{l,j})_{1\leq l,j\leq K}$ such that for all $(l,j)$:*

$$\hat{\mathbf{c}}_{l,j} \overset{def}{=} \begin{cases} 0 & \text{if } l = j \\ \hat{\mathbf{a}}_{l,j} & \text{otherwise,} \end{cases} \qquad \mathbf{c}_{l,j} \overset{def}{=} \begin{cases} 0 & \text{if } l = j \\ \mathbf{a}_{l,j} & \text{otherwise.} \end{cases} \tag{3.1}$$

These new definitions take into consideration only the mistakes of the classifier, hence the interest of keeping the non-diagonal entries and zeroing the diagonal ones.

Let $\mathbf{p} = [P(y = 1), ..., P(y = K)]$ be the vector of class priors distribution, then it is easy to see that:

$$R(h) \overset{def}{=} P_{(x,y)\sim\mathcal{D}}(h(x) \neq y) = \|\mathbf{pC}\|_1, \tag{3.2}$$

where $\|\cdot\|_1$ is the $l1$-norm. This result means that it is possible to retrieve the true error rate of $h$ from its confusion matrix.

Due to the definition of the confusion matrix in definition 7, one may notice that if, for a given classifier $h$, the norm of the confusion matrix is 0, then the classifier $h$ is perfect, as in it makes no mistakes. This implies that an interesting estimator for the goodness of a classifier is the norm of the confusion matrix: the closer it is to 0, the better the learnt classifier performs. Here the selected norm is the $l2$-norm, also called the operator norm, which is a generalization of vector $p$-norms to matrices and it is defined as follows:

$$\|\mathbf{C}\| \overset{def}{=} \max_{\mathbf{v}\neq 0} \frac{\|\mathbf{Cv}\|_2}{\|\mathbf{v}\|_2} \overset{def}{=} \sqrt{\lambda_{max}(\mathbf{C}^*\mathbf{C})},$$

The interest of choosing the operator norm as an error measure is that in the binary setting, the operator norm of the confusion matrix is:

$$\|\mathbf{C}\| = \max\{FN_{rate}, FP_{rate}\},$$

where $FN_{rate}$ and $FP_{rate}$ are defined in Section 1.4.3. This means that minimizing the operator norm of the confusion matrix is equivalent to minimizing the error rate on the least recognized class. If this is done using an iterative process, then minimizing the error on the least recognized class at each iteration aim to achieve roughly the same error rate for both classes.

Moreover, using the definition of the operator norm, Equation 3.2 and the equivalence between norms in finite dimensions, we obtain the following result on the link between the operator norm and the true risk:

$$R(h) = \|\mathbf{pC}\|_1 \leq \|\mathbf{p}\|_1\|\mathbf{C}\|_1 = \|\mathbf{C}\|_1 \leq \sqrt{K}\|\mathbf{C}\| \tag{3.3}$$

Equation 3.2 and Equation 3.3 together imply that minimizing the norm of the confusion matrix can be a good strategy in order to have a small risk. Our aim is thus to find a classifier $\hat{h}$ that verifies the following criterion:

$$\hat{h} = \underset{h\in\mathcal{H}}{\operatorname{argmin}} \|\mathbf{C}\| \tag{3.4}$$

## 3.2 A boosting approach to the minimization of the confusion matrix norm and its theoretical properties

In this section, we bound the operator norm of the matrix, which allows us to make the connection between the confusion matrix and the usual methods for imbalanced classes problems. In a second time, we make use of the bound on the confusion matrix in order to derive a boosting method for imbalanced classes based on a recent multi-class boosting framework.

### 3.2.1 Bounding the confusion matrix

The result given in Equation 3.4 minimizes the operator norm of the true confusion matrix, but it is difficult to use in a practical case, since the underlying distribution $\mathcal{D}$ is unknown. A popular way to overcome this difficulty is to make use of the empirical estimation of the confusion matrix. Theorem 1 in Ralaivola (2012) gives the relation between the *true* and the *estimated norm* of the confusion matrix. [2] We recall here a particular formulation of this theorem applied to the supervised learning setting, where the considered loss is the indicator function $\mathbb{I}$.

**Corollary 2** *For any $\delta \in (0;1]$, it holds with probability $1-\delta$ over a sample $S_{(\mathbf{x},y)\sim\mathcal{D}}$ that :*

$$\|\mathbf{C}\| \leq \|\mathbf{C}_S\| + \sqrt{2K \sum_{k=1}^{K} \frac{1}{m_k} \log \frac{K}{\delta}},$$

*where $\mathbf{C}_S$ is the empirical confusion matrix computed for a classifier $h$ over $S$.*

The previous corollary suggests that our goal may boils down to minimizing the empirical norm of the confusion matrix, which is fairly similar to the other optimization problems, where empirical error measures are considered. Unfortunately due to the nature of the considered confusion matrix, finding the analytical expression of the norm reveals to be quite challenging. This is why we propose to optimize an upper bound of $\|\mathbf{C}_S\|^2$ instead.

$$\|\mathbf{C}_S\|^2 = \lambda_{max}(\mathbf{C}_S^*\mathbf{C}_S) \leq Tr(\mathbf{C}_S^*\mathbf{C}_S)$$

The matrix $\mathbf{C}_S^*\mathbf{C}_S$ is positive semi-definite, meaning that all its eigenvalues are positive. Since the trace of a matrix is equal to the sum of all its eigenvalues, we choose to upper-bound the norm of the confusion matrix by the trace of $\mathbf{C}_S^*\mathbf{C}_S$. We can now focus on the value of $Tr(\mathbf{C}_S^*\mathbf{C}_S)$.

$$\begin{aligned} Tr(\mathbf{C}_S^*\mathbf{C}_S) &= \sum_{l=1}^{K} \mathbf{C}_S^*\mathbf{C}_S(l,l) = \sum_{l=1}^{K}\sum_{j=1}^{K} \hat{\mathbf{c}}_{l,j}^*\hat{\mathbf{c}}_{j,l} \\ &= \sum_{l=1}^{K}\sum_{j=1}^{K} \hat{\mathbf{c}}_{j,l}\hat{\mathbf{c}}_{j,l} = \sum_{l=1}^{K}\sum_{j\neq l} \hat{\mathbf{c}}_{l,j}\hat{\mathbf{c}}_{l,j} \qquad (3.5) \\ &\leq \sum_{l=1}^{K}\sum_{j\neq l} \hat{\mathbf{c}}_{l,j} = \sum_{l=1}^{K}\sum_{j\neq l}\frac{1}{m_l}\sum_{i=1}^{m} \mathbb{I}(y_i = l)\mathbb{I}(H(i) = j) \qquad (3.6) \end{aligned}$$

2. Note that even though the theorem is applied to the online setting in the original paper, it holds as well for the supervised case.

Most of the previous equalities are simple rewritings of the involved terms, while the inequality comes from the fact that the entries of the confusion matrix $\mathbf{C}_S$ are at most 1. We have now an upper bound of the norm depending only on the entries of the confusion matrix.

The drawback of this bound is the presence of the indicator function, which is not optimization friendly. One way to handle this difficulty is to replace the indicator function with loss functions defined over the two classes corresponding to the indices $l$ and $j$ in the bound.

For a given classifier $h$ and a sample $S$, let $\ell_{l,j}(h, \mathbf{x})$ be the loss of $h$ choosing class $j$ over class $l$ for the example $\mathbf{x}$, such that $\forall(\mathbf{x}, y) \in S$ and $1 \leq l, j \leq K$, $0 \leq \mathbb{I}(h(\mathbf{x}) \neq y) \leq \ell_{l,j}(h, \mathbf{x})$. The bound on the confusion matrix can be now expressed using loss functions:

$$\begin{aligned} Tr(\mathbf{C}_S^* \mathbf{C}_S) &\leq \sum_{l=1}^{K} \sum_{j \neq l} \frac{1}{m_l} \sum_{i=1}^{m} \mathbb{I}(y_i = l)\mathbb{I}(H(i) = j) \\ &= \sum_{i=1}^{m} \sum_{j \neq y_i} \frac{1}{m_{y_i}} \mathbb{I}(H(i) = j) \leq \sum_{i=1}^{m} \sum_{j \neq y_i} \frac{1}{m_{y_i}} \ell_{y_i,j}(h, \mathbf{x}_i). \end{aligned} \tag{3.7}$$

The resulting bound is a sum over two penalization terms for all the learning examples: the first term is $\frac{1}{m_{y_i}}$ and the second one represents the newly introduced loss function. The first term is often used in the imbalanced classes problems in order to simulate the resampling effect over the training sample. Indeed, by multiplying the weight of an example with the inverse of the number of examples having the same class, the distribution over the classes becomes more uniform, thus promoting all the classes the same way. Resampling over the examples achieves the same effect, since it aims to retain the same number of examples for each class.

The second term is common in the imbalanced classes setting, where cost-sensitive methods — either per-class based or per-couple of classes based — have been developed. In these methods this term depends on some misclassification cost computed, or simply given, *prior* to the learning phase.

Based on these observations, it follows that the last term of the bound given in Equation 3.7 is quite similar to usual terms in optimization problems for the imbalanced classes setting. As such it can be seen as a common base for most imbalanced classes learning problems, since it encompasses both resampling and penalization, as advocated in Zhou and Liu (2010). Motivated by these results, and as an intermediate conclusion, we strongly think that minimizing the operator norm of the confusion matrix defined as in Equation 3.1, yields a good strategy for the imbalanced classes problem.

### 3.2.2 Towards a boosting method

Let $\mathcal{H}$ be an ensemble of classifiers and $S = \{\mathbf{x}_i, y_i\}_{i=1}^m$ a learning sample. Our goal is to find a classifier $\hat{H}$ which can be written as a weighted combination of all the classifiers of $\mathcal{H}$ and at the same time, it minimizes the norm of the confusion matrix. That is:

$$\hat{H} = \underset{H \in \mathcal{H}}{\operatorname{argmin}} \|\mathbf{C}_S\|^2, \tag{3.8}$$

where:

$$H(\cdot) = \operatorname{argmax}_{l \in \{1, \cdots, K\}} \sum_{h \in \mathcal{H}} \alpha_h \mathbb{I}(h(\cdot) = l) \text{ and } \alpha_h \geq 0, \forall h. \tag{3.9}$$

In order to make use of the bound given in Equation 3.7, we need to choose loss functions adapted to our problem. We start off by defining score functions for each example $i$ and each class $l$ as follows:

$$f_H(i,l) = \sum_{h\in\mathcal{H}} \alpha_h \mathbb{I}(h(\mathbf{x}_i) = l), \forall i \in \{1,\cdots,m\}, l \in \{1,\cdots,K\}.$$

These simple functions measure the importance that the classifier $H$ gives to each class for a given example. Since the prediction rule is based on the argmax (Eq. (3.8)), $H$ returns the class with the highest score. If $H$ correctly classifies an example, than the score given to its true class is higher than the score given to any of the other classes. Hence a good idea for the loss functions would be to consider the difference between the score given to a class and the one given to the true class:

$$\ell_{y_i,j}(H,\mathbf{x}_i) = f_H(i,j) - f_H(i,y_i), \forall j \neq y_i.$$

If an example is correctly classified then the loss is negative for all the classes $j$, while if it is misclassified, then for at least one of the classes, the loss is positive. This observation reveals to be the main downside of these losses, since it implies that the requirements for the bound in Equation 3.7 might not be met. That is, it is possible that for some $j \neq y$, $\ell_{l,j}(h,\mathbf{x}) \leq \mathbb{I}(h(\mathbf{x}) \neq y)$. This is why we propose to consider the exponential losses instead, and to rewrite our goal Eq. 3.8 as:

$$\hat{H} = \underset{H\in\mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{m}\sum_{j\neq y_i} \frac{1}{m_{y_i}} \exp\Big(f_H(i,j) - f_H(i,y_i)\Big).$$

Finding the optimal solution for the last equation can be quite challenging, since it depends on the size of $\mathcal{H}$ and the weights associated to each classifier in $\mathcal{H}$. A popular approach when dealing with ensembles of classifiers is to use an iterative greedy method based on the boosting framework. The goal of such methods is to select at each iteration the best classifier $h \in \mathcal{H}$ — and its weight $\alpha$ — that minimizes the loss. The main advantage of boosting methods resides in the fact that the selected classifier $h$, and generally all the classifiers in $\mathcal{H}$, need only to perform slightly better than random guessing. They are also known as weak classifiers.

Suppose that at iteration $t$, the classifiers $h_1,\cdots,h_t \in \mathcal{H}$ have been chosen in order to minimize the loss of $H_t(\cdot) = \operatorname{argmax}_{l\in\{1,\cdots,K\}} \sum_{s=1\cdots t} \alpha_s \mathbb{I}(h_s(\cdot) = l)$. In the following iteration, the chosen classifier would be the one verifying:

$$h_{t+1}, \alpha_{t+1} = \underset{h\in\mathcal{H},\alpha}{\operatorname{argmin}} \sum_{i=1}^{m}\sum_{j\neq y_i} \frac{1}{m_{y_i}} \exp\Big(f_{t+1}(i,j) - f_{t+1}(i,y_i)\Big), \tag{3.10}$$

where $f_{t+1}(i,j) = \sum_{s=1\cdots t} \alpha_s \mathbb{I}(h_s(\mathbf{x}_i) = j) + \alpha \mathbb{I}(h(\mathbf{x}_i) = j)$

This last optimization problem is quite similar to the one defined for AdaBoost.MM Mukherjee and Schapire (2011). As such, the method proposed in this paper is an extension of AdaBoost.MM to the imbalanced classes setting.

In order to use the boosting framework given in Mukherjee and Schapire (2011), we need to define a cost matrix $\mathbf{D}$ so that $\forall i$, and $\forall l \neq y_i, \mathbf{D}(i,l) \leq \mathbf{D}(i,y_i)$. Due to the similarity of our minimization problem 3.10 and AdaBoost.MM's, the most straightforward choice for $\mathbf{D}$ is the following :

$$\mathbf{D}_t(i,l) \overset{def}{=} \begin{cases} \frac{1}{m_{y_i}} \exp(f_t(i,l) - f_t(i,y_i)) & \text{if } l \neq y_i \\ -\sum_{j\neq y_i} \frac{1}{m_{y_i}} \exp(f_t(i,j) - f_t(i,y_i)) & \text{otherwise.} \end{cases} \tag{3.11}$$

### 3.2.3 The Confusion Matrix Boosting Algorithm

The pseudo-code of the proposed method named CoMBo is given in Algorithm 4. The inputs : a training sample $S$, the total number of iterations $T$ and a weak learner $\mathcal{W}$. During the initialization step, the score functions $f$ are set to zero and the cost matrix $\mathbf{D}$ is initialized accordingly.

The training phase consists of two steps: the weak learner $\mathcal{W}$ is used in order to build the weak classifiers, and the predictions of $h_t$ are used to update the cost matrix $\mathbf{D}_t$. At each round $t$, $\mathcal{W}$ takes as input the cost matrix $\mathbf{D}_t$ and returns a weak classifier $h_t$. The cost matrix is then used to compute the weight $\alpha_t$ for $h_t$, which can be seen as the importance given to $h_t$. $\alpha_t$ depends on the edge $\delta_t$ obtained by $h_t$ over the cost matrix $\mathbf{D}_t$. For boosting methods based on AdaBoost, the edge measures the difference of performances of the classifier and of random guessing (Mukherjee and Schapire 2011). The underlying idea is that the better $h_t$ performs over $\mathbf{D}_t$, the greater the edge $\delta_t$ and $\alpha_t$.

The update rule for the cost matrix is designed so that the misclassification cost is increased for the examples that are misclassified by $h_t$, while it is decreased for the correctly classified ones. This forces the weak learner $\mathcal{W}$ to focus on the most difficult examples. Then, using the term $1/m_{y_i}$ in the update rule, allows the misclassification cost of an example to depend not only on the ability to correctly classify a hard example, as in AdaBoost.MM, but also on the number of examples of $S$ having the same class $y_i$.

The output hypothesis is a simple weighted majority vote over the whole set of weak classifiers. So, for a given example, the final prediction is the class that obtains the highest score.

### 3.2.4 Bounding the loss

First off we recall the minimal weak learning condition as given in Mukherjee and Schapire (2011).

**Definition 8** *Let $D^{eor}$ be the space of all cost matrices $\mathbf{D}$ which put the least cost on the correct label, that is $\forall(\mathbf{x}_i, y_i), l, \mathbf{D}(i, y_i) \leq \mathbf{D}(i, l)$. Let $B^{eor}_{\gamma}$ be the space of baselines $\mathbf{B}$ which are $\gamma$ more likely to predict the correct label for every example $(\mathbf{x}_i, y_i)$, i.e. $\forall l \neq y_i, \mathbf{B}(i, y_i) \geq \mathbf{B}(i, l) + \gamma$. Then, the minimal weak learning condition is given by :*

$$\forall \mathbf{D} \in D^{eor}, \exists h \in \mathcal{H} : \mathbf{D} \cdot \mathbf{1}_h \leq \max_{\mathbf{B} \in B^{eor}} \mathbf{D} \cdot \mathbf{B}, \tag{3.12}$$

*where $\mathcal{H}$ is a classifier space, and $\mathbf{1}_h$ is the matrix defined as $\mathbf{1}_h(i, l) = \mathbb{I}[h(i) = l]$.*

Much as in the case of MuMBo given in Chapter 2, we consider a particular case of baselines, which are the closest to the uniform. These baselines, noted $\mathbf{U}_{\gamma}$, have weights $(1 - \gamma)/k$ on incorrect labels and $(1 - \gamma)/k + \gamma$ on the correct ones. The weak learning condition is given by :

$$\mathbf{D} \cdot \mathbf{1}_h \leq \mathbf{D} \cdot \mathbf{U}_{\gamma} \tag{3.13}$$

All of the weak classifiers returned by $WL$ during the training phase verify this weak learner condition.

The following result shows that the general loss decreases with each iteration, provided that the weak classifier $h_t$ satisfies the weak learning condition Eq. 3.13. This result and its proof are fairly similar to the ones given for AdaBoost.MM.

**Algorithm 4:** CoMBo : Confusion Matrix BOosting

**Given**

— S = $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$ where $\mathbf{x}_i \in X$, $y_i \in \{1, \cdots, K\}$

— T the number of iterations, WL a weak learner

— $\forall i \in \{1, \cdots, m\}$, $\forall l \in \{1, \cdots, K\}$ $f_1(i, l) = 0$

— $\mathbf{D}_1(i, l) = \begin{cases} \frac{1}{m_{y_i}} & \text{if } y_i \neq l \\ \frac{-(K-1)}{m_{y_i}} & \text{if } y_i = l \end{cases}$

**for** $t = 1$ **to** $T$ **do**

Get $h_t$ with edge $\delta_t$ on $\mathbf{D}_t$, and $\alpha_t = \frac{1}{2} \ln \frac{1+\delta_t}{1-\delta_t}$

where :

$$\delta_t = \frac{-\sum_{i=1}^{m} \mathbf{D}_t(i, h_t(\mathbf{x}_i))}{\sum_{i=1}^{m} \sum_{l \neq y_i} \mathbf{D}_t(i, l)}$$

Update $\mathbf{D}$ :

$$\mathbf{D}_{t+1}(i, l) = \begin{cases} \frac{1}{m_{y_i}} \exp(f_{t+1}(i, l) - f_{t+1}(i, y_i)) & \text{if } l \neq y_i \\ -\frac{1}{m_{y_i}} \sum_{j \neq y_i}^{k} \exp(f_{t+1}(i, j) - f_{t+1}(i, y_i)) & \text{if } l = y_i \end{cases}$$

where $f_{t+1}(i, l) = \sum_{z=1}^{t} \mathbb{I}[h_z(i) = l]\alpha_z$

**end for**

Output final hypothesis :

$$H(\mathbf{x}) = \underset{l \in 1, ..., k}{\operatorname{argmax}} f_T(\mathbf{x}, l), \text{ where } f_T(\mathbf{x}, l) = \sum_{t=1}^{T} \mathbb{I}[h_t(\mathbf{x}) = l]\alpha_t$$

**Lemma 2** *Suppose the cost matrix $\mathbf{D}_t$ is chosen as in the Algorithm 4, and the returned classifier $h_t$ satisfies the edge condition for the baseline $\mathbf{U}_{\delta_t}$ and cost matrix $\mathbf{D}_t$, i.e. $\mathbf{D}_t \cdot \mathbf{1}_{h_t} \leq \mathbf{D}_t \cdot \mathbf{U}_{\delta_t}$.*

*Then choosing a weight $\alpha_t > 0$ for $h_t$ makes the loss at round $t$ at most a factor*

$$1 - \frac{1}{2}(e^{\alpha_t} - e^{-\alpha_t})\delta_t + \frac{1}{2}(e^{\alpha_t} + e^{-\alpha_t} - 2)$$

*of the loss before choosing $\alpha_t$, where $\delta_t$ is the edge of $h_t$.*

*Proof.* We give here a sketch of the proof, since most of the computation are similar to the ones given in Theorem 1.

For readability reasons, let us note by $L_t$, and $L_t(i)$ the following losses:

$$L_t = \sum_{i=1}^{m} \sum_{l \neq y_i} \mathbf{D}_t(i, l) = \sum_{i=1}^{m} \sum_{l \neq y_i} \frac{1}{m_{y_i}} \exp(f_t(i, l) - f_t(i, y_i)) \text{ and}$$

$$L_t(i) = \sum_{l \neq y_i} \mathbf{D}_t(i, l) = \sum_{l \neq y_i} \frac{1}{m_{y_i}} \exp(f_t(i, l) - f_t(i, y_i)).$$

The weak classifier $h_t$ returned by $WL$ satisfies the edge condition, that is:

$$\mathbf{D_t} \cdot \mathbf{1}_{h_t} \leq \mathbf{D_t} \cdot \mathbf{U}_{\delta_t}, \tag{3.14}$$

with $\delta_t$ being the edge of $h_t$ on $\mathbf{D_t}$.

Denote $S_+$ (resp. $S_-$) the set of examples of $S$ correctly classified (resp. misclassified) by $h_t$. Using the different definitions of $\mathbf{D_t}$, $\mathbf{1}_{h_t}$ and $\mathbf{U}_{\delta_t}$, the classification costs of $h_t$ and $\mathbf{U}_{\delta_t}$ are given by:

$$\begin{aligned}\mathbf{D}_t \cdot \mathbf{1}_{h_t} &= -\sum_{i \in S_+} L_{t-1}(i) + \sum_{i \in S_-} \frac{1}{m_{y_i}} \exp\left(f_{t-1}(i, h_t(i)) - f_{t-1}(i, y_i)\right) \\ &= -A_+^t + A_-^t \\ \mathbf{D}_t \cdot \mathbf{U}_{\delta_t} &= -\delta_t \sum_{i=1}^{m} L_{t-1}(i) = -\delta_t L_{t-1}\end{aligned}$$

Injecting these two costs in Eq. 3.14, we have :

$$A_+^t - A_-^t \geq \delta_t L_{t-1}. \tag{3.15}$$

Taking a closer look at the drop of the loss after choosing $h_t$ and $\alpha_t$, we have:

$$\begin{aligned}L_{t-1} - L_t &= \sum_{i \in S_+} L_{t-1}(i)(1 - e^{-\alpha_t}) + \sum_{i \in S_-} \frac{1}{m_{y_i}} \exp(\Delta f_{t-1}(i, h_t, y_i))(1 - e^{\alpha_t}) \\ &= (1 - e^{-\alpha_t}) A_+^t - (e^{\alpha_t} - 1) A_-^t \\ &= \left(\frac{e^{\alpha_t} - e^{-\alpha_t}}{2}\right)(A_+^t - A_-^t) - \left(\frac{e^{\alpha_t} + e^{-\alpha_t} - 2}{2}\right)(A_+^t + A_-^t)\end{aligned}$$

where $\Delta f_{t-1}(i, h_t, y_i)) = f_{t-1}(i, h_t(i)) - f_{t-1}(i, y_i)$.

The result in 3.15 gives a lower bound for $A_+^t - A_-^t$, while $A_+^t + A_-^t$ is upper-bounded by $L_{t-1}$. Hence,

$$L_{t-1} - L_t = \sum_{i \in S_+} L_{t-1}(i)(1 - e^{-\alpha_t}) \geq (\frac{e^{\alpha_t} - e^{-\alpha_t}}{2})\delta_t L_{t-1} - (\frac{e^{\alpha_t} + e^{-\alpha_t} - 2}{2}) L_{t-1}.$$

Therefore, the result of the lemma:

$$L_t \leq \left(1 - \tfrac{e^{\alpha_t} - e^{-\alpha_t}}{2}\delta_t + \tfrac{e^{\alpha_t} + e^{-\alpha_t} - 2}{2}\right) L_{t-1} = \left(\tfrac{(1-\delta_t)}{2} e^{\alpha_t} + \tfrac{(1+\delta_t}{2} e^{-\alpha_t})\right) L_{t-1}. \tag{3.16}$$

□

The expression of the loss drop given in the Lemma 2 can be further simplified. Indeed, if we choose the value of $\alpha_t$ as given in the pseudo-code of Algorithm 4, than the loss drop is simply equal to $\sqrt{1 - \delta_t^2}$. Note that for the choice of $\delta_t$, the arguments given in Section 2.3 are still valid, thus obtaining the value of $\delta_t$ in Algorithm 4. Since the value of $\delta_t^2$ is always positive, $\sqrt{1 - \delta_t^2}$ is smaller than 1, thus the loss $L_t$ is always smaller than $L_{t-1}$. The following theorem resumes this result.

**Theorem 4** *Let $\delta_1, \cdots, \delta_T$ be the edges of the classifiers $h_1, \cdots, h_T$ returned by WL at each round of the learning phase. Then the error after $T$ rounds is $K(K-1)\prod_{t=1}^{T}\sqrt{1-\delta_t^2} \leq K(K-1)\exp\left\{-(1/2)\sum_{t=1}^{T}\delta_t^2\right\}$.*

*Moreover, if there exists a $\gamma$ so that $\forall t, \delta_t \geq \gamma$, then the error after $T$ rounds is exponentially small, $K(K-1)e^{-T\gamma^2/2}$.*

## 3.3 Confusion matrices in the PAC-Bayesian framework

In Section 3.2, we used the result in Corollary 2 in order to justify the use of the empirical confusion matrix instead of the true one. However, other works have proven a link between the true confusion matrix and the empirical one. In this section, we briefly present results for the Gibbs classifier and the Bayes classifier in the PAC-Bayesian setting[3]. Let $\mathcal{H} \subseteq Y^X$ be an ensemble of classifiers, and $\mathcal{P}$ a distribution defined over $\mathcal{H}$. The Gibbs classifier $G_{\mathcal{P}}$ is a stochastic classifier which predicts the label of $x \in X$ by first drawing $h \in \mathcal{H}$ according to $\mathcal{P}$ and then returning $h(x)$. On the other hand, the Bayes classifier in the PAC-Bayesian setting, we'll denote it $B_{\mathcal{P}}$, is the majority vote classifier weighted by $\mathcal{P}$. The Bayes classifier is similar to the majority vote defined in Eq. 3.9, where for a given classifier $h \in \mathcal{H}$, $\alpha_h$ is the probability of choosing $h$ according to $\mathcal{P}$.

The goal of PAC-Bayesian methods is to find a posterior distribution $\mathcal{Q}$ leading to good generalization results, given a prior distribution $\mathcal{P}$ and a training set $S$. Let $\mathbf{C}^{G_{\mathcal{P}}}$ and $\mathbf{C}_S^{G_{\mathcal{P}}}$ denote the true and empirical confusion matrix of the Gibbs classifier $G_{\mathcal{P}}$. In Morvant et al. (2012), we show the following bound on the norms of the two matrices:

**Theorem 5** *Let $X$ be the input space, $Y = \{1, \ldots, K\}$ the output space, $\mathfrak{D}$ a distribution over $X \times Y$ (with $\mathfrak{D}_m$ the distribution of a $m$-sample) and $\mathcal{H}$ a family of classifiers from $X$ to $Y$. Then for every prior distribution $\mathcal{P}$ over $\mathcal{H}$ and any $\delta \in (0, 1]$, we have:*

$$\mathbb{P}_{S\sim\mathcal{D}_m}\left\{\forall \mathcal{Q} \text{ on } \mathcal{H},\ \|\mathbf{C}^{G_{\mathcal{Q}}}\| \leq \|\mathbf{C}_S^{G_{\mathcal{Q}}}\| + \sqrt{\frac{8K}{m_- - 8K}\left[KL(\mathcal{Q}||\mathcal{P}) + \ln\left(\frac{m_-}{4\delta}\right)\right]}\right\} \geq 1 - \delta.$$

*where $m_- = \min_{y=1,\ldots,K} m_y$ is the minimal number of examples from $S$ which belong to the same class, and $KL$ in the Kullback-Leibler divergence between distributions $\mathcal{Q}$ and $\mathcal{P}$ defined as:*

$$KL(\mathcal{Q}||\mathcal{P}) = \mathbb{E}_{h\sim\log\mathcal{Q}} \frac{\mathcal{Q}(h)}{\mathcal{P}(h)}.$$

*Proof.* Deferred in Section A.5, page 112. □

The following proposition gives the relation between the norm of the Gibbs classifier's confusion matrix and the Bayes classifier's one.

**Proposition 1** *Let $K \geq 2$ be the number of classes. Then $\mathbf{C}^{B_{\mathcal{Q}}}$ and $\mathbf{C}^{G_{\mathcal{Q}}}$ are related by the following inequality:*

$$\|\mathbf{C}^{B_{\mathcal{Q}}}\| \leq K\|\mathbf{C}^{G_{\mathcal{Q}}}\|. \tag{3.17}$$

The final classifier of CoMBo is a collection of several (weak) classifiers selected from a certain $\mathcal{H} \subset Y^X$. Normalizing the weights associated to the classifiers, gives a distribution on $\mathcal{H}$, similar to the posterior distribution $\mathcal{Q}$ in Theorem 5. As pointed out, the final classifier of CoMBo is similar to the Bayes classifier and, as a consequence, Equation 3.17 coupled with Theorem 5 gives a PAC-Bayesian bound on the norm of the true confusion matrix for the final hypothesis of CoMBo (albeit, CoMBo was not based on the PAC-Bayesian framework).

3. We refer the reader to the works by McAllester (1999a), Catoni (2007), Langford (2005) for more details on the PAC-Bayesian theory and framework.

| Dataset | # cls. | # ex. | distribution of classes | Imb. ratio |
|---|---|---|---|---|
| New-Thyroid | 3 | 215 | 150/35/30 | 5.00 |
| Balance | 3 | 625 | 49/288/288 | 5.88 |
| Car | 4 | 1728 | 1210/384/69/65 | 18.62 |
| Connect | 3 | 67557 | 44473/6449/16635 | 6.90 |
| Nursery-s | 4 | 12958 | 4320/328/4266/4044 | 13.17 |
| Glass | 6 | 214 | 70/76/17/13/9/29 | 8.44 |
| E.coli | 5 | 327 | 143/77/52/35/20 | 7.15 |
| Yeast | 10 | 1484 | 463/429/244/163/51/ 44/35/30/20/5 | 92.60 |
| Satimage | 6 | 6435 | 1533/703/1358/626/ 707/1508 | 2.45 |

Table 3.1 – *Class distributions of considered UCI datasets (the last column reports the ratio between the # of instances of the majority class and the # of instances of the minority class (imbalance ratio).)*

## 3.4 Experimental results

### 3.4.1 Datasets and experimental setup

In order to compare CoMBo with other approaches, it was run on 8 imbalanced multi-class classification datasets. They are all from the UCI Machine Learning Repository (Frank and Asuncion 2010). They exhibit various degrees of imbalance, as well as various numbers of instances and attributes. Table 3.1 summarizes information about these datasets.

For each dataset, we performed 10 runs of 5-fold cross-validation of CoMBo and Adaboost.MM with a low-size decision tree as a base learner ($\mathcal{W}$) and $T = 200$, and we averaged the results. Two kinds of results are reported: advanced accuracy performances are first compiled (MAUC, Gmean), then the behavior of CoMBo is compared to AdaBoost.MM with a deeper insight on the norm of the confusion matrix.

### 3.4.2 Performance results

Usual performance measures of multi-class classification include MAUC (an extension of the AUC to multi-class problems, (Hand and Till 2001)), and G-mean (the geometric mean of per-class recall values computed for all classes (Sun et al. 2006)).

Table 3.2 reports these measures for CoMBo, as well as results from two boosting-based algorithms relying on resampling: AdaBoost.NC with oversampling (Wang and Yao 2012), and SmoteBoost (Chawla et al. 2003). These algorithms were chosen for comparison because they showed the best results on the considered datasets, as given in (Wang and Yao 2012).

Looking at results in Table 3.2, CoMBo is quite promising w.r.t. current literature, without any tuning nor trying to optimize the reported measures. Only the best results of AdaBoost.NC and SmoteBoost from Wang and Yao (2012) are reported. Except for the dataset `Satimage` (the less imbalanced dataset), CoMBo challenges the other boosting-based approaches. The advantage of preferring CoMBo over the reported methods seems to be related to the imbalance ratio: the higher this ratio is, the more CoMBo makes the difference, which is consistent with the aim CoMBo. It is worth noticing that CoMBo does not need any prior initialization of the misclassification cost matrix: it computes and adjusts the cost matrix along the learning stages, based on the performances of the weak classifiers. In other words, CoMBo gets these good

| **G-mean** | CoMBo | AdaBoost.NC | SmoteBoost |
|---|---|---|---|
| Car | **0.967** $\pm$ 0.023 | 0.924 $\pm$ 0.024 | 0.944 $\pm$ 0.031 |
| Balance | **0.675** $\pm$ 0.088 | 0.321 $\pm$ 0.173 | 0.000 $\pm$ 0.000 |
| New-Thyroid | 0.914 $\pm$ 0.081 | 0.927 $\pm$ 0.056 | 0.940 $\pm$ 0.057 |
| Nursery-s | **1.000** $\pm$ 0.001 | 0.967 $\pm$ 0.006 | 0.996 $\pm$ 0.003 |
| Ecoli | 0.784 $\pm$ 0.066 | 0.790 $\pm$ 0.062 | 0.803 $\pm$ 0.059 |
| Glass | 0.431 $\pm$ 0.0.375 | 0.578 $\pm$ 0.249 | 0.561 $\pm$ 0.343 |
| Satimage | †0.825 $\pm$ 0.012 | 0.881 $\pm$ 0.009 | **0.898** $\pm$ 0.010 |
| Yeast | 0.107 $\pm$ 0.216 | 0.237 $\pm$ 0.270 | 0.140 $\pm$ 0.240 |
| **MAUC** | CoMBo | AdaBoost.NC | SmoteBoost |
| Car | 0.993 $\pm$ 0.003 | 0.982 $\pm$ 0.005 | **0.997** $\pm$ **0.000** |
| Balance | **0.884** $\pm$ 0.032 | 0.704 $\pm$ 0.037 | 0.703 $\pm$ 0.027 |
| New-Thyroid | **0.996** $\pm$ 0.010 | 0.983 $\pm$ 0.013 | 0.988 $\pm$ 0.003 |
| Nursery-s | **1.000** $\pm$ 0.000 | 0.998 $\pm$ 0.000 | 0.999 $\pm$ 0.000 |
| Ecoli | 0.961 $\pm$ 0.015 | 0.957 $\pm$ 0.002 | 0.963 $\pm$ 0.004 |
| Glass | **0.947** $\pm$ 0.027 | 0.881 $\pm$ 0.009 | 0.925 $\pm$ 0.009 |
| Satimage | †0.976 $\pm$ 0.003 | 0.990 $\pm$ 0.000 | **0.992** $\pm$ 0.000 |
| Yeast | 0.861 $\pm$ 0.025 | 0.857 $\pm$ 0.004 | 0.847 $\pm$ 0.003 |

Table 3.2 – *Comparison with algorithms addressing classification within imbalanced datasets. Means and standard deviations are given over 10 runs of 5-fold cross-validations. Except for CoMBo, the results are the* best *retrieved from Wang and Yao (2012), which analysed four algorithms. Results in boldface indicate a significantly best measure of the algorithm over all the others, according to the sudent T-test with a confidence of 95%; the † attached to the result on* `Satimage` *indicates that CoMBo is significantly worse than all other reported methods.*

results without any other parameters than the weak algorithm, and the number $T$ of rounds.

### 3.4.3 Improvements from Adaboost.MM

One may think that the good results of CoMBo are partly due to the fact that it is based on the theory underlying Adaboost.MM. Therefore, it is worth exploring the way CoMBo differs from Adaboost.MM in the processing of imbalanced datasets through the norm of the confusion matrix. Results are presented Table 3.3: the confusion matrix norms are reported, together with the accuracies, the MAUC and the G-mean.

These results confirm the good results of CoMBo on imbalanced datasets as evidenced by the low values of the norms. They illustrate the impact of minimizing the norm of the confusion matrix. Let us note that the accuracy with CoMBo tends to be a bit worse than the one of Adaboost.MM, although this was to be expected since CoMBo does not tend to maximize the overall accuracy, but rather is aims to achieve the same accuracy for each class. Meanwhile, as expected, the norm of the confusion matrix is always smaller with CoMBo. The performances of the classifier trained with CoMBo is smoothed throughout all the classes, whatever the number of examples they feature in the dataset. That way, majority classes are not as favored as they usually are in multi-class approaches.

On non-reported datasets, computed measures for both algorithms are very close. On reported results, we observe that the lower the confusion matrix norm, the higher the gain on MAUC and on G-mean. These results let us think that there might be no

| dataset | CoMBo $\|\mathbf{C}\|$ | AdaMM $\|\mathbf{C}\|$ | CoMBo accuracy | AdaMM accuracy |
|---|---|---|---|---|
| Balance | **0.460** $\pm$ 0.097 | 0.559 $\pm$ 0.112 | 0.856 $\pm$ 0.039 | 0.875 $\pm$ 0.032 |
| Car | 0.082 $\pm$ 0.064 | 0.116 $\pm$ 0.080 | 0.974 $\pm$ 0.009 | **0.977** $\pm$ 0.012 |
| Connect | **0.308** $\pm$ 0.015 | 0.670 $\pm$ 0.040 | 0.728 $\pm$ 0.015 | **0.805** $\pm$ 0.009 |
| New-Thyroid | 0.194 $\pm$ 0.158 | 0.186 $\pm$ 0.157 | 0.949 $\pm$ 0.033 | 0.952 $\pm$ 0.033 |
| Nursery-s | 0.002 $\pm$ 0.004 | 0.003 $\pm$ 0.006 | 1.000 $\pm$ 0.000 | 1.000 $\pm$ 0.001 |
| Yeast | **0.815** $\pm$ 0.150 | 1.101 $\pm$ 0.194 | 0.572 $\pm$ 0.025 | 0.577 $\pm$ 0.025 |
| | G-mean | G-mean | MAUC | MAUC |
| Balance | **0.675** $\pm$ 0.088 | 0.566 $\pm$ 0.190 | **0.884** $\pm$ 0.032 | 0.855 $\pm$ 0.039 |
| Car | 0.967 $\pm$ 0.023 | 0.954 $\pm$ 0.034 | **0.993** $\pm$ 0.003 | 0.988 $\pm$ 0.004 |
| Connect | **0.703** $\pm$ 0.013 | 0.497 $\pm$ 0.035 | **0.863** $\pm$ 0.008 | 0.852 $\pm$ 0.009 |
| New-Thyroid | 0.914 $\pm$ 0.081 | 0.915 $\pm$ 0.075 | 0.996 $\pm$ 0.010 | 0.996 $\pm$ 0.005 |
| Nursery-s | **1.000** $\pm$ 0.001 | 0.999 $\pm$ 0.002 | 1.000 $\pm$ 0.000 | 1.000 $\pm$ 0.000 |
| Yeast | 0.107 $\pm$ 0.216 | 0.000 $\pm$ 0.000 | **0.861** $\pm$ 0.025 | 0.847 $\pm$ 0.003 |

Table 3.3 – *Adaboost.MM vs. CoMBo. In non-reported datasets, results of both algorithms are equivalent). Means and standard deviations are given over 10 runs of 5-fold cross-validations. Results in boldface indicate a significantly best measure according to the sudent T-test with a confidence of 95%.*

gain using CoMBo instead of Adaboost.MM in other cases, but there is no loss either (the computational times are the same).

$$\underset{\text{CoMBo}}{\begin{pmatrix} 0.000 & 0.147 & 0.068 \\ 0.146 & 0.000 & 0.169 \\ 0.056 & 0.203 & 0.000 \end{pmatrix}} \qquad \underset{\text{Adaboost.MM}}{\begin{pmatrix} 0.000 & 0.011 & 0.055 \\ 0.656 & 0.000 & 0.179 \\ 0.246 & 0.048 & 0.000 \end{pmatrix}}$$

Table 3.4 – *Confusion matrices obtained with Adaboost.MM and CoMBo, on the dataset* `Connect`.

Finally, Table 3.4 illustrates what actually occurs on the dataset `Connect` [4]. Class 1 corresponds to the majority class, while class 2 and 3 are the minority ones. The errors on minority classes 2 (83.5%) and 3 (29.4%) are high with Adaboost.MM which promotes the majority class (only 6.6% of errors). These differences are reduced with CoMBo: the error on the majority class reaches 20.2% while errors on minority classes decrease respectively to 31.5% and 25.9%. However, the real error is still higher with CoMBo: misclassified examples of the majority classes getting more numerous, it directly impacts the overall error rate. Such a behavior of CoMBo points out that it equally considers each class during the learning process, independently from any tricky misclassification cost.

### 3.4.4 G-mean, MAUC and CoMBo

**G-mean** The results given in Table 3.3 show that the difference between the norm of the confusion matrices obtained for CoMBo and AdaBoost.MM is related to the differences of G-means: the bigger the difference of the norms, the better the performances of CoMBo w.r.t. the G-mean. This is mainly due to the aforementioned mentioned *smoothing* effect achieved from CoMBo on the confusion matrix. In Section

4. The dataset `Connect` was chosen for illustration because of readability (it only features 3 classes) and because it is much imbalanced.

3.1.2, when we introduced the confusion matrix and the operator norm, we argued that in the binary case, the norm of the confusion matrix is equal to the largest value between $FP_{rate}$ and $FN_{rate}$ and an iterative minimization process of the norm would achieve a similar value for $FP_{rate}$ and $FN_{rate}$. The generalization of the confusion matrix's norm to the multi-class case can be seen as a generalization of this fact: the value of the norm depends on the highest values contained in the matrix. Likewise, an iterative process minimizing the norm of the confusion matrix (which is the case of CoMBo) would achieve lower values for all the entries of the confusion matrix. In other words, CoMBo favors matrices that contain small values for all the entries, over matrices that contain high values for some entries and close-to-zero values for the others [5]. If we calculate the *recalls* for the classes — which correspond to the diagonal entries in the probabilistic confusion matrices —, then CoMBo achieves roughly the same recall for each class.

In order to illustrate this, let us reconsider the example given in Table 3.4. First, recall that the G-mean is defined as the geometrical mean of al the per-class recalls. In this case, CoMBo achieves the following recalls for the three classes: 0.785 (class 1), 0.685 (class 2) and 0.741 (class 3). On the other hand, AdaBoost.MM achieves these recalls: 0.934 (class 1), 0.166 (class 2) and 0.706 (class 3). As previously noted, AdaBoost.MM is better than CoMBo for class 1, but is way worse for class 2. However, contrary to AdaBoost.MM, CoMBo is more constant in its results over the classes, since each class has roughly the same recall. A direct consequence of this is that the G-mean for CoMBo is greater than AdaBoost.MM's, which is weighed down from the recall of class 2.

Although the link between the minimization of the confusion matrix norm and that of G-mean seem immediate from this example, a formal proof is still lacking.

**MAUC** In the binary setting, the Area Under (the ROC) Curve (AUC) is computed from the ranking of all the training example based of the scores they obtain for the positive class. The examples that obtain the highest scores are ranked at the top, while the lowest scores are ranked at the bottom. The highest value for AUC is obtained when all the positive examples are on top of the negative examples. The multi-class extension of AUC, MAUC, is based on the same principle. However, since there are multiple classes, MAUC is computed from as many ranking as there are classes: one ranking per class and the goal is to have on top the examples of the corresponding class. The highest value for MAUC is obtained when, for each class, the examples of the corresponding class are on the top.

In their paper, Mukherjee and Schapire (2011) show that among the most popular multi-class boosting algorithm, the one that has the best weak learning condition in their framework is AdaBoost.MR. Recall that AdaBoost.MR uses weak learners that return a score for each class and the prediction is based on the ranking of the classes w.r.t. the scores. The goal of AdaBoost.MR is to learn a (final) classifier that, for each example, ranks its true class in the top position. It implies that their framework encourages methods that give the highest score to the true class, while at the same time maximizing the difference between the true class and the others [6]. This also applies to CoMBo. In algorithm 4 (CoMBo), the final hypothesis is a weighted majority vote of all the weak classifiers learnt during the training phase. Another way to look at

5. Usually, the higher values are in the lines corresponding to the minority classes, while the close-to-zero ones correspond to the majority classes.

6. This last part is also related to the notion of margin in the multi-class setting and the fact that Theorem 2 holds for most multi-class boosting methods.

| setting | algorithm | error | V1 | V2 | V3 | V4 | V5 |
|---|---|---|---|---|---|---|---|
| test (gold) | AdaBoost | # | **40** | 43 | **107** | **104** | **82** |
| | | % | **26.3** | 28.3 | **70.4** | **68.4** | **53.9** |
| | CoMBo | # | 46 | **42** | 111 | 116 | 92 |
| | | % | 30.3 | **27.6** | 73.0 | 76.3 | 60.5 |
| test (ASR) | AdaBoost | # | **65** | **55** | **107** | **104** | **82** |
| | | % | **42.7** | **36.2** | **70.4** | **68.4** | **53.9** |
| | CoMBo | # | 70 | 62 | 111 | 116 | 92 |
| | | % | 46.1 | 40.8 | 73.0 | 76.3 | 60.5 |

Table 3.5 – *Classification errors for CoMBo and AdaBoost on the five views of DECODA. Results in boldface indicate when one algorithm is better than the other.*

this hypothesis is that for a given example, a score is computed for each class and the predicted class is the one with the highest score: if we rank the classes based on their scores, the prediction is the class on the top. CoMBo achieves perfect accuracy if, for all the examples, their true class in on the top of the ranking, that is, for a given class, ranking the examples based on the scores for that class, would put the examples of that class in the top positions. Since this is also true for the other classes, it implies that in this case the MAUC attains its highest value. Therefore, at each CoMBo (and AdaBoost.MM) chooses a classifier that allows the weighted combination (of all the learnt classifiers) to be a step closer to the perfect ranking.

Even though this is by no means a formal proof of the link between CoMBo (and AdaBoost.MM) and the minimization of the MAUC, we think that it provides a fair explanation for the results observed in Table 3.2, where CoMBo achieves better MAUC than the other methods in most cases. As for the results in Table 3.3, the fact that CoMBo performs better than AdaBoost.MM (MAUC-wise) is linked to the *smoothing* effect obtained from the minimization of the confusion matrix's norm (as discussed for the G-mean).

## 3.5 An imbalanced approach to DECODA

In this section, we tackle the imbalanced nature of the DECODA corpus, described in Section 1.5 (page 22). As shown in Table 1.1, DECODA is obviously an imbalanced dataset; some classes are more represented than the others (*etfc, itnr* versus *aapl, vgc*). As such, we use it to test CoMBo and compare it to AdaBoost. Since both methods are mono-view ones, we test both algorithms on each one of the views separately. Table 3.5 resumes the results obtained for both the *gold* and *ASR* settings.

Firstly, the results given in Table 3.5 confirm that the views defined for DECODA are of different strengths. As expected the two content based views (the first and the second view) are clearly stronger than the other ones, and both AdaBoost and CoMBo perform better on these views. Secondly, the results for the third, forth and fifth views are the same for the *gold* setting and for the *ASR* one. Looking back at their descriptions in Section 1.5.2, this was to be expected, since neither of them is really affected from the ASR system. Indeed, the speakers' turns (third view) are directly computed from the audio signal, same as the parameters computed for the forth view; the named entities of the fifth view are generally given to the ASR. Finally, comparing the results of AdaBoost and CoMBo shows that AdaBoost outperforms CoMBo on nearly all the cases. This also was to be expected, since CoMBo tends to have a smaller accuracy compared to methods such as AdaBoost that directly minimize the risk.

| Setting | CoMBo MAUC | AdaB MAUC | CoMBo G-mean | AdaB G-mean | CoMBo $\|\mathbf{C}\|$ | AdaB $\|\mathbf{C}\|$ |
|---|---|---|---|---|---|---|
| view 1 (gold) | 0.901 | **0.912** | **0.618** | 0.0 | **0.501** | 0.730 |
| view 2 (gold) | **0.898** | 0.884 | **0.602** | 0.448 | 0.630 | **0.599** |
| view 3 (gold) | **0.548** | 0.529 | 0.0 | 0.0 | **1.357** | 1.556 |
| view 4 (gold) | 0.539 | **0.559** | 0.0 | 0.0 | **0.896** | 1.359 |
| view 5 (gold) | **0.789** | 0.784 | **0.355** | 0.0 | **0.832** | 1.141 |
| view 1 (ASR) | 0.839 | **0.855** | **0.476** | 0.0 | **0.684** | 0.780 |
| view 2 (ASR) | **0.828** | 0.804 | **0.507** | 0.217 | **0.621** | 0.829 |
| view 3 (ASR) | **0.548** | 0.529 | 0.0 | 0.0 | **1.357** | 1.556 |
| view 4 (ASR) | 0.539 | **0.559** | 0.0 | 0.0 | **0.896** | 1.359 |
| view 5 (ASR) | **0.789** | 0.784 | **0.355** | 0.0 | **0.832** | 1.141 |

Table 3.6 – *The MAUC, G-mean and confusion matrix's norms for AdaBoost and CoMBo on DECODA. Results in boldface indicate when one algorithm is better than the other.*

In order to observe the impact of CoMBo, let us have a look at Tables 3.6 and 3.7, which give the MAUC, G-mean and confusion matrix's norm, and the error rates for each of the classes and for the classifiers learnt on the first view of DECODA. Table 3.6 shows that the results for the MAUC for both methods are surprisingly close, even though CoMBo has a lower accuracy than AdaBoost. On the other hand, CoMBo outperforms AdaBoost on G-mean *and* on the norm of the confusion matrix, suggesting that CoMBo recognizes better the minority classes than AdaBoost. This is further confirmed in the results given in Table 3.7. Indeed, while AdaBoost performs better for majority classes, such as *timetable/traffic info*, *itinerary*, CoMBo is more adapted for minority classes, as *phone call*, *horr* (view 1) and *vgc* (view 2).

## 3.6 Conclusion for this chapter

### 3.6.1 Discussion

The framework proposed in this chapter raises several questions and perspectives, some of which will be discussed in this section.

Some potential extensions consist in using the result obtained in Equation 3.7 in order to derive other cost sensitive algorithms. As briefly discussed in Section 3.2, the optimization term depends on the different loss functions defined over the classes. The only restriction on these losses is that their value should be greater than the one returned by the indicator function. This implies that it is possible to embed various informations into the loss functions, as long as they respect this condition. For instance, it is easy to include in the proposed method penalization terms defined over two classes simply by replacing the loss functions considered in Section 3.2 with $\ell_{y_i,j}(h, \mathbf{x}_i) = c_{y_i,j} \exp(f_H(i,j) - f_H(i,y_i))$, where $c_{y_i,j}$ is a penalization term given as a prior or coming from a cost matrix.

It is interesting to notice that the result in Equation 3.7 regroups optimization criterions used in most of the boosting-based methods for the imbalanced classes, such as the ones by Ting (2000) and Sun et al. (2007). It also shows that these methods implicitly minimize the norm of the confusion matrix — which, as advocated in this chapter, is a fair optimization criterion for the imbalanced classes problem —, thus giving a better (theoretical) insight on why these methods are adapted for the imbalanced problem.

| | setting | test (gold) | | | | test (ASR) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | algorithm | AdaBoost | | CoMBo | | AdaBoost | | CoMBo | |
| | error | # | % | # | % | # | % | # | % |
| view 1 | aapl | 12 | 80.0 | **5** | **33.3** | 14 | 93.3 | **11** | **73.3** |
| | etfc | **3** | **7.3** | 9 | 21.9 | **13** | **31.7** | 15 | 36.6 |
| | horr | 4 | 80.0 | **3** | **60.0** | 4 | 80.0 | **3** | **60.0** |
| | itnr | **0** | **0.0** | 2 | 9.5 | **6** | **28.6** | 7 | 33.3 |
| | nvgo | 7 | 31.8 | **6** | **27.3** | 12 | 54.5 | **10** | **45.5** |
| | objt | **3** | **25.0** | 5 | 41.7 | 4 | 33.3 | 4 | 33.3 |
| | pv | **0** | **0.0** | 1 | 10.0 | 3 | 30.0 | 3 | 30.0 |
| | vgc | 5 | 45.5 | 5 | 45.5 | **5** | **45.5** | 8 | 72.7 |
| | other | **6** | **40.0** | 10 | 66.7 | **4** | **26.7** | 9 | 60 |
| | total | **40** | **26.3** | 46 | 30.3 | **65** | **42.7** | 70 | 46.1 |
| view 2 | aapl | 10 | 66.7 | **8** | **53.3** | 12 | 80.0 | **10** | **66.7** |
| | etfc | **3** | **7.3** | 5 | 12.2 | **3** | **7.3** | 12 | 29.3 |
| | horr | 4 | 80.0 | **3** | **60.0** | 3 | 60.0 | 3 | 60.0 |
| | itnr | **1** | **4.8** | 2 | 9.5 | **4** | **19.0** | 6 | 28.6 |
| | nvgo | **6** | **27.3** | 7 | 31.8 | **7** | **31.8** | 9 | 40.9 |
| | objt | 3 | 25.0 | **2** | **16.7** | 4 | 33.3 | **3** | **25.0** |
| | pv | 3 | 30.0 | 3 | 30.0 | **4** | **40.0** | 6 | 60.0 |
| | vgc | 6 | 54.5 | 6 | 54.5 | 9 | 81.8 | **7** | **63.6** |
| | other | 7 | 46.7 | 6 | 40.0 | 7 | 46.7 | **6** | **40** |
| | total | 43 | 28.3 | **42** | **27.6** | **55** | **36.2** | 62 | 40.8 |

Table 3.7 – *Per class classification errors for CoMBo and AdaBoost on the first view of DECODA. Results in boldface indicate when one algorithm is better than the other.*

Motivated by the theoretical and experimental results — think of the difference between the estimated error of the proposed method and AdaBoost.MM's — other works will be focussed on finding tighter bounds for the operator norm of the confusion matrix. Depending on the singular values computed from the confusion matrix, the bound given in Equation 3.7 can be quite loose. Finding a tighter bound would then be a first step towards a more efficient method in minimizing both the risk and the norm of the confusion matrix alike.

Continuing on the same line of thought, an important question that arises from the proposed framework is the following: is it possible to obtain similar results for other norms? Even though minimizing the various norms of the confusion matrix may be the same thing due to the equivalence between the norms, the main difference consists in whether it is possible to find an analytical expression of the norm. Some of the considered norms can be the $l1-$norm, entry-wise norms or even more exotic ones.

A second question that naturally comes to mind is if it is possible to consider other definitions for the confusion matrix, such as loss-based matrices, where the entries are replaced by loss functions. And more generally, it would be interesting to see if there exists a relationship between the choice of the confusion matrix and choice of the norm to be minimized.

Finally, based on the empirical results, we think that the norm of the confusion matrix is quite an useful tool for measuring the performance of a model in the multi-class and/or imbalanced classes setting, alongside classical measures such as MAUC, G-mean, F-measure, etc.

### 3.6.2 Conclusion

The contributions of this chapter are two-fold: firstly, we make use of the norm of the confusion matrix as an error measure in order to derive a unifying bound for most cost-sensitive approaches; secondly, using the bound as a starting point, we show step by step how to obtain an extension of AdaBoost.MM for the imbalanced classes framework. Due to the nature of the considered confusion matrix, we used an upper bound of the matrix's operator norm, namely the entry-wise $l1$-norm (Equation 3.6); as an intermediate step the (squared) Frobenious norm was used (Equation 3.5). Empirical results show that the proposed method compares favorably to other cost-based boosting methods. These performances are both due to the inherited boosting framework — which is more adapted to the multi-class problems than other frameworks — and to the weighting scheme obtained from the choice of adapted loss functions in Section 3.2. Although much work remains to be done, as discussed in Section 3.6.1, the proposed method is, to the best of our knowledge, the first boosting based approach that aims to actively minimize the norm of the confusion matrix, by including it in the learning process.

# 4 Extending the confusion matrix's norm analysis to the multi-view framework



When introducing the notion of *views* in Section 1.3, we pointed out three different types of views: *strong*, *weak* and *imbalanced* views. While the first two views are related to the information in the input space, the third type of view is mainly related to the information in the output space, and it englobes both the notion of weak view (views that recognize only a few classes) and that of imbalanced classes (in a view, some classes are more represented than the others). Chapter 2 introduced MuMBo, a multi-view boosting method, designed to deal with the first two type of views by installing and promoting the cooperation between the views. Chapter 3 introduced

a novel framework for the imbalanced classes problem, which rely on the operator norm of the confusion matrix of a given classifier.

In this chapter we tackle the imbalanced views problem by adding the notion of cooperation between the views in the imbalanced framework of Chapter 3, thus aiming to get the best of both worlds. We start by defining a multi-view classifier in Section 4.2.1 which depends on per-class cooperation coefficients computed for all the views. Next we make use of the framework of Chapter 3, that is, we use the norm of the confusion matrix as an optimization criterion. This allows us to derive several ensemble learning methods, mainly presented in Section 4.2.3, which rely upon the aforementioned cooperation coefficients. In Section 4.3, we present a multi-view version of CoMBo, dubbed $\mu$CoMBo, which deals with imbalanced views, and its empirical performances on the Arrhythmia dataset are given in Section 4.4. The performances of $\mu$CoMBo on the DECODA corpus are briefly described in Section 4.5.

Section 4.7 delves into the links between MuMBo (algorithm 3) and $\mu$CoMBo, leading the way to the discussions, conclusions and future works proposed in Section 4.8.

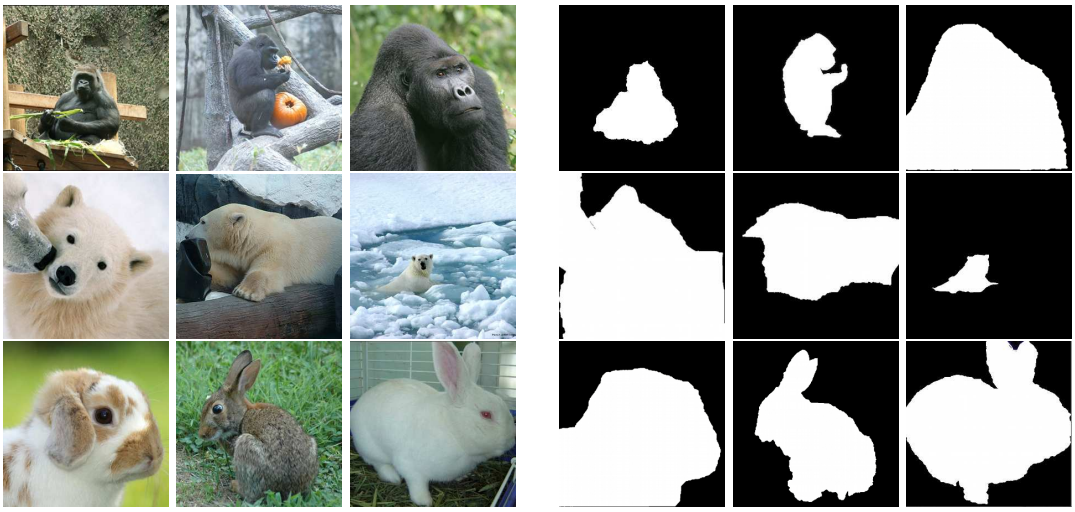
## 4.1 Embedding view cooperation into the confusion matrix

The goal of multi-view methods is to exploit the information contained in different views in order to improve the performances of the resulting classifier. In chapter 2, we present an algorithm — namely MuMBo, Algorithm 3 — whose goal is to point out the best examples for each view, that is, examples that are easier to recognize for classifiers trained on that view. This selection was mainly based on the assumption that, due to the different distributions on each view, some of the examples may be outliers — data that are distant form the rest of the data — in one view but not in the others. As such MuMBo uses only the information given by the views that are related to the input space, without taking into consideration the classes of the examples. However, in some applications, such as image and/or text categorization, it would be interesting to take into account information relative to the output space. In this chapter, we are mainly interested in two types of output space information:

— the first type regroups all the cases where a view is most appropriate for a class due to the embedded information,
— the second type delves into cases where each view represents an imbalanced classes problem.

In order to illustrate the first type of information, let us consider the following example (figure 4.1): the goal is to classify the images in *gorilla*, *polar bear* and *rabbit* and the considered views are the color histogram (figure 4.1a) and the shapes contained in the images (figure 4.1b). Both of the views are by no means weak ones (see section 2.2 for the definition of weak view), however each is more appropriated for training classifiers specialized in one class. Intuitively, the information contained in the first view should be more appropriate for recognizing the *gorilla* class, since polar bears and rabbits can be white, while gorillas are mainly black. It is safe to assume that the easier to recognize in the second view are the examples of class *rabbit* (think of the ears and the lack of *donkey*).

As an example for the second type, think of the cases where due to the lack of information, some examples contain only the descriptions of a subset of views. For example, in automatic medical diagnosis, the results of the different medical trials are excellent candidates for views. However, sometimes due to the number of trials, for



(a) *The original images of gorilla, polar bear and rabbit (the first view)*

(b) *The animals are separated from the background (the second view)*

Figure 4.1 – *Examples of images form the classes gorilla, polar bear and rabbit taken from the Animal with Attributes dataset Lampert et al. (2009).*

some patients only a limited number of views are available. This implies that each view may be optimal only for the detection of a small number of diseases.

In both examples, the information contained in one view may not be sufficient for training a classifier that performs equally on the ensemble of classes. However they both suggest that promoting some sort of cooperation between the (classifiers learnt on the different) views may be a good strategy in order to overcome these shortcomings. More precisely, for the first type of information, the idea is to install some sort of cooperation between the views — *à la MuMBo* — so that each view can deal with the most appropriate classes according to the informations it contains. The second type of information boils down to the imbalanced classes problem, which was mainly studied in chapter 3. Taking the best of both worlds in order to build classifiers that perform better over the whole set of classes (output space) is the goal of the methods presented in this chapter.

The intuition behind the proposed methods is to transpose the cooperation between the different views from the input space (MuMBo) to the output space, while at the same time dealing with the imbalanced classes problem (CoMBo). The first method, called Carako, is an ensemble learning method similar to the late fusion approach, where one classifier is learnt for each view and the goal is to find the optimal combination of the classifiers. On the other hand, the second method, called $\mu$CoMBo, uses a boosting process similar to MuMBo, where one distribution is maintained per view during the learning phase, and the final classifier is a combination of all the weak classifiers. Both methods share a common base with CoMBo, that is, tackling the imbalanced classes problem through the minimization of the operator norm of the confusion matrix of the final classifier. This classifier is a generalization of the final hypothesis given in algorithm 3 and algorithm 4, where different weights depending on the predicted class is affected to the classifiers, instead of an unique weight, independently from the prediction.

## 4.2 Ensemble methods for cooperation-embedded confusion matrix's norm minimization

The goal of the methods presented in this section is to minimize the norm of the confusion matrix where the classifier is learnt in a multi-view setting. The first step consists in defining a multi-view classifier which takes into consideration the classifiers learnt on each view. Next, proceeding in a similar fashion as in chapter 3, we derive a bound on the norm of the confusion matrix. This bound is used as a base for the first method. Next we replace the classifiers learnt on each view by boosted classifiers, that is, classifiers learnt through a boosting process. This allows to derive a boosting method, where at each iteration one weak classifier is learnt per view. In the last part of this section, we prove the boosting properties of the second method.

### 4.2.1 A multi-view classifier and its optimization problem

The main motivation behind the methods presented in this chapter is to transpose the cooperation between the views from the input space to the output space. In MuMBo, the cooperation is promoted through the examples during the training phase. On the contrary, within the output space, in this case the cooperation should be carried by the classifiers/predictors. Intuitively, the most straightforward solution is to enforce the cooperation during the building process of the classifier, so that it can embed information from all the views. A viable solution is then to consider a weighted majority classifier build from predictors learnt on the views. For example:

$$H(x) = \text{argmax}_{c \in 1 \cdots K} \sum_{j=1}^{v} \beta_j \mathbb{I}[h_j(x) = c],$$

where $H$ is the final classifier, $v$ the number of views and $h_j$, $\beta_j \geq 0$ are the classifier learnt on view $j$ and its weight.

Although this majority classifier takes into consideration the predictions of all the views, it associates a unique weight to each classifier, independently from the prediction and/or the ability of that view to recognize the right class. This is why we propose to choose a more general from for the weighted majority, where a weight is associated to each classifier and each class. More precisely:

$$H(x) = \text{argmax}_{c \in 1 \ldots K} \sum_{j=1}^{v} \beta_{j,c} \mathbb{I}[h_j(x) = c]. \tag{4.1}$$

Compared to the previous classifier, the weights affected to each classifier $j$ depend on the label $c$. As such, the number of weighting coefficient is $vK$ instead of simply $v$. Note that in order to obtain the previous classifier it suffices to replace, for a given $j$ and for all $c$, $\beta_{j,c}$ by $\beta_j$.

This new definition implements quite well the idea behind the cooperation between the different views on a per-class basis. Indeed, the weight associated to a classifier for a given class can be seen as the confidence that the classifier gets right for the examples of that class. The higher the weight, the easier it should be for the classifier to correctly recognize the examples of that class. If the goal is to find the most appropriate view for each class, then for each $c$, only one of the $\beta_{j,c}$ needs to be positive, let's say equal to 1. That is, the sole condition we put on the classifier is the

following:

$$\forall c \in \{1,\dots,K\}, \sum_{j=1}^{v} \beta_{j,c} = 1 \text{ and } \forall j \in \{1,\dots,v\}, \beta_{j,c} \in \{0,1\}. \tag{4.2}$$

In Chapter 3, we showed that using the norm of the confusion matrix as an optimization criterion leads to methods for the imbalanced classes setting. Since our goal is to deal with imbalanced views (see Section 1.3.1 for the definition of imbalanced views), we propose to use a similar procedure as in Chapter 3, in order to obtain a multi-view method for the imbalanced setting. Recall that in Section 3.1.2 the true and the empirical confusion matrices were defined as follows:

**Definition 9** *For a given classifier $h$ and a training set $S = \{(x_i, y_i)\}_{i=1}^{m}$, we define the true and empirical confusion matrices of $h$ by respectively $\mathbf{C} = (\mathbf{c}_{l,k})_{1\leq l,k\leq K}$ and $\mathbf{C}_S = (\hat{\mathbf{c}}_{l,k})_{1\leq l,k\leq K}$ such that for all $(l,k)$:*

$$\mathbf{c}_{l,k} \overset{def}{=} \begin{cases} 0 & \text{if } l = k \\ \mathbb{P}_{(x,y)\sim\mathfrak{D}}(h(x) = k | y = l) & \text{otherwise.} \end{cases}$$

$$\hat{\mathbf{c}}_{l,k} \overset{def}{=} \begin{cases} 0 & \text{if } l = k \\ \sum_{i=1}^{m} \frac{1}{m_l} \mathbb{I}(h(x_i) = k)\mathbb{I}(y_i = l) & \text{otherwise,} \end{cases}$$

This definition also applies for the multi-view classifier defined in Equation 4.1, which brings us to the minimization problem. Let $\mathcal{H}_j$ be the classifier space associated to view $j$, $\forall j \in \{1,\dots,v\}$, then the goal is to find a classifier $\hat{H}$ as in Equation 4.1 that verifies:

$$\hat{H} = \underset{\forall j, h_j \in \mathcal{H}_j, \forall j, c, \beta_{j,c}}{\operatorname{argmin}} \|\mathbf{C}\|, \text{ s.t. } \forall c \in \{1,\dots,K\}, \sum_{j=1}^{v} \beta_{j,c} = 1,$$

where $\mathbf{C}$ is the confusion matrix associated to $\hat{H}$, and $\|\cdot\|$ is the operator norm of the matrix (see section 3.1.2). As in section 3.2, the direct minimization of the operator norm of the confusion matrix is quite challenging and we choose to minimize an upper bound of the norm. In section 3.2, we argued that an upper bound of the operator norm is the entry-wise $l1$-norm, which is also an upper bound of the (squared) entry-wise $l2$-norm, also known as the Frobenius norm. Finally, the optimization problem that we'll be dealing with in this chapter is the following:

$$\hat{H} = \underset{\forall j, h_j \in \mathcal{H}_j, \forall j, c, \beta_{j,c}}{\operatorname{argmin}} \|\mathbf{C}\|_1, \text{ s.t.}$$

$$\forall c \in \{1,\dots,K\}, j \in \{1,\dots,v\}, \beta_{j,c} \in \{0,1\} \text{ and } \forall c \in \{1,\dots,K\}, \sum_{j=1}^{v} \beta_{j,c} = 1.$$

### 4.2.2 On why binary is simpler but not the best

Equation 4.2 sets a condition for the optimization problem: the values of the coefficients should be either 0 or 1 (a binary choice). The interest of this choice is that it associates one class to one view, thus the aim is to find for each class the view that better recognizes it. This also implies that the confusion matrix of the classifier $H$ is made up of lines coming from the confusion matrices of the classifiers learnt of the views. Despite these interesting properties and the fact that the coefficients are easy

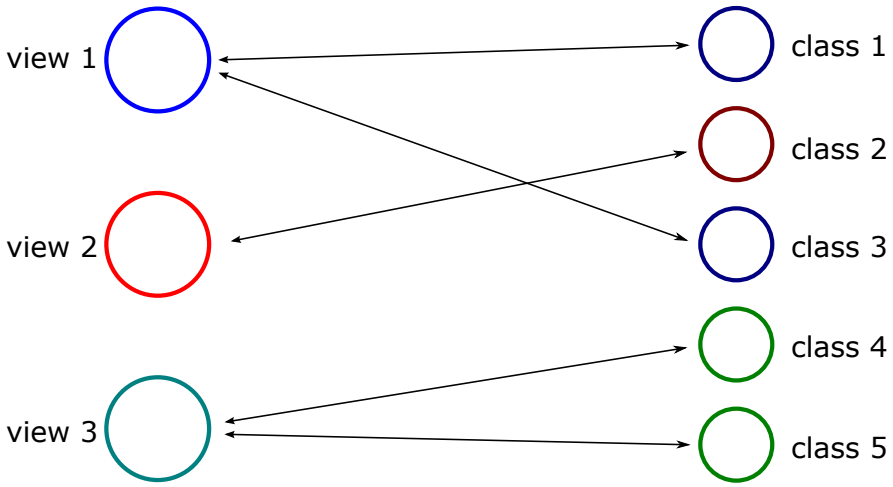



Figure 4.2 – *A diagram connecting the views and the classes they recognize.*

to interpret, unfortunately the binary values for the coefficients are not always the good choice. In order to illustrate this, let us take the following example.

Suppose that the number of classes for a given problem is five and the number of views defined over the examples is three. Suppose also that the solution of the learning problem associates view 1 with classes 1 and 3, view 2 with class 2 and view 3 with classes 4 and 5. As previously argued, this means that the views are responsible for correctly detecting their corresponding classes. The connections between the views and the classes are given in the diagram in figure 4.2. Now suppose that for a given example and for whatever reasons, the classifier of view 1, $h_1$, predicts class 2, the classifier of the second view, $h_2$ predicts class 1 and $h_3$ predicts class 2. Now, computing the predictions as given in Equation 4.1, we obtain the following score for all the classes:

$$\begin{aligned}\text{class 1: } & \beta_{1,1}\mathbb{I}[h_1(x)=1]+\beta_{2,1}\mathbb{I}[h_2(x)=1]+\beta_{3,1}\mathbb{I}[h_3(x)=1]\\ & =1\cdot 0+0\cdot 1+0\cdot 1=0\end{aligned}$$

$$\begin{aligned}\text{class 2: } & \beta_{1,2}\mathbb{I}[h_1(x)=2]+\beta_{2,2}\mathbb{I}[h_2(x)=2]+\beta_{3,2}\mathbb{I}[h_3(x)=2]\\ & =0\cdot 1+1\cdot 0+0\cdot 0=0\end{aligned}$$

$$\begin{aligned}\text{class 3: } & \beta_{1,3}\mathbb{I}[h_1(x)=3]+\beta_{2,3}\mathbb{I}[h_2(x)=3]+\beta_{3,3}\mathbb{I}[h_3(x)=3]\\ & =1\cdot 0+0\cdot 0+0\cdot 0=0\end{aligned}$$

$$\begin{aligned}\text{class 4: } & \beta_{1,4}\mathbb{I}[h_1(x)=4]+\beta_{2,4}\mathbb{I}[h_2(x)=4]+\beta_{3,4}\mathbb{I}[h_3(x)=4]\\ & =0\cdot 0+0\cdot 0+1\cdot 0=0\end{aligned}$$

$$\begin{aligned}\text{class 5: } & \beta_{1,5}\mathbb{I}[h_1(x)=5]+\beta_{2,5}\mathbb{I}[h_2(x)=5]+\beta_{3,5}\mathbb{I}[h_3(x)=5]\\ & =0\cdot 0+0\cdot 0+1\cdot 0=0\end{aligned}$$

So, each class obtains the same score, that is 0, which means that for this example, the final classifier cannot decide.

This example shows that using binary values for the coefficients can be quite tricky, since it can lead to situations where the final classifier is not able to output a prediction. One way to bypass this, is to enforce the hypothesis on the views and require each view to recognize perfectly well at least one class and one class to be

recognized by at least one view. Obviously this condition is quite strong and difficult to achieve for real-life problems. A more realistic solution is to consider continuous values (between 0 and 1) for the coefficients, which can be seen as the confidence of a view recognizing a class. This finally brings us to the true optimization problem for this chapter:

$$\hat{H} = \underset{\forall j, h_j \in \mathcal{H}_j, \forall j,c, \beta_{j,c}}{\operatorname{argmin}} \|\mathbf{C}\|_1, \text{ s.t.}$$

$$\forall c \in \{1,\dots,K\}, j \in \{1,\dots,v\}, \beta_{j,c} \in [0,1] \text{ and } \forall c \in \{1,\dots,K\}, \sum_{j=1}^{v} \beta_{j,c} = 1. \quad (4.3)$$

### 4.2.3 Ensemble methods based on class cooperation

As argued in section 3.2, instead of minimizing the norm of the true confusion matrix — which depends on the unknown distribution $\mathcal{D}$ (see definition 9) —, we choose to minimize the norm the empirical confusion matrix, which is a fair approximation (see Corollary 2 in Section 3.2). Using the same arguments as in Equation 3.7, we have the following bound on the norm of the confusion matrix:

$$\begin{aligned}\|\mathbf{C}_S\|_1 &= \sum_{l=1}^{K}\sum_{c\neq l}\frac{1}{m_l}\sum_{i=1}^{m}\mathbb{I}(y_i = l)\mathbb{I}(H(i) = c) \\ &= \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\mathbb{I}(H(i) = c) \leq \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\ell_{y_i,c}(h, x_i). \end{aligned} \quad (4.4)$$

Recall that $\ell_{l,k}(h,x)$ defines the loss of $h$ choosing class $k$ over class $l$ for the example $x$, such that $\forall (x,y) \in S$ and $1 \leq l,k \leq K$, $0 \leq \mathbb{I}(h(x) \neq y) \leq \ell_{l,k}(h,x)$.

The classifier used in this chapter, defined in Equation 4.1, associates to an example $i$ the class $c$ that obtains the highest score $\phi(i,c)$, which are computed as follows:

$$\forall i \in \{1,\cdots,m\}, c \in \{1,\cdots,K\},\ \phi(i,c) = \sum_{j \in \{1\cdots v\}} \beta_{j,c}\mathbb{I}[h_j(i) = c].$$

It is easy to verify [1] that the loss $\exp(\phi(i,c) - \phi(i,y_i))$, defined for an example $(x_i, y_i) \in S$, satisfies the conditions put on the losses $\ell_{y_i,c}$. Replacing these losses in the bound given in Equation 4.4, we have:

$$\begin{aligned}\|\mathbf{C}_S\|_1 &\leq \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\ell_{y_i,c}(h,x_i) \\ &\leq \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp(\phi(i,c) - \phi(i,y_i)) \\ &= \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp\left(\sum_{j=1}^{v}\beta_{j,l}\mathbb{I}[h_j(i) = l] - \sum_{j=1}^{v}\beta_{j,k}\mathbb{I}[h_j(i) = k]\right) \\ &= \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp\left(\sum_{j=1}^{v}\left[\beta_{j,l}\mathbb{I}[h_j(i) = l] - \beta_{j,y_i}\mathbb{I}[h_j(i) = y_i]\right]\right) \end{aligned} \quad (4.5)$$

The last expression gives rise to a first method for learning a multi-view classifier (as defined in Equation 4.1) based on the cooperation between the different views.

1. An extensive explanation for the choice of the exponential loss and why it satisfies the conditions is given in section 3.2.

**Algorithm 5:** Carako version 1

**Given** :
$S = \{(x_i, y_i)\}_{i=1}^{m}$ where $x_i \in X_1 \times X_2 \times ... \times X_v$, $y_i \in \{1, ..., K\}$

*First method:*
Compute $\beta_{j,c}, h_j$ minimizing Equation 4.5, for $1 \leq c \leq K, 1 \leq j \leq v$, *w.r.t.* the conditions in Eq. 4.3

*Second method:*
**Initialize** :
For all $1 \leq c \leq K, 1 \leq j \leq v, \beta_{j,c} =$ rand(0,1) (w.r.t. to Eq. 4.3)

**while** *stopping criterion* not met:

1. Learn $h_1, \ldots, h_v$ minimizing Equation 4.5 (with fixed $\beta_{j,c}$)
2. Compute $\beta_{j,c}$ minimizing Eq. 4.5 using the classifiers from 1. *w.r.t.* the conditions in Eq. 4.3

**Output** : $H(\cdot) = \text{argmax}_{c \in \{1, \cdots, K\}} \sum_{j=1}^{v} \beta_{j,c} \mathbb{I}[h_j(\cdot) = c]$

This formulation is quite similar to the most general formulation of Multiple Kernel Learning (MKL) methods(Lanckriet et al. (2004)), where the kernels and the weighting coefficients are leant at the same time. The differences being that in our case the classifiers are not limited to kernels, the number of coefficients is bigger and we use the exponential as a loss function.

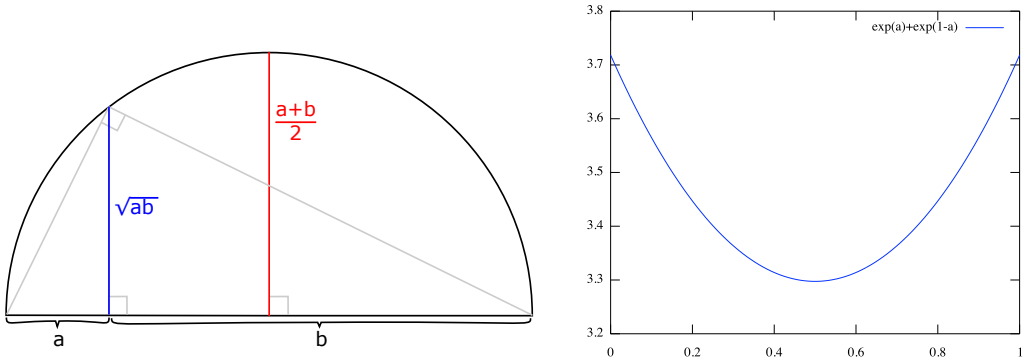
Another possible approach for this method is to use a "two steps" procedure: start by fixing some random values for the coefficients and use them to train classifiers that minimize Equation 4.5; then use the predictions of the classifiers to compute new values for the coefficients that that minimize 4.5. This two-steps procedure is to be reiterated till a stopping criterion is met (for instance, the error on the training set is zeroed or it does not change after few iterations, etc.). The pseudo code for both methods is given in algorithm 5.

It is interesting to notice that the first method uses the cooperation between the views both in the training phase of the classifiers and in the choice of the coefficients. However, due to the expression of Equation 4.5, the classifiers need to be learnt at the same time — which can make the learning procedure quite tedious — and it is not quite clear what is the minimization goal for each classifier learnt on the views. This is why we propose to simplify the expression of Equation 4.5 and to limit the cooperation between the views only to the choice of the coefficients. In order to do that, we will make use of the following theorem.

**Proposition 2** *Let $a, b \in \mathbb{R}$, so that $a, b \leq 1$ and $a + b \leq 1$, then the following inequality is true:*

$$\exp(a + b) \leq \exp(a) + \exp(b). \tag{4.6}$$

*Proof.* In order to prove the result of the theorem, we consider two cases: when at least one of $a$ and $b$ is negative, and when both $a$ and $b$ take values in $[0, 1]$.




(a) *Arithmetic mean versus geometric mean,* $a, b \geq 0$

(b) *The graph of* $\exp(a) + \exp(1-a), a \in [0,1]$

Figure 4.3 – *The left figure shows the relation between the arithmetic and geometric mean, while the right figure shows the exponential function.*

**case:** $a \leq 0$ **and/or** $b \leq 0$ If at least one of $a$ and $b$, then the inequality 4.6 holds, since the exponential of a negative number is a positive number at most 1. Suppose (without loss of generality) that $a \leq 0$. Then we have:

$$\exp(a+b) = \exp(a)\exp(b) \leq \exp(b) \leq \exp(a) + \exp(b).$$

**case:** $a, b \in ]0,1]$ The previous reasoning cannot be used in this case, since the exponential of both $a$ and $b$ is bigger than 1. Instead, we will use the relation between the arithmetic and geometric means. Recall that for two positive numbers $x$ and $y$, the arithmetic mean is defined as $\frac{x+y}{2}$ and the geometric mean is equal to $\sqrt{xy}$, and the geometric mean is smaller than the arithmetic one (a not-so-formal proof is given in figure 4.3a):

$$\sqrt{xy} \leq \frac{x+y}{2} \Leftrightarrow xy \leq \frac{(x+y)^2}{4}.$$

Using this property of the means, we have:

$$\begin{aligned}
\exp(a+b) = \exp(a)\exp(b) &\leq \frac{(\exp(a)+\exp(b))^2}{4} \\
&= (\exp(a)+\exp(b))\frac{(\exp(a)+\exp(b))}{4} \\
&\leq (\exp(a)+\exp(b))\frac{(\exp(a)+\exp(1-a))}{4} \\
&\leq \exp(a)+\exp(b)
\end{aligned}$$

The first inequality comes for the relation between the means; for the second inequality, we used the hypothesis that $a+b$ is at most one (and exponential is an increasing function). The last inequality comes from the fact that, for the interval $[0,1]$, the maximum value of the function $f : x \to e^x + e^{1-x}$ is $1+e$. This function is also shown in figure 4.3b. □

The following theorem is a generalization of Proposition 2 for more than two variables.

**Theorem 6** *Let* $n \in \mathbb{N}$ *and* $a_1, \ldots, a_n \leq 1$, *so that* $\sum_{j=1}^{n} a_j \leq 1$. *Then the following inequality is true:*

$$\exp(\sum_{j=1}^{n} a_j) \leq \sum_{j=1}^{n} \exp(a_j). \tag{4.7}$$

**Algorithm 6:** Carako version 2

**Given** :
$S = \{(x_i, y_i)\}_{i=1}^{m}$ where $x_i \in X_1 \times X_2 \times ... \times X_v$, $y_i \in \{1, ..., K\}$

*First method:*
Compute $\beta_{j,c}, h_j$ minimizing Equation 4.8, for $1 \leq c \leq K, 1 \leq j \leq v$, *w.r.t.* the conditions in Eq. 4.3

*Second method:*
**Initialize** :
For all $1 \leq c \leq K, 1 \leq j \leq v, \beta_{j,c} =$ rand(0,1) (w.r.t. to Eq. 4.3)

**while** *stopping criterion* not met:

1. $\forall j \in \{1, \ldots, v\}$, train $h_j$ minimizing the loss $\ell(h_j)$ (Eq. 4.9)
2. $\forall c \in \{1, \ldots, K\}$, compute $\beta_{j,c}$ minimizing the loss $\ell(c)$ (Eq. 4.10), *w.r.t.* the conditions in Eq. 4.3

**Output** : $H(\cdot) = \text{argmax}_{c \in \{1, \cdots, K\}} \sum_{j=1}^{v} \beta_{j,c} \mathbb{I}[h_j(\cdot) = c]$

*Proof.* Suppose that (without loss of generality) the variables are sorted based on their values, from the highest to the lowest, that is, we suppose that :

$$a_1 \geq a_2 \geq \ldots \geq a_n.$$

Using the result of Proposition 2 we have:

$$\exp(a_1 + \sum_{j=2}^{n} a_j) \leq \exp(a_1) + \exp(\sum_{j=2}^{n} a_j).$$

The condition on the sorted variables is necessary since it ensures that the remaining sum, $\sum_{j=2}^{n} a_j$, is smaller than 1, thus verifying the conditions in Proposition 2. Reiterating over the remaining sum gives:

$$\exp(a_1 + \sum_{j=2}^{n} a_j) \leq \exp(a_1) + \exp(\sum_{j=2}^{n} a_j) \leq \exp(a_1) + \exp(a_2) + \exp(\sum_{j=3}^{n} a_j)$$
$$\leq \ldots \leq \sum_{j=1}^{k} \exp(a_j) + \exp(\sum_{j=k+1}^{n} a_j) \leq \ldots \leq \sum_{j=1}^{n} \exp(a_j).$$

□

The first condition we set on the coefficients in the optimization problem in Equation 4.3 is that their values should be in $[0, 1]$. The second condition requires that, for each class, the coefficients sum up to 1. Together, these conditions imply that the inner sums in Equation 4.5 take their values in $[-1, 1]$, same as each of their elements. The consequence is that we can apply Theorem 6 to further simplify the expression given in Equation 4.5.

$$
\begin{aligned}
\|\mathbf{C}_{\mathcal{S}}\|_1 &\leq \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\ell_{y_i,c}(h,x_i)\\
&\leq \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp\left(\sum_{j=1}^{v}\left[\beta_{j,l}\mathbb{I}[h_j(i)=l]-\beta_{j,y_i}\mathbb{I}[h_j(i)=y_i]\right]\right)\\
&\leq \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\sum_{j=1}^{v}\exp\left(\beta_{j,l}\mathbb{I}[h_j(i)=l]-\beta_{j,y_i}\mathbb{I}[h_j(i)=y_i]\right)\\
&= \sum_{j=1}^{v}\sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp\left(\beta_{j,l}\mathbb{I}[h_j(i)=l]-\beta_{j,y_i}\mathbb{I}[h_j(i)=y_i]\right) \qquad (4.8)
\end{aligned}
$$

We have thus:

$$\|\mathbf{C}_{\mathcal{S}}\|_1 \leq \sum_{j=1}^{v}\ell(h_j), \qquad (4.9)$$

where $\ell(h_j) = \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp\left(\beta_{j,l}\mathbb{I}[h_j(i)=l]-\beta_{j,y_i}\mathbb{I}[h_j(i)=y_i]\right)$, defines the loss of the classifier $h_j$ — whose predictions are weighted by the coefficients $\beta_{j,c}, \forall c \in \{1,\ldots,K\}$ — on the training set $S$. Minimizing the norm of the confusion matrix for a multi-view classifier $H$, as defined in Equation 4.1, ends up finding the classifiers and coefficients that minimize the loss $\sum_{j=1}^{v}\ell(h_j)$. Moreover, for fixed values of the coefficients, Equation 4.9 suggests that, for each view, it suffices to find the classifier whose error $\ell(\cdot)$ is minimal, instead of finding the classifiers that minimize a loss depending on their combination (which was the case in Equation 4.5).

Although the training procedure where all the coefficients and classifiers are learnt at the same time is still a viable solution, the previous remark suggests that the two-step procedure is better adapted for this case. The first step consists in finding, for each view, the classifier whose error $\ell(\cdot)$ is minimal. The second step consists in finding the coefficients that minimize the whole loss (Equation 4.8). It is interesting to notice that due to the formulation of Equation 4.8, it is possible to learn the coefficients of one class independently from the coefficients of the other classes. More precisely, let $S_j^+$ denote the set of examples correctly classified by $h_j$ and $S_j^-$ the set of misclassified ones. The result in Equation 4.8 can be written as [2]:

2. In Equation 4.10, due to the lack of space, $e$ is used instead of exp is some Equations.

$$
\begin{aligned}
\|\mathbf{C}_{\mathcal{S}}\|_1 \leq & \sum_{j=1}^{v}\sum_{i=1}^{m}\sum_{l\neq y_i}\frac{1}{m_{y_i}}\exp\Big(\beta_{j,l}\mathbb{I}[h_j(i)=l]-\beta_{j,y_i}\mathbb{I}[h_j(i)=y_i]\Big)\\
= & \sum_{j=1}^{v}\sum_{i\in S_j^+}\sum_{l\neq y_i}\frac{1}{m_{y_i}}e^{-\beta_{j,y_i}}+\sum_{j=1}^{v}\sum_{i\in S_j^-}\frac{1}{m_{y_i}}\Big(\sum_{l\neq y_i\neq h_j(i)}[e^0+e^{-\beta_{j,h_j(i)}}]\Big)\\
= & \sum_{j=1}^{v}\sum_{i\in S_j^+}\frac{K-1}{m_{y_i}}e^{-\beta_{j,y_i}}+\sum_{j=1}^{v}\sum_{i\in S_j^-}\Big(\frac{K-2}{m_{y_i}}+\frac{1}{m_{y_i}}e^{-\beta_{j,h_j(i)}}\Big)\\
= & \sum_{j=1}^{v}\sum_{c=1}^{K}\sum_{i\in S_j^+}\frac{K-1}{m_{y_i}}\exp(-\beta_{j,c})\mathbb{I}[y_i=c]\\
& +\sum_{j=1}^{v}\sum_{c=1}^{K}\sum_{i\in S_j^-}\Big(\frac{K-2}{m_{y_i}}+\frac{1}{m_{y_i}}\exp(\beta_{j,c})\Big)\mathbb{I}[h_j(i)=c]\\
= & \sum_{c=1}^{K}\sum_{j=1}^{v}\Big(e^{-\beta_{j,c}}\sum_{i\in S_j^+}\frac{K-1}{m_{y_i}}\mathbb{I}[y_i=c]+e^{\beta_{j,c}}\sum_{i\in S_j^-}\frac{\mathbb{I}[h_j(i)=c]}{m_{y_i}}\Big)\\
& +\sum_{c=1}^{K}\sum_{j=1}^{v}\sum_{i\in S_j^-}\frac{K-2}{m_{y_i}}\mathbb{I}[h_j(i)=c] \qquad (4.10)
\end{aligned}
$$

The first equality is obtained by splitting the training sample in $S_j^+$ and $S_j^-$, for all views $j$. In the third equality, we replace $y_i$ and $h_j(i)$ by a label $c$, which allows us to regroup the non constant terms (that is, those depending on $\beta_{j,c}$) in the last equation.

This last equation suggests that for a given class $c$, the coefficients $\beta_{j,c}, \forall j \in \{1,\ldots,v\}$ should be the ones that minimize the *per class* loss $\ell(c)$:

$$
\sum_{j=1}^{v}\Big(e^{-\beta_{j,c}}\sum_{i\in S_j^+}\frac{K-1}{m_{y_i}}\mathbb{I}[y_i=c]+e^{\beta_{j,c}}\sum_{i\in S_j^-}\frac{\mathbb{I}[h_j(i)=c]}{m_{y_i}}\Big) \qquad (4.11)
$$

The pseudo-codes for both methods (batch learning and two-steps) are given in algorithm 6. The *stopping criterion* is the same as in algorithm 5, and as such it can be related to the empirical loss of the classifier computed on the training set $S$, or the drop of the loss in Equation 4.11, and so on.

Contrary to the methods given in algorithm 6, in this case, the cooperation between the views is only embedded in the cooperation coefficients $\beta_{j,c}$. On the positive side though, in the two-step method, the classifiers can be learnt in parallel, reducing the training time for the algorithms.

## 4.3 Towards a boosting method

### 4.3.1 Deriving a(nother) boosting method from the confusion matrix norm minimization

The two-step procedures in both algorithms 5 and 6 suggest that at each iteration the learnt classifiers should minimize some training error weighted by some cooperation coefficients associated to the training examples. Particularly, in algorithm 6, the learning procedure for each view consists in learning a classifier that minimizes a

weighted training error, re-weighting the examples based on the performances of the classifier and reiterating till a stopping criterion is met. Interestingly, the procedure is fairly similar to iterative boosting methods presented in Section 1.2 (and the ones in Chapter 2 and Chapter 3). This brings us to the next step, which consists in replacing the classifiers $h_j$ in Equation 4.9 with boosted classifiers.

An iterative boosting method runs for $T$ steps and its output hypothesis is computed as follows:

$$h(\cdot) = \text{argmax}_{c\in\{1,\dots,K\}} \sum_{t=1}^{T} \alpha_t \mathbb{I}[h_t(\cdot) = c],$$

where, $\alpha_t$ are positive real-valued coefficients that represent the importance given to $h_t$. When defining the multi-view classifier in Equation 4.1, we argued that each classifier should be associated with an importance coefficient depending on the prediction. Since our goal is to use boosted classifiers made up of weak classifiers [3], having a single coefficient per class and view might not be a good strategy, since each of the weak classifiers (learnt on one view) has different performances. This is why we propose to associate to each classifier not only its importance coefficient, but also coefficients depending on its actual prediction:

$$h_j(\cdot) = \underset{c\in\{1,\dots,K\}}{\text{argmax}} \sum_{t=1}^{T} \alpha_{t,j}\beta_{t,j,c}\mathbb{I}[h_{t,j}(\cdot) = c], \text{ where } \beta_{t,j,c} \in [0,1]. \tag{4.12}$$

In this case the coefficient associated to a view $j$ for a class $c$ is :

$$\beta_{j,c} = (\sum_{t=1}^{T} \alpha_{t,j}\beta_{t,j,c}) / \sum_{t=1}^{T} \alpha_{t,j}.$$

Keeping the condition that $\forall t \in \{1,\dots,T\}, \forall c \in \{1,\dots,K\}, \sum_{j=1}^{v} \beta_{t,j,c} = 1$ , ensures that $\beta_{j,c}$ is always smaller than 1. The downside is that it does not guarantee that $\forall c \in \{1,\dots,K\}, \sum_{j=1}^{v} \beta_{j,c} = 1$, but rather that $\sum_{j=1}^{v} \beta_{j,c} \leq 1$.

The classifier defined in Equation 4.12 is similar to the classifier defined in 4.1: for a given example, it computes a score for every class. In particular for examples in the training sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, these scores are computed as follows:

$$f_{T,j}(i,c) = \sum_{t=1}^{T} \alpha_{t,j}\beta_{t,j,c}\mathbb{I}[h_{t,j}(i) = c], \text{ for } 1 \leq j \leq v, 1 \leq c \leq K, 1 \leq i \leq m.$$

Armed with these score functions, we are now ready to tackle the last optimization

3. See Section 1.2 for the definition of weak classifier/learner.

**Algorithm 7:** $\mu$CoMBo

**Given**

— S = $\{(x_1, y_1), ..., (x_m, y_m)\}$ where $x_i \in X_1 \times ... \times X_v$, $y_i \in \{1, ..., K\}$

— T the number of iterations

— $\forall i \in \{1, \ldots, m\}$, $\forall j \in \{1, \ldots, v\}$, $\forall c \in \{1, \ldots, K\}$ $f_{0,j}(i, c) = 0, \beta_{0,j,c} = 1/v$ and

$$\mathbf{D}_{0,j}(i, c) = \begin{cases} \frac{1}{m_{y_i}} & \text{if } l \neq y_i \\ -\frac{K-1}{m_{y_i}} & \text{if } l = y_i \end{cases}$$

**for** $t = 1$ **to** $T$ **do**

$\forall j$: Get $h_{t,j}$ and $\alpha_{t,j} = \frac{1}{2} \ln \frac{1+\delta_{t,j}}{1-\delta_{t,j}}$, where $\delta_t = \frac{-\sum_{i=1}^{m} \mathbf{D}_{t,j}(i, h_{t,j}(\mathbf{x}_i))}{\sum_{i=1}^{m} \sum_{c \neq y_i} \mathbf{D}_{t,j}(i,c)}$

Compute $\beta_{t,j,c}$, $\forall, j \in \{1, \ldots, v\}, c \in \{1, \ldots, K\}$ minimizing Eq. 4.13

Update cost matrices (for each $j = 1, \ldots, V$):

$$\mathbf{D}_{t,j}(i, c) = \begin{cases} \frac{1}{m_{y_i}} \exp(f_{t,j}(i, c) - f_{t,j}(i, y_i)) & \text{if } c \neq y_i \\ -\frac{1}{m_{y_i}} \sum_{l=1; l \neq y_i}^{K} \exp(f_{t,j}(i, l) - f_{t,j}(i, y_i)) & \text{if } c = y_i \end{cases}$$

$$\text{where } f_{t,j}(i, c) = \sum_{z=1}^{t} \alpha_{z,j} \beta_{z,j,c} \mathbb{I}1[h_{z,j}(i) = c]$$

**end for**

Output final hypothesis :

$$H(\cdot) = \underset{c \in 1,...,K}{\operatorname{argmax}} \sum_{t=1}^{T} \sum_{j=1}^{v} \beta_{t,j,c} \alpha_{t,j} \mathbb{I}[h_{t,j}(\cdot) = c]$$

problem of this chapter. Continuing from where we left off in Equation 4.8, we have:

$$\begin{aligned} \|\mathbf{C}_{\mathcal{S}}\|_1 &\leq \sum_{j=1}^{v} \sum_{i=1}^{m} \sum_{c \neq y_i} \frac{1}{m_{y_i}} \exp\left(\beta_{j,c} \mathbb{I}[h_j(i) = c] - \beta_{j,y_i} \mathbb{I}[h_j(i) = y_i]\right) \\ &\leq \sum_{j=1}^{v} \sum_{i=1}^{m} \sum_{c \neq y_i} \frac{1}{m_{y_i}} \exp\left(\beta_{j,c} \mathbb{I}[h_j(i) = c]\right) \\ &\leq \sum_{j=1}^{v} \sum_{i=1}^{m} \sum_{c \neq y_i} \frac{1}{m_{y_i}} \exp\left(\mathbb{I}[h_j(i) = c]\right) \\ &\leq \sum_{j=1}^{v} \sum_{i=1}^{m} \sum_{c \neq y_i} \frac{1}{m_{y_i}} \exp\left(\ln(2 + \exp\left[f_{T,j}(i, c) - f_{T,j}(i, y_i)\right])\right) \\ &= \sum_{j=1}^{v} \sum_{i=1}^{m} \sum_{c \neq y_i} \frac{1}{m_{y_i}} \exp\left(f_{T,j}(i, c) - f_{T,j}(i, y_i)\right) + 2vm(K-1) \end{aligned} \tag{4.13}$$

The second inequality comes from the fact that $\beta_{j,y_i}$ is positive, while in the third inequality we used the fact that $\beta_{j,c}$ is at most 1. For the forth inequality a particular case of the logistic loss (defined in Equation 1.2). If $h_j$ predicts class $c$ for example $i$ ($\mathbb{I}[h_j(i) = c] = 1$), then $f_{T,j}(i, c) \geq f_{T,j}(i, y_i)$ and $\exp(f_{T,j}(i, c) - f_{T,j}(i, y_i)) \geq 1$. Since the difference between the score might be infinitely small, we use 2 in the logistic loss, instead of 1 in the original definition, which ensures that $\ln(2 + \exp\left[f_{T,j}(i, c) -\right.$

$f_{T,j}(i,y_i)\Big]) \geq 1$. In the case where $h_j$ predicts another class for $i$, other than $c$, then $ln(2+\exp\Big[f_{T,j}(i,c)-f_{T,j}(i,y_i)\Big]) > 0$, since $\exp(a) > 0, \forall a \in \mathbb{R}$.

We have thus:

$$\|\mathbf{C}_{\mathcal{S}}\|_1 \leq \sum_{j=1}^{v} \ell(h_j) + 2vm(K-1), \tag{4.14}$$

where $\ell(h_j) = \sum_{i=1}^{m}\sum_{c\neq y_i} \frac{1}{m_{y_i}} \exp\Big(f_{T,j}(i,c) - f_{T,j}(i,y_i)\Big)$, defines the loss of the combinations of all the classifiers learnt on view $j$. Equation 4.13 suggests that for each view and at each iteration, a classifier that minimizes the loss $\ell(h_j)$ should be learnt. It is interesting to notice that the loss $\ell(h_j)$ is quite similar to the loss of CoMBo, as given in section 3.2 (page 58) except for the coefficients $\beta_{t,j,c}$. In other words, it means that a CoMBo-like process should take place in each view. It follows that a cost matrix should be maintained for each view and the classifier (and its importance coefficient) should be computed in a similar way as in CoMBo; in particular the weak classifier at each iteration should verify the weak learning condition (given in Equation 1.5, page 9) for the cost matrix associated to the view.

Similarly to the two-steps procedures in algorithms 5 and 6, the algorithm derived from the minimization of the loss in Equation 4.13 uses a two step procedure at each iteration: first a classifier is learnt for each view and the importance coefficient is computed as in algorithm 7, and second, the cooperation coefficients are computed so that they minimize the loss. The pseudo-code of the new method is given in algorithm 7. Due to the boosting nature of the method, the stopping criterion used is the number of iterations.

### 4.3.2 On the theoretical properties of $\mu$CoMBo

In this section we show that the general loss given in Equation 4.13 is driven down at each iteration, provided that each of the classifiers learnt on the views satisfies the weak learning condition (Equation 1.5) and the importance coefficients $\alpha$ are chosen as given in algorithm 7. Interestingly enough, Equation 4.14 suggests that the whole loss in made up of the losses calculated on each of the views, thus showing that each of the per view losses decreases with each iteration w.r.t. the weak classifiers is sufficient. Theorem 7 proves that the per view loss does indeed decrease after each iteration. This result and its proof are similar to the ones given for MuMBo in Theorem 1 (page 35) and CoMBo in Lemma 2 (page 60).

**Theorem 7** *For a given view $j \in \{1,\ldots,v\}$, suppose the cost matrix $\mathbf{D}_{t,j}$ is chosen as in the algorithm 7, and the returned classifier $h_{t,j}$ satisfies the edge condition for the baseline $\mathbf{U}_{\delta_{t,j}}$ and cost matrix $\mathbf{D}_{t,j}$, i.e. $\mathbf{D}_{t,j}\cdot\mathbf{1}_{h_{t,j}} \leq \mathbf{D}_{t,j}\cdot\mathbf{U}_{\delta_{t,j}}$, where $\mathbf{1}_h$ is the matrix defined as $\mathbf{1}_h(i,l) = \mathbb{I}[h(i)=l]$.*

*Then choosing a weight $\alpha_{t,j} > 0$ for $h_{t,j}$, allows*

$$\ell_t(h_j) \leq \kappa_{t,j}\ell_{t-1}(h_j),$$

*to hold, with:*

$$\kappa_{t,j} = 1 - \frac{1}{2}\left(e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}\right)\delta_{t,j} + \frac{1}{2}\left(e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2\right)$$

*Proof.* We give here a sketch of the proof in the same fashion as Lemma 2 (page 60), since most of the computation are similar to the ones given in Theorem 1 (page 35).

First, in order to simplify the reading of this proof, we define the two following losses:

$$L_{t,j} = \sum_{i=1}^{m} \sum_{l \neq y_i} \mathbf{D}_{t,j}(i,l) = \sum_{i=1}^{m} \sum_{l \neq y_i} \frac{1}{m_{y_i}} \exp(f_{t,j}(i,l) - f_{t,j}(i,y_i)) \text{ and}$$
$$L_{t,j}(i) = \sum_{l \neq y_i} \mathbf{D}_{t,j}(i,l) = \sum_{l \neq y_i} \frac{1}{m_{y_i}} \exp(f_{t,j}(i,l) - f_{t,j}(i,y_i)).$$

The only hypothesis we made was that the weak classifiers $h_t$ should verify the weak learning condition (Equation 1.5 in page 9), that is:

$$\mathbf{D_{t,j}} \cdot \mathbf{1}_{h_{t,j}} \leq \mathbf{D_{t,j}} \cdot \mathbf{U}_{\delta_{t,j}}, \tag{4.15}$$

with $\delta_{t,j}$ being the edge of $h_{t,j}$ on $\mathbf{D_{t,j}}$.

Let $S_+$ (resp. $S_-$) be the set of examples of $S$ correctly classified (resp. misclassified) by $h_{t,j}$. The left and right side of Equation 4.15 can be written as follows, using the definitions of $\mathbf{D_{t,j}}$, $\mathbf{1}_{h_{t,j}}$ and $\mathbf{U}_{\delta_{t,j}}$:

$$\begin{aligned}
\mathbf{D}_{t,j} \cdot \mathbf{1}_{h_{t,j}} &= -\sum_{i \in S_+} L_{t-1,j}(i) + \sum_{i \in S_-} \frac{1}{m_{y_i}} \exp\left(f_{t-1,j}(i, h_{t,j}(i)) - f_{t-1,j}(i,y_i)\right) \\
&= -A_+^{t,j} + A_-^{t,j} \\
\mathbf{D}_{t,j} \cdot \mathbf{U}_{\delta_{t,j}} &= -\delta_{t,j} \sum_{i=1}^{m} L_{t-1,j}(i) = -\delta_{t,j} L_{t-1,j}
\end{aligned}$$

Injecting these two costs in 4.15, we have :

$$A_+^{t,j} - A_-^{t,j} \geq \delta_{t,j} L_{t-1,j}. \tag{4.16}$$

Computing the drop of the loss after choosing $h_t$ and $\alpha_t$, we have:

$$\begin{aligned}
L_{t-1,j} - L_{t,j} &= \sum_{i \in S_+} L_{t-1,j}(i)(1 - e^{-\alpha_{t,j}}) + \sum_{i \in S_-} \frac{1}{m_{y_i}} \exp(\Delta f_{t-1,j}(i, h_{t,j}, y_i))(1 - e^{\alpha_{t,j}}) \\
&= (1 - e^{-\alpha_{t,j}}) A_+^{t,j} - (e^{\alpha_{t,j}} - 1) A_-^{t,j} \\
&= \left(\frac{e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}}{2}\right)(A_+^{t,j} - A_-^{t,j}) - \left(\frac{e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2}{2}\right)(A_+^{t,j} + A_-^{t,j})
\end{aligned}$$

where $\Delta f_{t-1,j}(i, h_{t,j}, y_i)) = f_{t-1,j}(i, h_t(i)) - f_{t-1,j}(i, y_i)$.

The result in 4.16 gives a lower bound for $A_+^{t,j} - A_-^{t,j}$, while $A_+^{t,j} + A_-^{t,j}$ is upper-bounded by $L_{t-1,j}$. Hence,

$$\begin{aligned}
L_{t-1,j} - L_{t,j} &= \sum_{i \in S_+} L_{t-1,j}(i)(1 - e^{-\alpha_{t,j}}) \\
&\geq \left(\frac{e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}}{2}\right)\delta_{t,j} L_{t-1,j} - \left(\frac{e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2}{2}\right) L_{t-1,j}.
\end{aligned}$$

Therefore, the result of the theorem:

$$\begin{aligned}
L_{t,j} &\leq \left(1 - \frac{e^{\alpha_{t,j}} - e^{-\alpha_{t,j}}}{2}\delta_{t,j} + \frac{e^{\alpha_{t,j}} + e^{-\alpha_{t,j}} - 2}{2}\right) L_{t-1,j} \\
&= \left(\frac{(1 - \delta_{t,j})}{2} e^{\alpha_{t,j}} + \frac{(1 + \delta_{t,j})}{2} e^{-\alpha_{t,j}}\right) L_{t-1,j}.
\end{aligned}$$

□

The result given in Theorem 7 shows that choosing a classifier that satisfies the weak learning condition ensures that the loss on the views decreases. It also implies that choosing the importance coefficient as given in algorithm 7, the loss for a view $j$ is simply $\sqrt{1-\delta_{t,j}}$. That is, at each iteration, after the first step which consists in choosing a classifier per view, the whole drop in loss is:

$$\sum_{j=1}^{v}\ell_t(h_j) \leq \sum_{j=1}^{v}\sqrt{1-\delta_{t,j}}\ell_{t-1}(h_j), \tag{4.17}$$
$$\text{where } \ell_t(h_j) = \sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp\Big(f_{t,j}(i,c) - f_{t,j}(i,y_i)\Big).$$

As long as the classifiers learnt on the views achieve positive edges on their corresponding cost matrices, the whole loss is guaranteed to decrease.

Note that in the right side of Equation 4.17, the loss $\ell_{t-1}(h_j)$ depends on the score functions defined as:

$$f_{t,j}(i,c) = \sum_{z=1}^{t-1}\alpha_{z,j}\beta_{z,j,c}\mathbb{I}[h_{z,j}(i)=c] + \alpha_{t,j}\beta_{t,j,c}\mathbb{I}[h_{t,j}(i)=c],$$

since the classifiers are learnt before the cooperation coefficients $\beta$, that is, we simply suppose that all these coefficients are equal to 1. Obviously the final loss, after the second step of iteration $t$, is much bigger given that all the $\beta$ take their values in $[0,1]$ and at most one of the $\beta_{t,j,c}$ is at 1 for a given class $c$. However, since these coefficients are computed as a solution to an optimization problem, finding an analytical expression for the actual drop after iteration $t$ is quite challenging.

## 4.4 Experimental results

In this section we present the experimental results for the methods proposed in this chapter, and more precisely the methods proposed in algorithm 6, Carako, and algorithm 7, $\mu$CoMBo. The dataset used in the experiments is a subset of the Animal with Attributes database proposed in Lampert et al. (2009). Originally *Animal with Attributes* is made up of 30475 images of 50 animals classes. We chose it because it embeds the two problems addressed in this chapter:

1. it is highly imbalanced: some classes are way more represented than others,
2. it comes with six pre-extracted feature representations (thus, views) for each image.

The first part of this section gives a short description of the considered dataset and the protocol used for the experimental setup. Next we present the empirical results, while discussing the observed performances.

### 4.4.1 Description of the corpus and experimental setup

**Gorillas, lions and beavers** The aim of the methods presented in this chapter is to deal with imbalanced multi-view datasets: each view is suited for recognizing few classes, and the training sample is (highly) imbalanced. *Animal with Attributes* is an excellent case study for such datasets, since it comes with an imbalanced classes problem *and* with several views embedding different type of information. In order to accentuate the imbalanced nature of *Animal with Attributes*, we propose to consider a

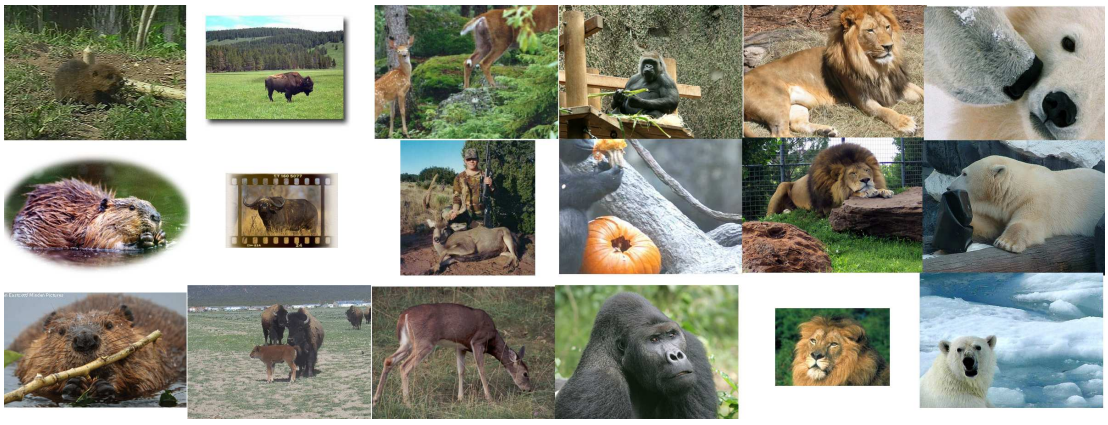


Figure 4.4 – *Examples of the six selected classes of Animal with Attributes.*

subset of the original dataset, by retaining only a few classes and views. As such, we choose six classes as follows:

— one from the *minority* classes (class *beaver*),
— one from the *majority* classes (class *deer*),
— and four other intermediate ones (classes *buffalo*, *gorilla*, *lion* and *polar bear*).

Some examples of the chosen classes are given in Figure 4.4.

Concerning the views, only four from the original six views are selected, based both on the nature of information contained therein (local versus global) and the possibility of the view to recognize all the classes or some of them. More precisely, the considered views are:

— Color Histogram features (2688 attributes),
— Local Self-Similarity features (2000 attributes),
— PyramidHOG (PHOG) (252 attributes),
— Scale-Invariant Feature Transform features (2000 attributes).

As briefly argued in the introductory example in page 2, the color histogram contains global information of the images while the other are made up of rather local information. At the same time, the first view is more adapted for *gorilla* and *polar bear* (since they are black and white respectively), while the third view is more adapted for *deer* since their shapes are easier to recognize than those of the other animals.

Table 4.1 sums up the training corpus made up of all the examples of the six classes. We report the number of classes (# cls), the number of examples (# ex), the total number of attributes of all the views (# attars), the class distribution and the imbalance ration (Imb. ratio), that is the ratio of the number of majority examples *deer* and minority ones *beaver*.

| | # cls | # ex | # attrs | # classes distribution beaver/buffalo/deer/ gorilla/lion/polar bear | Imb. ratio |
|---|---|---|---|---|---|
| total | 6 | 3915 | 6940 | 184/559/1072/802/483/815 | 5.83 |
| train | 6 | 2740 | 6940 | 130/408/744/553/338/567 | 5.72 |
| test | 6 | 1175 | 6940 | 54/151/328/249/145/248 | 6.07 |

Table 4.1 – *Description of the (smaller) Animal with Attributes dataset Lampert et al. (2009).*

**Experimental setup** The evaluation procedure consists is a simple train-test: the examples are randomly split into a training sample and a testing one. The training

| Algorithm | Acc. | *beaver* | *buffalo* | *deer* | *gorilla* | *lion* | *p.bear* |
|---|---|---|---|---|---|---|---|
| AdaBoost.MM (early) | 37.5% | 0.0 % | 0.0% | 82.9% | 61.8% | 0.0% | 6.0% |
| AdaBoost.MM (late) | 48.2% | 0.0 % | 0.0% | **91.8%** | 34.9% | 0.0% | 71.8% |
| CoMBo (early) | 41.4% | 37.0% | 19.9% | 31.1% | 58.2% | 39.3% | 53.6% |
| CoMBo (late) | 12.9% | 0.0 % | **100.0%** | 0.0 % | 0.0% | 0.0 % | 0.0% |
| Carako | 48.4% | 3.7% | 28.4% | 67.7% | 52.6% | 11.0% | 62.5% |
| MuMBo | 56.0% | 0.0% | 19.9% | 76.5% | **73.9%** | 8.3% | 73.0% |
| $\mu$CoMBo | **56.4%** | **44.4%** | 35.1% | 45.7% | 67.5% | **51.0%** | **78.2%** |

Table 4.2 – *Overall and per class accuracy for all the methods. Results in boldface indicate when one algorithm is better than the others.*

sample contains 70% of the examples, while 30% are used for testing. Seven ensemble methods are tested, divided in:

— early fusion : AdaBoost.MM and CoMBo (Algorithm 4) on the concatenation of all the views,

— late fusion : AdaBoost.MM, CoMBo and Carako (Algorithm 6)

— multi-view methods : MuMBo (Algorithm 3) and $\mu$CoMBo (Algorithm 7).

For the boosting-based methods, we used 1-level decision trees (stumps) as weak learners and they were run for 100 iterations, while for Carako, the depth of the trees was limited to 10.

We report here results obtained for the Accuracy, MAUC, G-mean and confusion matrix norm of all the methods.

### 4.4.2 Performance results

**Accuracy** Table 4.2 gives the accuracy on the testing sample for the seven methods: both the overall accuracy and the per-class one are reported. On the overall accuracy, the better performing methods are MuMBo and $\mu$CoMBo. This is an encouraging result since it means that adding the cooperation between the views, either in the input or output space, leads to better results. The same is valid for Carako also, since it performs better than the fusion methods, albeit it is comparable to late fusion AdaBoost.MM.

Concerning the per-class accuracies, as expected, both CoMBo (early fusion) and its multi-view formulation $\mu$CoMBo exhibit similar properties: they tend to "neglect" the majority classes, while focusing on the minority ones. This is even more evident for $\mu$CoMBo, which achieves more than 35% of accuracy for all the classes; although is achieves a mere 45.7% for *deer*. Symmetrically, both AdaBoost.MM (both early and late fusion) and MuMBo (which can be seen as a multi-view formulation of AdaBoost.MM) favor the majority classes over the minority ones. Interestingly, Carako's behavior is closer to MuMBo's than to $\mu$CoMBo's. This is mainly due to the use of Carako as a simple late fusion method with a hint of imbalanced learning.

Last but not least, the late fusion of CoMBo fares pretty badly on all the cases

| Algorithm | Measure | view 1 | view 2 | view 3 | view 4 |
|---|---|---|---|---|---|
| AdaBoost.MM | MAUC | 0.710 | 0.650 | 0.607 | 0.657 |
| | G-mean | 0.0 | 0.0 | 0.0 | 0.0 |
| CoMBo | MAUC | 0.759 | 0.727 | 0.685 | 0.719 |
| | G-mean | 0.412 | 0.348 | 0.326 | 0.325 |

Table 4.3 – *MAUC and G-mean for AdaBoost.MM and CoMBo on each of the views.*

(except for an excellent 100% accuracy on *buffalo*). We suppose that this is due to the fact that on each view CoMBo tries to promote the minority classes, hence resulting in classifiers with low confidence in their predictions. The combination of all the classifiers makes it quite possible for noisy/incorrect predictions to take the upper hand on the correct predictions. This intuition is further confirmed by the results in Table 4.3, where CoMBo performs better than AdaBoost in all the views (both in MAUC and G-mean), while its combination, the late fusion version of CoMBo, fails miserably.

**MAUC, G-mean and confusion** Since we are dealing with an imbalanced dataset, more relevant measures than the accuracy must be studied. In table 4.4, we present the results of the seven algorithms for three relevant measures of performance: MAUC, G-mean and the norm of the confusion matrix (proposed in Chapter 3). The empirical

| Algorithm | MAUC | G-mean | $\|\mathbf{C}\|$ |
|---|---|---|---|
| AdaBoost (early) | 0.689 | 0.0 | 1.464 |
| AdaBoost (late) | 0.748 | 0.0 | 1.573 |
| CoMBo (early) | 0.750 | 0.376 | 0.647 |
| CoMBo (late) | 0.695 | 0.0 | 2.236 |
| Carako | 0.689 | 0.252 | 0.960 |
| MuMBo | 0.778 | 0.0 | 0.946 |
| $\mu$CoMBo | **0.821** | **0.518** | **0.51** |

Table 4.4 – *MAUC, G-mean and norm of the confusion matrix for the seven methods on Animal with Attributes. Results in boldface indicate when one algorithm is better than the others.*

results strongly suggest that $\mu$CoMBo is more adapted for the problem at hand than the other methods. This is encouraging since $\mu$Combo was primarily designed for dealing with imbalanced multi-view datasets: the results in Table 4.4 acknowledge that microCombo actually fulfills our expectations.

Similarly to the results in Table 4.2, the performances of the various methods on the G-mean imply that CoMBo, $\mu$CoMBo and Carako, all promote some sort of smoothing process among the classes, thus a better equity among them. Once again AdaBoost.MM and MuMBo have poor G-mean since they fail to recognize *beaver*.

It is interesting to notice that even though Carako performs better than MuMBo G-mean wise, they have pretty much the same norm for the confusion matrix. We suppose that this is related to the per-class accuracies (Table 4.2), where both methods recognize well (resp. poorly) the same classes.

## 4.5 An imbalanced multi-view approach to DECODA

Sections 2.5 and 3.5 presented the results on DECODA, while tackling the multi-view and imbalanced aspect respectively. In this section, we regroup both aspects of the corpus and give the results for the various methods presented in this thesis. As table 3.5 (page 67) suggests, the views defined for the call classification task on the DECODA corpus are of different strength. The goal is to study whether multi-view methods can improve the performances of the strongest views (first and second view) through the use of the other views. We test six different methods:

1. MuMBo as presented in algorithm 3,
2. early fusion CoMBo (all the views merged together),

| Algorithm | error | test(gold) | test(ASR) |
|---|---|---|---|
| MuMBo | # | 31 | 38 |
| | % | 20.4 | 25.0 |
| CoMBo (early) | # | 34 | 43 |
| | % | 22.4 | 28.3 |
| CoMBo (late) | # | 30 | 38 |
| | % | 19.7 | 25.0 |
| $\mu$CoMBo | # | 30 | 41 |
| | % | 19.7 | 27.0 |
| AdaBoost (early) | # | 36 | 49 |
| | % | 23.7 | 32.2 |
| AdaBoost (late) | # | 34 | 45 |
| | % | 22.4 | 29.6 |

Table 4.5 – *Reported results for the six multi-view methods tested on the DECODA corpus.*

3. late fusion CoMBo (fusion of the decisions take separately on each view),
4. $\mu$CoMBo (the simplified version given in section 4.6),
5. early fusion AdaBoost (all the views merged together),
6. late fusion AdaBoost (fusion of the decisions take separately on each view).

The considered weak learners for all the learners are simple decision stumps. It is interesting to notice that, due to the choice of the weak learners, the six methods can be divided into two groups: the view selection methods and the views combination one. The former group includes MuMBo, early CoMBo and early AdaBoost, since at each iteration they select one stump, thus they select the view associated to the stump. $\mu$CoMBo, late CoMBo and late AdaBoost, which constitute the latter group, learn one stump per view and add *all* these classifiers to the final one. The algorithms are run for 5000 iterations and the reported results are those of the best runs. Table 4.5 summarizes the different results.

Comparing the results in Table 4.5 with the results given in Table 3.5 (page 67), it is obvious that employing multiple views in the learning process, either through multi-view methods or fusion (both early and late) ones, gives better performances than considering the views separately. For instance, in the *gold* setting, AdaBoost's error drops form 40 examples misclassified in the first view, to 34 examples for the late fusion case. This gain in performances is even more obvious for CoMBo, which passes from 42 misclassified examples in the second view, to 30 errors for the late fusion. Interestingly enough, AdaBoost and CoMBo switch places when passing form the per-view case to the fusion one. In the former case AdaBoost outperforms CoMBo in nearly all the settings, while in the latter case CoMBo is overall better than AdaBoost. The reason for this phenomenon is that the various views recognize differently the semantic categories (classes) and CoMBo takes advantage of the combination of all the views in order to select the most appropriate features. Table 4.6 shows that AdaBoost relies on the second view, for the major views, and the forth view, for the minor ones. On the other hand, CoMBo choses the first view as the most informative major view, and the fifth view for the minor ones. However, both methods tend to avoid the third view, since it is not that informative.

In the *gold* setting, nearly all the methods are equivalent, albeit MuMBo, $\mu$CoMBo and late CoMBo do perform better. However, in the *ASR* setting, the differences between the various methods are quite glaring. Among the view selection methods,

| algorithm | view 1 | view 2 | view 3 | view 4 | view 5 |
|---|---|---|---|---|---|
| AdaBoost (early) | 42.0% | 46.5% | 0% | 6.3% | 5.2% |
| CoMBo (early) | 47.7% | 42.3 % | 0% | 4.5 % | 5.5 % |
| MuMBo | 40.0% | 44.7% | 1.1% | 9.0% | 5.2% |

Table 4.6 – *Selection rate for each view, for AdaBoost (early), CoMBo (early) and MuMBo.*

MuMBo is the one that achieves the better performances, even more so when compared to early AdaBoost. This is particularly encouraging as our main motivation for using MuMBo was to be able to spread the classification weights more equally on the different views in order to be more robust when one view fails, as this is the case in the *ASR* setting where the main lexical views are affected by the high WER of the ASR transcriptions. Table 4.6 confirms that MuMBo chooses the minor view more frequently than the other two methods.

In the view combination group, the best method is late CoMBo, whose error is the same as MuMBo, while $\mu$CoMBo achieves a slightly bigger error. Once again, AdaBoost achieves the worst score. Even though late CoMBo and (simplified) $\mu$CoMBo may seem as two different algorithms and/or approaches, one may notice that late fusion CoMBo is an even more simplified version of $\mu$CoMBo: the cooperation coefficients for all the views and all the classes are put to 1. This explains the same performances in the *gold* setting and the comparable ones in the *ASR* one.

Finally, Table 4.7 gives the performances of the six methods for MAUC, G-mean and the norm of the confusion matrix. The overall better performing method is the late fusion version of CoMBo, since it does better than the others in 5 cases out of 6. However, whatever the case, the difference with other methods is not significant. For the MAUC, MuMBo and the late version of AdaBoost achieve nearly the same results as late CoMBo, both for the gold and ASR setting. For the G-mean and matrix's nom, late CoMBo is rivaled by early CoMBo and $\mu$CoMBo in the gold setting, and by MuMBo in the ASR one. It is interesting to notice that the results of Table 4.7 confirm our intuition on the link between the G-mean and the norm of the confusion matrix in Section 3.4.4 (page 65). Indeed, a small G-mean implies a high norm for the confusion matrix and vice versa.

The results in this section suggest that the choice of the learning method should depend on the error measure. For instance, if we're interested only on (minimizing) the empirical error, then based on Table 4.5, the best choice would be MuMBo. On the other hand, if we are interested in other measure, then Table 4.7 suggests that other methods should be considered, such as the late fusion version of CoMBo. However, due to the limited number of samples, further studies are needed in order to assert these preliminary conclusions.

| Algorithm | test (gold) | | | test (ASR) | | |
|---|---|---|---|---|---|---|
| | MAUC | G-mean | $\|\mathbf{C}\|$ | MAUC | G-mean | $\|\mathbf{C}\|$ |
| MuMBo | **0.948** | 0.0 | 0.992 | 0.904 | 0.558 | 0.693 |
| CoMBo (early) | 0.931 | 0.704 | 0.584 | 0.824 | 0.263 | 1.1546 |
| CoMBo (late) | 0.940 | **0.747** | **0.493** | **0.910** | **0.596** | **0.589** |
| $\mu$CoMBo | 0.894 | 0.526 | 0.537 | 0.866 | 0.373 | 0.705 |
| AdaBoost (early) | 0.930 | 0.498 | 0.595 | 0.886 | 0.303 | 0.738 |
| AdaBoost (late) | 0.942 | 0.0 | 0.829 | 0.904 | 0.0 | 1.125 |

Table 4.7 – *MAUC, G-mean and norm of the confusion matrix for the various methods on DECODA. Results in boldface indicate when one algorithm is better than the others.*

## 4.6 A simplified version of $\mu$CoMBo

At each iteration, $\mu$CoMBo uses a two steps procedure: first the best classifiers are learnt for all the views and then the cooperation coefficients are learnt through an optimization process. While the first part is closely related to the weak learning algorithms used for each view, the second one can be quite time consuming since it depends on the number of classes *and* views. In order to bypass this drawback, we propose a two-steps process:

— for each coefficient, computing the value that minimizes the loss in equation 4.13,

— for each class, normalizing the coefficients.

The second step ensures that some sort of cooperation is maintained.

Once the classifiers are learnt at the first step of each iteration $t$, the examples can be divided in two groups, for each view: let $S_j^+$ denote the set of examples correctly classified by the weak classifier $h_{t,j}$ learnt on view $j$ at iteration $t$, and $S_j^-$ the set of examples misclassified by $h_{t,j}$. Continuing from where we left off in equation 4.13, we have [4]:

$$
\begin{aligned}
\|\mathbf{C}_{\mathcal{S}}\|_1 &\leq \sum_{j=1}^{v}\sum_{i=1}^{m}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp\Big(f_{t,j}(i,c)-f_{t,j}(i,y_i)\Big)+2vm(K-1)\\
&=\sum_{j=1}^{v}\sum_{i\in S_j^+}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp\Big(f_{t-1,j}(i,c)-f_{t-1,j}(i,y_i)-\alpha_{t,j}\beta_{t,j,y_i}\Big)\\
&\quad+\sum_{j=1}^{v}\sum_{i\in S_j^-}\sum_{c\neq y_i\neq h_{t,j}(i)}\frac{1}{m_{y_i}}\exp\Big(f_{t-1,j}(i,c)-f_{t-1,j}(i,y_i)\Big)\\
&\quad+\sum_{j=1}^{v}\sum_{i\in S_j^-}\frac{1}{m_{y_i}}\exp\Big(\alpha_{t,j}\beta_{t,j,c}+f_{t-1,j}(i,c)-f_{t-1,j}(i,y_i)\Big)\\
&=\sum_{j=1}^{v}\sum_{l=1}^{K}\sum_{i\in S_j^+}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp\Big(f_{t-1,j}(i,c)-f_{t-1,j}(i,l)-\alpha_{t,j}\beta_{t,j,l}\Big)\mathbb{I}[y_i=l]\\
&\quad+\sum_{j=1}^{v}\sum_{l=1}^{K}\sum_{i\in S_j^-}\sum_{c\neq y_i\neq h_{t,j}(i)}\frac{1}{m_{y_i}}\exp\Big(f_{t-1,j}(i,c)-f_{t-1,j}(i,y_i)\Big)\mathbb{I}[h_{t,j}(i)=l]\\
&\quad+\sum_{j=1}^{v}\sum_{l=1}^{K}\sum_{i\in S_j^-}\frac{1}{m_{y_i}}\exp\Big(\alpha_{t,j}\beta_{t,j,l}+f_{t-1,j}(i,l)-f_{t-1,j}(i,y_i)\Big)\mathbb{I}[h_{t,j}(i)=l]\\
&=\sum_{j=1}^{v}\sum_{l=1}^{K}\Big[e^{-\alpha_{t,j}\beta_{t,j,l}}\sum_{i\in S_j^+}\sum_{c\neq y_i}\frac{1}{m_{y_i}}\exp\Big(f_{t-1,j}(i,c)-f_{t-1,j}(i,l)\Big)\mathbb{I}[y_i=l]\\
&\quad+e^{\alpha_{t,j}\beta_{t,j,l}}\sum_{i\in S_j^-}\frac{1}{m_{y_i}}\exp\Big(+f_{t-1,j}(i,l)-f_{t-1,j}(i,y_i)\Big)\mathbb{I}[h_{t,j}(i)=l]\Big]\\
&\quad+\sum_{j=1}^{v}\sum_{l=1}^{K}\sum_{i\in S_j^-}\sum_{c\neq y_i\neq h_{t,j}(i)}\frac{1}{m_{y_i}}\exp\Big(f_{t-1,j}(i,c)-f_{t-1,j}(i,y_i)\Big)\mathbb{I}[h_{t,j}(i)=l]. \quad (4.18)
\end{aligned}
$$

4. For convenience sake, the last term will be only recall in the first equation and omitted in the others, since it's a constant.

Let us define the following quantities before continuing:

$$A = \sum_{i \in S_j^+} \sum_{c \neq y_i} \frac{1}{m_{y_i}} \exp\Big(f_{t-1,j}(i,c) - f_{t-1,j}(i,l)\Big) \mathbb{I}[y_i = l],$$

$$B = \sum_{i \in S_j^-} \frac{1}{m_{y_i}} \exp\Big(f_{t-1,j}(i,l) - f_{t-1,j}(i,y_i)\Big) \mathbb{I}[h_{t,j}(i) = l],$$

$$C = \sum_{j=1}^{v} \sum_{l=1}^{K} \sum_{i \in S_j^-} \sum_{c \neq y_i \neq h_{t,j}(i)} \frac{1}{m_{y_i}} \exp\Big(f_{t-1,j}(i,c) - f_{t-1,j}(i,y_i)\Big) \mathbb{I}[h_{t,j}(i) = l].$$

Equation 4.18 can be rewritten as:

$$\|\mathbf{C}_{\mathcal{S}}\|_1 \leq \sum_{j=1}^{v} \sum_{l=1}^{K} \left[e^{-\alpha_{t,j}\beta_{t,j,l}} A + e^{\alpha_{t,j}\beta_{t,j,l}} B\right] + C. \tag{4.19}$$

Since the coefficients $\beta_{t,j,l}$ are independent one from the other, then, for a given view $j$ and a class $l$, equation 4.19 suggests that the value of the coefficient $\beta_{t,j,l}$ should computed as the solution of the following equation (which is simply the partial derivative of the right-handed side of equation 4.19 w.r.t. $\beta_{t,j,l}$):

$$-\alpha_{t,j} \exp(-\alpha_{t,j}\beta_{t,j,l}) A + \alpha_{t,j} \exp(\alpha_{t,j}\beta_{t,j,l}) B = 0. \tag{4.20}$$

Thus, we have:

$$\begin{aligned}
\alpha_{t,j} \exp(-\alpha_{t,j}\beta_{t,j,l}) A &= \alpha_{t,j} \exp(\alpha_{t,j}\beta_{t,j,l}) B \\
\exp(-\alpha_{t,j}\beta_{t,j,l}) A &= \exp(\alpha_{t,j}\beta_{t,j,l}) B \\
-\alpha_{t,j}\beta_{t,j,l} + \ln A &= \alpha_{t,j}\beta_{t,j,l} + \ln B \\
2\alpha_{t,j}\beta_{t,j,l} &= \ln A - \ln B \\
\beta_{t,j,l} &= \frac{1}{2\alpha_{t,j}} \ln \frac{A}{B}
\end{aligned} \tag{4.21}$$

Equation 4.21 implies that the value of $\beta_{t,j,l}$ depends on the ratio between the examples of class $l$ correctly classified by the classifier $h_{t,j}$ and the misclassified ones. This rate may be smaller than 1, giving thus a negative value for $\beta_{t,j,l}$. In order to avoid this case, we propose to chose $\beta_{t,j,l}$ as follows:

$$\beta_{t,j,l} = \max\{0, \frac{1}{2\alpha_{t,j}} \ln \frac{A}{B}\}$$

The last step consists in normalizing the values for these coefficients:

$$\beta_{t,j,l} \leftarrow \frac{\beta_{t,j,l}}{\sum_{j=1}^{v} \beta_{t,j,l}}$$

As such, these new coefficients correspond to some sort of confidence associated to the classifiers: the closer they are to zero, the less the classifier recognizes the class. This allows to keep a bare minimum of communication and/or cooperation between the views.

## 4.7 Where MuMBo meets $\mu$CoMBo

In section 2.6, we proposed to extend MuMBo by generalizing both the choice of the cooperation coefficients and the choice of the unique classifier. For the coefficients we proposed to choose real values in $[0, 1]$, while for the unique classifier an interesting choice was to consider a linear combination of all the per view classifiers. The reader might notice that both choices are similar to those used for $\mu$CoMBo, albeit the cooperation coefficients don't play the same role. In this section, we take a closer look to how these choices impact the loss computed for MuMBo.

Recall that the loss of MuMBo at iteration $t$ is given by the following expression (see Theorem 1, page 35):

$$L_t = \sum_{i=1}^{m} \sum_{l \neq y_i} \exp\Big(f_t(i,l) - f_t(i,y_i)\Big),$$

and the unique classifier at each iteration was computed as :

$$h_t(x) = \underset{l \in Y}{\operatorname{argmax}} \sum_{j=1}^{v} \mathbb{I}[h_{t,j}(x) = l]\beta_{t,j}. \tag{4.22}$$

The definition of the unique classifier in equation 4.22 suggests that if an example's class is predicted as $l$, then at least one of the weak classifiers does the same. Hence, we have:

$$\begin{aligned}
L_t &= \sum_{i=1}^{m} \sum_{l \neq y_i} \exp\Big(f_t(i,l) - f_t(i,y_i)\Big) \\
&= \sum_{i=1}^{m} \sum_{l \neq y_i} \exp\Big(\sum_{p=1}^{t} \alpha_p \mathbb{I}[h_p(i) = l] - \sum_{p=1}^{t} \alpha_p \mathbb{I}[h_p(i) = y_i]\Big) \\
&\leq \sum_{i=1}^{m} \sum_{l \neq y_i} \sum_{j=1}^{v} \exp\Big(\sum_{p=1}^{t} \alpha_p \mathbb{I}[h_p(i) = l] - \sum_{p=1}^{t} \alpha_p \mathbb{I}[h_p(i) = y_i]\Big) \\
&\leq \sum_{i=1}^{m} \sum_{l \neq y_i} \sum_{j=1}^{v} \exp\Big(\sum_{p=1}^{t} (\alpha_p + \alpha_{p,j} d_{p,j}(i) + \Delta_{p,j}(i))\Big),
\end{aligned} \tag{4.23}$$

where $\Delta_{p,j}(i) = \alpha_{p,j} d_{p,j}(i) \mathbb{I}[h_p(i) = l] - \alpha_{p,j} d_{p,j}(i) \mathbb{I}[h_p(i) = y_i]$.

In boosting procedures, the classifier obtained at each iteration needs only to perform better than random. That is, we can suppose that, without loss of generality, the coefficients $\alpha_p$ and $\alpha_{p,j}$ are at most 1. Also, in section 2.6, we proposed to choose the cooperation coefficients $d_{p,j}(i)$ in $[0, 1]$. Injecting these in equation 4.23, we have :

$$\begin{aligned}
L_t &\leq \sum_{i=1}^{m} \sum_{l \neq y_i} \sum_{j=1}^{v} \exp\Big(\sum_{p=1}^{t} (\alpha_p + \alpha_{p,j} d_{p,j}(i) + \Delta_{p,j}(i))\Big) \\
&= \sum_{i=1}^{m} \sum_{l \neq y_i} \sum_{j=1}^{v} \exp\Big(\sum_{p=1}^{t} (\alpha_p + \alpha_{p,j} d_{p,j}(i))\Big) \exp\Big(\sum_{p=1}^{t} \Delta_{p,j}(i)\Big) \\
&\leq \sum_{i=1}^{m} \sum_{l \neq y_i} \sum_{j=1}^{v} \exp(2T) \exp\Big(f_{t,j}(i,l) - f_{t,j}(i,y_i)\Big) \\
&\leq \exp(2T)\Big[\sum_{i=1}^{m} \sum_{l \neq y_i} \sum_{j=1}^{v} \exp\Big(f_{t,j}(i,l) - f_{t,j}(i,y_i)\Big)\Big].
\end{aligned} \tag{4.24}$$

Equation 4.24 suggests that minimizing the loss for each view, minimizes at the same time the total loss depending on the unique classifiers, since $\exp(2T)$ is a constant. Interestingly, this was also the case for $\mu$CoMBo, which implies that for both algorithms a similar process takes place: the loss is minimized by first choosing classifiers that verify the weak learning condition 1.5, and then selecting appropriate cooperation coefficients. This is further confirmed by the fact that the optimization criterions depending on the classifiers learnt on the views (given in equations 4.13 and 4.24) are similar for both algorithms:

$$\text{MuMBo: } \sum_{j=1}^{v}\sum_{i=1}^{m}\sum_{l\neq y_i} \exp\Big(f_{t,j}(i,l) - f_{t,j}(i,y_i)\Big),$$
$$\mu\text{CoMBo: } \sum_{j=1}^{v}\sum_{i=1}^{m}\sum_{l\neq y_i} \frac{1}{m_{y_i}} \exp\Big(f_{t,j}(i,l) - f_{t,j}(i,y_i)\Big).$$

Even though the motivations for the two methods were different, the resulting optimization problems are similar. This brings us to one of the most important results in this thesis:

> *In boosting based methods, modeling the cooperation between the various views by adding cooperation coefficients, results in a two step learning process for each iteration, first learning weak classifiers for all the views and then computing the values for the coefficients.*

## 4.8 Conclusion for this chapter

### 4.8.1 Discussion

As discussed in section 3.6, a key point of the framework presented in Chapter 3 and Chapter 4 is the construction of the confusion matrix and of the norm. Concerning the former case, in definition 9 the confusion matrix was based on the simple identity function, that is, the 0-1 loss. The downside of this choice is that it does not take advantage of the particular form of the multi-view classifier defined in equation 4.1. Indeed, the indicator function is only applied to the output of the final classifier, independently from the predictions of the classifiers learnt of the views. This opens up several research problems and future prospects related to the choice *and* construction of the confusion matrix. For instance, it would be interesting to consider replacing each entry of the confusion matrix by a loss function depending on the two classes, similar to the ones used in equation 3.7. The loss functions could then embed prior informations on how each of the classes are recognized by the various views.

The choice of the norm for the confusion is also an interesting topic for further works. Both in Chapter 3 and Chapter 4, we gave only one possible choice for the norm of the confusion matrix and the methods that result from it. The norm of the confusion matrix defines the optimization problem, the bounds that can be computed, and as a consequence the methods that can be derived. Future works on matrices norms could follow:

1. finding tighter bounds for the various norms and/or defining norms that can be computed analytically for a given matrix (in the spirit of the works by Ralaivola (2012)),
2. establishing a relation between the choice of the confusion matrix, coupled with its norm, and the cooperation between the views.

A common shortcoming for the methods presented in this chapter, is that the cooperation coefficients should be computed as a solution of an optimization problem. Although it might not be a problem for small datasets, in the prospect of ever growing available data, it can become quite a limitation to the use of $\mu$CoMBo (or even Carako) on practical examples. Further research will be focused on developing methods, or pinpointing cases, where the coefficients can be computed effectively (for instance, as in the simplified version of $\mu$CoMBo in section 4.6).

The methods presented in this chapter are mainly based on the supervised setting, and as pointed in section 1.3, the notion of cooperation between the views is also encountered in other frameworks, such as semi-supervised learning (Blum and Mitchell (1998)), active learning (Muslea and Knoblock (2006)), and so on. It would be intriguing to export the confusion matrices based framework on these settings, which could lead to new methods, or alternative explanation for existing ones.

Last but not least, in recent years, several works on the confusion matrix as an error measure have been proposed. In Morvant et al. (2012) we propose a bound on the norm of the confusion matrix for the Gibbs classifier (described in Section 3.3, page 62) in a PAC-Bayesian setting. Extending the definition of the Gibbs classifier to the multi-view setting would allow to obtain similar results for the PAC-Bayesian setting; and generalizing these results for the majority vote classifier would give way to developing novel multi-view methods based on the PAC-Bayes setting.

### 4.8.2 Conclusion

The main contribution in this chapter was the extension of the multi-class imbalance framework presented in Chapter 3 to the multi-view case, while embedding the notion of cooperation between the views introduced in Chapter 2. The latter was mainly introduced in the construction of the multi-view classifier in equation 4.1, which consisted of a weighted combination of the various classifiers learnt on the views. The former allowed us to derive several multi-view methods by applying the same procedure as in Chapter 3: first we defined the confusion matrix, then we used its norm in an optimization problem. The first methods based on this setting (given in Algorithm 5 and Algorithm 6) are ensemble methods whose goal is to find the optimal combinations for the various classifiers. Their strength resides in the fact that they can deal with classifiers outputted from different supervised methods. The last method proposed in this chapter, called $\mu$CoMBo and described in Algorithm 7, is a particular case of the other methods were the classifiers are learnt through a boosting procedure and it can be seen as a multi-view version of CoMBo (Algorithm 4). In section 4.8.1, we provided some leads on potential future research that can be based on the works presented in this chapter. Finally, as a secondarily contribution for the chapter, but a crucial one for this thesis, section 4.7 established a link between MuMBo and $\mu$CoMBo, resulting in the observation that forcing the cooperation between the views in a multi-view boosting process results in manipulating cooperation coefficients during the training phase.

# Conclusion

## There and back again

The aim of the works presented in this thesis is to develop and study methods that make use of uneven views. These works were mainly motivated by the simple observation that even though they describe the same data, some views contain information related to only one part of the instance space. In this concluding chapter we briefly summarize the main contributions presented in this thesis, which allows us to re-examine the discussion lead in the introductory chapter and, at the same time, outline the solutions proposed to address the questions posed therein.

Our first contribution in Chapter 1, is to formalize the concept of *uneven views* and for that purpose, we defined three notions on uneven views: strong views, weak views and imbalanced views. These notions are linked to the concepts of strong and weak learnability for the PAC setting, and the imbalanced classes problem. Compared to other view definitions in the literature, such as the notion of *homogeneity* employed by Janodet et al. (2009), the proposed notions are mainly based on the performances of the classifiers, similarly to the notion of *sufficiency* employed by Blum and Mitchell (1998). Building upon these notions, the methods proposed in this thesis are divided into two groups: methods that rely on strong/weak views and methods based on imbalanced views.

In Chapter 2, we introduce MuMBo (algorithm 3) a multi-view boosting method designed to make use of both strong and weak views. The key idea behind MuMBo is to install some sort of cooperation between the views so that each example could be processed by the most appropriate view. Compared to the literature, where multi-view boosting methods rely on one distribution, our approach maintains as many distributions as there are views, so that each view keeps its specificities. As the purpose of our approach was to have a general notion of cooperation, we show in Section 2.6 that the values of the cooperation coefficients of MuMBo could be either binary or continuous, thus defining MuMBo not as a simple algorithm, but as a whole family of algorithms. Although it was proved to be theoretically sound, the general version of MuMBo opened up several research problems related to the choice of the cooperation coefficients.

Before tackling the imbalanced views problem, we made a not so confusing stop. Chapter 3 delves on a well-known problem in the machine learning community: the imbalanced classes learning. Despite the number of methods that have been developed for this problem, to the best of our knowledge, the common goal for these methods is still unclear. Our main contribution for Chapter 3 is to propose a framework which gives a common base for all such methods. The starting point for the framework is the confusion matrix, or rather, the norm of the confusion matrix, and the principal result is an upper bound of the said norm. A supplementary contribution for this chapter is also CoMBo (algorithm 4), a boosting algorithm designed to actively minimize the norm of the confusion matrix. Among others, this work gives

a whole new perspective on a well-known problem, thus giving rise to several research problems, related both to the choice of the confusion matrix and its associated norm. An important question raised in Chapter 3 concernes the link between these two quantities. Indeed, finding a link between the choice of the confusion matrix and the considered norm would pave the way for novel algorithms for dealing with imbalanced classes.

Armed with the theoretical framework in Chapter 3, Chapter 4 applies the cooperation idea used in Chapter 2, to the imbalanced multi-view learning framework; that is, the views considered in this setting fall in the third category of the views defined in Chapter 1. The main contribution in this chapter consists in a series of algorithms based on the cooperation between the views. Contrary to MuMBo, the methods of Chapter 4 use the cooperation in order to find for each class the most appropriate view(s). In order to apply the framework of Chapter 3, we define the multi-view classifier as a general case of the weighted majority vote. The derivation procedure for the algorithms followed the same path as for CoMBo: bounding the norm of the confusion matrix for the multi-view classifier. The most interesting result for Chapter 4 is the multi-view boosting algorithm, called $\mu$CoMBo (algorithm 7), which eventually is similar to MuMBo (aside from the fact that the cooperation coefficients were defined for the classes rather than for the examples). Research problems raised by the works for that chapter are similar to the ones for Chapter 3, concerning the confusion matrix and its norm. In particular, defining a confusion matrix and a matrix norm that take advantage of the multi-view nature of the data would lay the foundations for novel multi-view methods.

Taken together, the methods presented in Chapter 2 and Chapter 4 provide a full set of tools that can deal with uneven views: MuMBo deals with weak and strong views, while $\mu$CoMBo is more adapted to imbalanced ones. One of the more significant results to emerge from this study is that promoting the cooperation between the views (either in the input space for the strong/weak views, or in the output space for the imbalanced ones) in a boosting process comes down to learning some cooperation coefficients at each iteration of the learning process. In general, therefore, it seems that *promoting the cooperation between the views* is equivalent to a two-steps procedure for each iteration: first the weak classifiers are learnt and then the cooperation coefficient are computed based on some optimization criterion. While this results is not new in itself for it has been used in different procedures, the fact of using it as a mean to simulate and promote the cooperation between the views is, in our opinion, a novelty of the works presented in this thesis. It is also interesting to notice that both methods, MuMBo and $\mu$CoMBo are applicable even in the case when, for certain examples, some of the views are missing. Indeed, it suffices to zero the weight of those examples in the distributions corresponding to the missing views.

Finally, throughout this thesis, the speech category classification problem is used as a testing ground for all the proposed methods, especially the multi-view ones. For that purpose, five views of various strengths were defined for representing a dialog between two humans.The empirical results show that the (multi-view) methods proposed in this thesis performed better than classical ones, especially in the noisy dataset. Research problems raised by the empirical results are both theoretical and empirical, such as testing the methods on larger datasets in order to prove the resistance to attribute noise of the methods, and establish a link between multi-view with cooperation and robustness against noise.

## Valinor ahoy

The works proposed in this thesis have addressed the principal question posed in the introduction, that of being able to learn performing classifiers when the considered views contain different information. However the methods proposed here are only one little step towards the active use of the cooperation between the views during the training procedure. As such, they open up research problems related to the theoretical aspect of multi-view learning and the practical use of these methods. Some of the possible directions that can be explored are presented in the conclusions of each chapter. Here we build on those conclusions and discuss more general prospects.

One of the main contributions of this thesis is to show that adding the cooperation in a boosting process comes down to adding cooperation coefficients in the cost functions. A limitation of the theoretical properties proved for both MuMBo and $\mu$CoMBo, is that the bounds do not depend on these coefficients, that is, the role of the cooperation coefficients is marginalized. Further research should be focused on the study of the effects that these coefficients have on the theoretical properties and how they influence the training procedure and the generalization properties of the methods. This research should also include *how* the coefficients are to be computed, since obtaining them as a solution to an optimization problem can be quite consuming on resources. For instance, it would be interesting to find a link between the strength of a view (that is, strong view, weak view or imbalanced view) and the analytical expression of the coefficients.

Nowadays, due to the great number of available data sources, the problem of learning from big data dictates the new trends for machine learning. Social media, such as Twitter, Facebook, Youtube, to name a few, offer an interesting setting for developing multi-view methods. Indeed, the users' profiles on these sites are made of images, videos, texts/messages, personal infos, and so on, each defining one or multiple views. Although MuMBo and $\mu$CoMBo do not make any assumption on the maximum number of views they can deal with, they might be time consuming when dealing with big data and big views (much like other classical methods in machine learning). It would be thus interesting to study and/or develop methods dealing with big data while at the same time using the cooperation between the views.

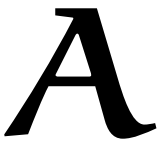
Dealing with multiple views can be quite tricky, especially when the number of view is quite high. Some of the views might contain redundant information, others might be way too noisy to be usable and so on and so forth. In Section 1.5, we show that the views defined for the DECODA project are of different strengths, but each contains useful information. The views were chosen so that each could bring some information that the others couldn't. Further research and experimentations in this direction (that of the choice of the views) would be of great help in understanding the role that each view has in the learning process and how to select the most appropriate views. These researches could be based on the recent works proposed by Kadri et al. (2013), where we used the covariance kernel as a similarity measure between two views.

When deriving the optimization problems for CoMBo and $\mu$CoMBo, we used an upper-bound on the norm of the confusion matrix. As discussed in the concluding sections for Chapters 3 and 4, further research should be focused on finding better definitions for the confusion matrix and the norm, much like in the case of the works proposed by Ralaivola (2012), where the norm of the confusion matrix can be directly computed, instead of using a bound. Defining such matrices and norms could also

be a solution for the big data problem, since the optimization problem might come in the form of view and/or data selection.

Throughout this thesis we show that multi-view methods usually come in the form of ensemble methods, where the resulting hypothesis is a combination of multiple classifiers. Kuncheva and Whitaker (2003) advocate that the performances of ensemble learning methods are linked to the diversity of the classifiers: the classifiers need to be diverse in order to perform better. In Chapter 2 we argue that the performances of MuMBo might imply that indirectly MuMBo encourages the diversity of the final ensemble of classifiers. It would be interesting to prove a formal relation between the multi-view methods presented in this thesis and the diversity of the final ensemble of classifiers. For doing that, it might be necessary to extend the notion of diversity for the multi-view setting; notion that can be based of the previously mentioned similarity between views. Defining such a notion would open the path to methods that select the classifiers (and views) that maximize the diversity of the ensemble, thus hopefully improving the computational time and resources needed for these methods.

# Appendix





In Section 3.3, we briefly introduce the results obtained on the norm of the confusion matrix in the PAC-Bayesian framework. This chapter presents the integral work we proposed in Morvant et al. (2012). Since we tackle a new framework, the notations use in this chapter differ from the ones in the other chapters, and are introduced in Section A.2.

## A.1 Introduction

The PAC-Bayesian framework, first introduced in McAllester (1999a), is an important field of research in learning theory. It borrows ideas from the philosophy of Bayesian inference and mix them with techniques used in statistical approaches of learning. Given a family of classifiers $\mathcal{F}$, the ingredients of a PAC-Bayesian bound are a *prior distribution* $\mathfrak{P}$ over $\mathcal{F}$, a learning sample $S$ and a *posterior distribution* $\mathfrak{Q}$ over $\mathcal{F}$. Distribution $\mathfrak{P}$ conveys some prior belief on what are the best classifiers from $\mathcal{F}$ (prior any access to $S$); the classifiers expected to be the most performant for the classification task at hand therefore have the largest weights under $\mathfrak{P}$. The posterior distribution $\mathfrak{Q}$ is learned/adjusted using the information provided by the training set $S$. The essence of PAC-Bayesian results is to bound the risk of the *stochastic* Gibbs classifier associated with $\mathfrak{Q}$ Catoni (2004) —in order to predict the label of an example $\mathbf{x}$, this predictor first draws a classifier $f$ from $\mathcal{F}$ according to $\mathfrak{Q}$ and then returns $f(\mathbf{x})$.

When specialized to appropriate function space $\mathcal{F}$ and relevant families of prior and posterior distributions, PAC-Bayes bounds can be used to characterize the error of different existing classification methods. An example deals with the risk of methods based upon the idea of the majority vote. We may notice that if $\mathfrak{Q}$ is the posterior distribution, the error of the $\mathfrak{Q}$-weighted majority vote classifier (which makes a prediction for $\mathbf{x}$ according to $\sum_f f(\mathbf{x})\mathfrak{Q}(f)$) is bounded by twice the error of the Gibbs classifier. If the classifiers from $\mathcal{F}$ the $\mathfrak{Q}$ puts a lot of weight on are good enough, the bound on the risk of the Gibbs classifier can therefore be an informative bound for the $\mathfrak{Q}$-weighted majority vote. Langford and Shawe-Taylor (2002) give a PAC-Bayes bound for Support Vector Machine (SVM), which depends on the margin of the examples. In their study, both the prior and posterior distribution are normal distributions, with different means and variances. Empirical results show that this bound is a good estimator of the risk of SVMs Langford (2005).

PAC-Bayes bounds can also be used to derive new supervised learning algorithms. For example, Lacasse et al. (2007) have introduced an elegant bound on the risk of the majority vote, which holds for any space $\mathcal{F}$. This bound is used to derive an algorithm, namely MinCq Laviolette et al. (2011), which achieves empirical results on par with state-of-the-art methods. Some other important results are given in Catoni (2007), Seegerr (2002), McAllester (1999b) and Langford et al. (2001).

In this paper, we address the multi-class classification problem. Some related works are therefore multi-class formulations for the SVMs, such as the frameworks of Weston and Watkins (1998), Lee et al. (2004) and Crammer and Singer (2002). As majority vote methods, we can also cite multi-class adaptations of the boosting method called AdaBoost Freund and Schapire (1996), such as the framework given in Mukherjee and Schapire (2011), the AdaBoost.MH/AdaBoost.MR algorithms Schapire and Singer (1999) and the SAMME algorithm Zhu et al. (2009).

The originality of our work is that we consider the *confusion matrix* of the Gibbs classifier as an error measure. We believe that in the multi-class framework, it is more relevant to consider the confusion matrix as the error measure than the mere misclassification error, which corresponds to the probability for some classifier $h$ to err for its prediction on $\mathbf{x}$. The information as to what is the probability for an instance of class $p$ to be classified into class $q$ (with $p \neq q$) by some predictor is indeed crucial in some applications (think of the difference between false-negative and false-positive predictions in a diagnosis automated system). To the best of our knowledge, we are the first to propose a generalization bound on the confusion matrix in the

PAC-Bayesian framework. The result that we propose heavily relies on a matrix concentration inequality for sums of random matrices introduced by Tropp (2011). One may anticipate that generalization bounds for the confusion matrix may also be obtained in other framework than the PAC-Bayesian framework (e.g. uniform stability, online learning).

The rest of this paper is organized as follows. Sec. A.2 introduces the setting of multi-class learning and some of the basic notation used throughout the paper. Sec. A.3 briefly recalls the folk PAC-Bayes bound as introduced in McAllester (2003). In Sec. A.4, we present the main contribution of this paper, our PAC-Bayes bound on the confusion matrix, followed by its proof in Sec. A.5. We discuss some future works in Sec. A.6.

## A.2 Setting and Notations

This section presents the general setting that we consider and the different tools that we will make use of.

### A.2.1 General Problem Setting

We consider classification tasks over the *input space* $X \subseteq \mathbb{R}^d$ of dimension $d$. The *output space* is denoted by $Y = \{1, \ldots, Q\}$, where $Q$ is the number of classes. The learning sample is denoted by $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where each example is drawn *i.i.d.* from a fixed —but unknown— probability distribution $\mathfrak{D}$ defined over $X \times Y$. $\mathfrak{D}_m$ denotes the distribution of a $m$-sample. $\mathcal{F} \subseteq \mathbb{R}^X$ is a family of classifiers $f : X \to Y$. $\mathfrak{P}$ and $\mathfrak{Q}$ are respectively the *prior* and the *posterior* distributions over $\mathcal{F}$. Given the prior distribution $\mathfrak{P}$ and the training set $S$, the learning process consists in finding the posterior distribution $\mathfrak{Q}$ leading to a good generalization.

Since we make use of the prior distribution $\mathfrak{P}$ on $\mathcal{F}$, a PAC-Bayes generalization bound depends on the Kullback-Leibler divergence (KL-divergence):

$$KL(\mathfrak{Q}\|\mathfrak{P}) = \mathbb{E}_{f\sim\mathfrak{Q}} \log \frac{\mathfrak{Q}(f)}{\mathfrak{P}(f)}. \tag{A.1}$$

The function $\text{sign}(x)$ is equal to $+1$ if $x \geq 0$ and $-1$ otherwise. The indicator function $\mathbb{I}(x)$ is equal to 1 if $x$ is true and 0 otherwise.

### A.2.2 Conventions and Basics on Matrices

Throughout the paper we consider only real-valued square matrices $\mathbf{C}$ of order $Q$ (the number of classes). ${}^t\mathbf{C}$ is the transpose of the matrix $\mathbf{C}$, $\mathbf{Id}_Q$ denotes the identity matrix of size $Q$ and $\mathbf{0}$ is the zero matrix.

The results given in this paper are based on a concentration inequality of Tropp (2011) for a sum of random self-adjoint matrices. In the case when a matrix is not self-adjoint and is real-valued, we use the dilation of such a matrix, given in Paulsen (2002), which is defined as follows:

$$\mathcal{S}(\mathbf{C}) \stackrel{def}{=} \begin{pmatrix} \mathbf{0} & \mathbf{C} \\ {}^t\mathbf{C} & \mathbf{0} \end{pmatrix}. \tag{A.2}$$

The symbol $\|\cdot\|$ corresponds to the *operator norm* also called the *spectral norm*: it returns the largest singular value of its argument, which is defined by

$$\|\mathbf{C}\| = \max\{\lambda_{\max}(\mathbf{C}), -\lambda_{\min}(\mathbf{C})\}, \tag{A.3}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are respectively the algebraic maximum and minimum singular value of $\mathbf{C}$. Note that the dilation preserves spectral information, so we have:

$$\lambda_{\max}\big(\mathcal{S}(\mathbf{C})\big) = \|\mathcal{S}(\mathbf{C})\| = \|\mathbf{C}\|. \tag{A.4}$$

Since $\|\cdot\|$ is a regular norm, the following equality obviously holds:

$$\forall a \in \mathbb{R},\ \|a\mathbf{C}\| = |a|\|\mathbf{C}\|. \tag{A.5}$$

Given the matrices $\mathbf{C}$ and $\mathbf{D}$ both made of nonnegative elements and such that $0 \leq \mathbf{C} \leq \mathbf{D}$ (element-wise), we have:

$$0 \leq \mathbf{C} \leq \mathbf{D} \Rightarrow \|\mathbf{C}\| \leq \|\mathbf{D}\|. \tag{A.6}$$

### A.2.3 Tools used in the proofs

The work presented in this paper relies on the results given in Theorems 8, 9 and 10.

**Theorem 8** (Concentration Inequality for Random Matrices Tropp (2011)) *Consider a finite sequence $\{\mathbf{M}_i\}$ of independant, random, self-adjoint matrices with dimension $Q$, and let $\{\mathbf{A}_i\}$ be a sequence of fixed self-adjoint matrices. Assume that each random matrix satisfies $\mathbb{E}\mathbf{M}_i = \mathbf{0}$ and $\mathbf{M}_i^2 \preccurlyeq \mathbf{A}_i^2$ almost surely. Then, for all $\epsilon \geq 0$,*

$$\mathbb{P}\left\{\lambda_{\max}\Big(\sum_i \mathbf{M}_i\Big) \geq \epsilon\right\} \leq Q.\exp\left(\frac{-\epsilon^2}{8\sigma^2}\right),$$

*where $\sigma^2 \stackrel{def}{=} \|\sum_i \mathbf{A}_i^2\|$ and $\preccurlyeq$ refers to the semidefinite order on self-adjoint matrices.*

**Theorem 9** (Markov's inequality) *Let $Z$ be a random variable and $z \geq 0$, then:*

$$\mathbb{P}\left(|Z| \geq z\right) \leq \frac{\mathbb{E}(|Z|)}{z}.$$

**Theorem 10** (Jensen's inequality) *Let $X$ be an integrable real-valued random variable and $g(\cdot)$ be a convex function, then:*

$$f(\mathbb{E}[Z]) \leq \mathbb{E}[g(Z)].$$

## A.3 The Usual PAC-Bayes Theorem

In this section, we recall the main PAC-Bayesian bound in binary classification case as presented in McAllester (2003), Seegerr (2002), Langford (2005). The set of labels we consider is $Y = \{-1, 1\}$ (with $Q = 2$) and, for each classifier $f \in \mathcal{F}$, the predicted label of $\mathbf{x} \in X$ is given by $\text{sign}(f(\mathbf{x}))$. The true risk $R(f)$ and the empirical error $R_S(f)$ of $f$ are defined as:

$$R(f) \stackrel{def}{=} \mathbb{E}_{(\mathbf{x},y)\sim\mathfrak{D}}\mathbb{I}(f(\mathbf{x} \neq y)),$$
$$R_S(f) \stackrel{def}{=} \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(f(\mathbf{x}_i \neq y_i)).$$

The learner's aim is to choose a posterior distribution $\mathfrak{Q}$ on $\mathcal{F}$ such that the risk of the $\mathfrak{Q}$-weighted majority vote (also called the Bayes classifier) $B_{\mathfrak{Q}}$ is as small as possible. $B_{\mathfrak{Q}}$ is defined by:

$$B_{\mathfrak{Q}}(\mathbf{x}) = \text{sign}\left[\mathbb{E}_{f\sim\mathfrak{Q}} f(\mathbf{x})\right].$$

The true risk $R(B_{\mathfrak{Q}})$ and the empirical error $R_S(B_{\mathfrak{Q}})$ of the Bayes classifier are defined as the probability that it commits an error on an example:

$$R(B_{\mathfrak{Q}}) \stackrel{def}{=} \mathbb{P}_{(\mathbf{x},y)\sim\mathfrak{D}}\left(B_{\mathfrak{Q}}(\mathbf{x}) \neq y\right). \tag{A.7}$$

However, the PAC-Bayes approach does not directly bound the risk of $B_{\mathfrak{Q}}$. Instead, it bounds the risk of the stochastic Gibbs classifier $G_{\mathfrak{Q}}$ which predicts the label of $\mathbf{x} \in X$ by first drawing $f$ according to $\mathfrak{Q}$ and then returning $f(\mathbf{x})$. The true risk $R(G_{\mathfrak{Q}})$ and the empirical error $R_S(G_{\mathfrak{Q}})$ of $G_{\mathfrak{Q}}$ are therefore:

$$R(G_{\mathfrak{Q}}) = \mathbb{E}_{f\sim\mathfrak{Q}} R(f)\ ;\ R_S(G_{\mathfrak{Q}}) = \mathbb{E}_{f\sim\mathfrak{Q}} R_S(f). \tag{A.8}$$

Note that in this setting, we have $R(B_{\mathfrak{Q}}) \leq 2R(G_{\mathfrak{Q}})$.

We present the PAC-Bayes theorem which gives a bound on the error of the stochastic Gibbs classifier.

**Theorem 11** *For any $\mathfrak{D}$, any $\mathcal{F}$, any $\mathfrak{P}$ of support $\mathcal{F}$, any $\delta \in (0,1]$, we have,*

$$\mathbb{P}_{S\sim\mathfrak{D}_m}\Bigg(\forall\mathfrak{Q} \text{ on } \mathcal{F},\ kl\big(R_S(G_{\mathfrak{Q}}), R(G_{\mathfrak{Q}})\big) \leq \frac{1}{m}\left[KL(\mathfrak{Q}\|\mathfrak{P}) + \ln\frac{\xi(m)}{\delta}\right]\Bigg) \geq 1-\delta,$$

*where $kl(a,b) \stackrel{def}{=} a\ln\frac{a}{b} + (1-a)\ln\frac{1-a}{1-b}$,*
*and $\xi \stackrel{def}{=} \sum_{i=0}^{m}\binom{m}{i}(i/m)^i(1-i/m)^{m-i}$.*

We now provide a novel PAC-Bayes bound in the context of multi-class classification by considering the confusion matrix as an error measure.

## A.4 Multiclass PAC-Bayes Bound

### A.4.1 Definitions and Setting

As said earlier, we focus on multi-class classification. The output space is $Y = \{1,\ldots,Q\}$, with $Q > 2$. We only consider learning algorithms acting on learning sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where each example is drawn *i.i.d* according to $\mathfrak{D}$, such that $|S| \geq Q$ and $m_{y_j} \geq 1$ for every class $y_j \in Y$, where $m_{y_j}$ is the number of examples of real class $y_j$. In the context of multi-class classification, an error measure can be a performance tool called *confusion matrix*. We consider the classical definition of the confusion matrix based on conditional probabilities: it is inherent (and desirable) to minimize the effects of unbalanced classes. Concretely, for a given classifier $f \in \mathcal{F}$ and a sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \sim \mathfrak{D}_m$, the *empirical confusion matrix* $\mathbf{D}_S^f = (\hat{d}_{pq})_{1\leq p,q\leq Q}$ of $f$ is defined as follows:

$$\forall(p,q),\ \hat{d}_{pq} \stackrel{def}{=} \sum_{i=1}^{m}\frac{1}{m_{y_i}}\mathbb{I}(f(\mathbf{x}_i)=q)\mathbb{I}(y_i=p).$$

The *true confusion matrix* $\mathbf{D}^f = (d_{pq})_{1\leq p,q\leq Q}$ of $f$ over $\mathfrak{D}$ corresponds to:

$$\begin{aligned}\forall (p,q),\ d_{pq} &\stackrel{def}{=} \mathbb{E}_{\mathbf{x}|y=p}\mathbb{I}\big(f(\mathbf{x})=q\big)\\ &= \mathbb{P}_{(\mathbf{x},y)\sim\mathfrak{D}}(f(\mathbf{x})=q|y=p).\end{aligned}$$

If $f$ correctly classifies every example of the sample $S$, then all the elements of the confusion matrix are 0, except for the diagonal ones which correspond to the correctly classified examples. Hence the more there are non-zero elements in a confusion matrix outside the diagonal, the more the classifier is prone to err. Recall that in a learning process the objective is to learn a classifier $f \in \mathcal{F}$ with a low true error (*i.e.* with good generalization guarantees), we are thus only interested in the errors of $f$. Our objective is then to find $f$ leading to a confusion matrix with the more zero elements outside the diagonal. Since the diagonal gives the conditional probabilities of 'correct' predictions, we propose to consider a different kind of confusion matrix by discarding the diagonal values. Then the only non-zero elements of the new confusion matrix correspond to the examples that are misclassified by $f$. For all $f \in \mathcal{F}$ we define the empirical and true confusion matrices of $f$ by respectively $\mathbf{C}_S^f = (\hat{c}_{pq})_{1\leq p,q\leq Q}$ and $\mathbf{C}^f = (c_{pq})_{1\leq p,q\leq Q}$ such that for all $(p,q)$:

$$\hat{c}_{pq} \stackrel{def}{=} \begin{cases} 0 & \text{if } q = p \\ \hat{d}_{pq} & \text{otherwise,}\end{cases} \tag{A.9}$$

$$c_{pq} \stackrel{def}{=} \begin{cases} 0 & \text{if } q = p \\ d_{pq} = \mathbb{P}_{(\mathbf{x},y)\sim\mathfrak{D}}(f(\mathbf{x})=q|y=p) & \text{otherwise.}\end{cases} \tag{A.10}$$

Note that if $f$ correctly classifies every example of a given sample $S$, then the empirical confusion matrix $\mathbf{C}_S^f$ is equal to $\mathbf{0}$. Similarly, if $f$ is a perfect classifier over the distribution $\mathfrak{D}$, then the true confusion matrix is equal to $\mathbf{0}$. Therefore a relevant task is to minimize the size of the confusion matrix, thus having a confusion matrix as close to $\mathbf{0}$ as possible.

### A.4.2 Main Result: Confusion PAC-Bayes Bound for the Gibbs Classifier

Our main result is a PAC-Bayes generalization bound over the Gibbs classifier $G_{\mathfrak{Q}}$ in this particular context, where the empirical and true error measures are respectively given by the confusion matrices from (A.9) and (A.10). In this case, we can define the true and the empirical confusion matrices of $G_{\mathfrak{Q}}$ respectively by:

$$\mathbf{C}^{G_{\mathfrak{Q}}} = \mathbb{E}_{f\sim\mathfrak{Q}}\mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_S^f\ ;\ \mathbf{C}_S^{G_{\mathfrak{Q}}} = \mathbb{E}_{f\sim\mathfrak{Q}}\mathbf{C}_S^f.$$

Given $f \sim \mathfrak{Q}$ and a sample $S \sim \mathfrak{D}_m$, our objective is to bound the difference between $\mathbf{C}^{G_{\mathfrak{Q}}}$ and $\mathbf{C}_S^{G_{\mathfrak{Q}}}$, the true and empirical errors of the Gibbs classifier. Remark that the error rate $P(f(\mathbf{x}) \neq y)$ of a classifier $f$ might be directly computed as the 1-norm of ${}^t\mathbf{C}^f\mathbf{p}$, where $\mathbf{p}$ is the vector of prior probabilities. However, in our case, concentration inequalities are only available for the operator norm. Since we have $\|\mathbf{u}\|_1 \leq \sqrt{Q}\|\mathbf{u}\|_2$ for any $Q$-dimensional vector $\mathbf{u}$, we have that $P(f(\mathbf{x}) \neq y) \leq \sqrt{Q}\|\mathbf{C}^f\|_{op}$. Thus trying to minimize the operator norm of $\mathbf{C}^f$ is a relevant strategy to control the risk. Here is our main result, a bound on the operator norm of the difference between $\mathbf{C}^{G_{\mathfrak{Q}}}$ and $\mathbf{C}_S^{G_{\mathfrak{Q}}}$.

**Theorem 12** *Let $X \subseteq \mathbb{R}^d$ be the input space, $Y = \{1, \ldots, Q\}$ the output space, $\mathfrak{D}$ a distribution over $X \times Y$ (with $\mathfrak{D}_m$ the distribution of a $m$-sample) and $\mathcal{F}$ a family of classifiers from $X$ to $Y$. Then for every prior distribution $\mathfrak{P}$ over $\mathcal{F}$ and any $\delta \in (0, 1]$, we have:*

$$\mathbb{P}_{S\sim\mathfrak{D}_m}\Bigg\{\forall \mathfrak{Q} \text{ on } \mathcal{F}, \|\mathbf{C}_S^{G_{\mathfrak{Q}}} - \mathbf{C}^{G_{\mathfrak{Q}}}\| \leq \sqrt{\frac{8Q}{m_- - 8Q}\left[KL(\mathfrak{Q}||\mathfrak{P}) + \ln\left(\frac{m_-}{4\delta}\right)\right]}\Bigg\} \geq 1 - \delta,$$

*where $m_- = \min_{y=1,\ldots,Q} m_y$ is the minimal number of examples from $S$ which belong to the same class.*

*Proof.* Deferred to Section A.5. □

Note that, for all $y \in Y$, we need the following hypothesis: $m_y > 8$, which is not too strong a limitation.

Finally, we rewrite Theorem 12 in order to provide a bound on the size $\|\mathbf{C}^{G_{\mathfrak{Q}}}\|$.

**Corollary 3** *We consider the hypothesis of the Theorem 12. We have:*

$$\mathbb{P}_{S\sim\mathfrak{D}_m}\Bigg\{\forall \mathfrak{Q} \text{ on } \mathcal{F}, \|\mathbf{C}^{G_{\mathfrak{Q}}}\| \leq \|\mathbf{C}_S^{G_{\mathfrak{Q}}}\| + \sqrt{\frac{8Q}{m_- - 8Q}\left[KL(\mathfrak{Q}||\mathfrak{P}) + \ln\left(\frac{m_-}{4\delta}\right)\right]}\Bigg\} \geq 1 - \delta.$$

*Proof.* By application of the reverse triangle inequality $|\|\mathbf{A}\| - \|\mathbf{B}\|| \leq \|\mathbf{A} - \mathbf{B}\|$ to Theorem 12. □

Both Theorem 12 and Corollary 3 yield a bound on the estimation (through the operator norm) of the true confusion matrix of the Gibbs classifier over the posterior distribution $\mathfrak{Q}$, though this is more explicit in the corollary. Let the number of classes $Q$ be a constant, then the true risk is upper-bounded by the empirical risk of the Gibbs classifier and a term depending on the number of training examples, especially on the value $m_-$ which corresponds to the minimal quantity of examples that belong to the same class. This means that the larger $m_-$, the closer the empirical confusion matrix of the Gibbs classifier is to its true matrix.

### A.4.3 Upper Bound on the Risk of the Majority Vote Classifier

We recall that the Bayes classifier $B_{\mathfrak{Q}}$ is well known as majority vote classifier under a given posterior distribution $\mathfrak{Q}$. In the multiclass setting, $B_{\mathfrak{Q}}$ is such that for any example it returns the majority class under the measure $\mathfrak{Q}$ and we define it as:

$$B_{\mathfrak{Q}}(\mathbf{x}) = \text{argmax}_{c\in Y}\left[\mathbb{E}_{f\in\mathfrak{Q}}\mathbb{I}(f(\mathbf{x}) = c)\right]. \tag{A.11}$$

We define the conditional Gibbs risk $R(G_{\mathfrak{Q}}, p, q)$ and Bayes risk $R(G_{\mathfrak{Q}}, p, q)$ as

$$R(G_{\mathfrak{Q}}, p, q) = \mathbb{E}_{\mathbf{x}\sim D_{|y=p}}\mathbb{E}_{f\sim\mathfrak{Q}}\mathbb{I}(f(\mathbf{x}) = q), \tag{A.12}$$

$$R(B_{\mathfrak{Q}}, p, q) = \mathbb{E}_{\mathbf{x}\sim D_{|y=p}}\mathbb{I}\left(\text{argmax}_{c\in Y}\, g(c, q)\right). \tag{A.13}$$

where

$$g(c,q) = \left[\mathbb{E}_{f\in\mathfrak{Q}}\mathbb{I}(f(\mathbf{x}) = c) = q\right]$$

The former is the $(p,q)$ entry of $\mathbf{C}^{G_\mathfrak{Q}}$ (if $p \neq q$) and the latter is the $(p,q)$ entry of $\mathbf{C}^{B_\mathfrak{Q}}$.

**Proposition 3** *Let $Q \geq 2$ be the number of classes. Then $R(B_\mathfrak{Q}, p, q)$ and $R(G_\mathfrak{Q}, p, q)$ are related by the following inequality :*

$$\forall (q,p), R(B_\mathfrak{Q}, p, q) \leq QR(G_\mathfrak{Q}, p, q). \tag{A.14}$$

*Proof.* Consider a labeled pair $(\mathbf{x}, y)$. Let us introduce the notation $\gamma_q(\mathbf{x})$ for $q \in Y$ such that:

$$\gamma_q(\mathbf{x}) = \mathbb{E}_{f\sim\mathfrak{Q}}\mathbb{I}(f(\mathbf{x}) = q) = \sum_{f:f(\mathbf{x})=q} \mathfrak{Q}(q).$$

Obviously,

$$\sum_{q\in Y} \gamma_q(\mathbf{x}) = 1.$$

Recall that the conditional Gibbs risk $R(G_\mathfrak{Q}, p, q)$ and Bayes risk $R(G_\mathfrak{Q}, p, q)$ are defined as:

$$R(G_\mathfrak{Q}, p, q) = \mathbb{E}_{\mathbf{x}\sim D_{|y=p}}\mathbb{E}_{f\sim\mathfrak{Q}}\mathbb{I}(f(\mathbf{x}) = q) = \mathbb{E}_{\mathbf{x}\sim D_{|y=p}}\gamma_q(\mathbf{x}), \tag{A.12}$$

$$R(B_\mathfrak{Q}, p, q) = \mathbb{E}_{\mathbf{x}\sim D_{|y=p}}\mathbb{I}(\text{argmax}_{c\in Y}\, \gamma_c(\mathbf{x}) = q) \tag{A.13}$$

The former is the $(p,q)$ entry of $\mathbf{C}^{G_\mathfrak{Q}}$ (if $p \neq q$) and the latter is the $(p,q)$ entry of $\mathbf{C}^{B_\mathfrak{Q}}$.

For $q \neq y$ to be predicted by the majority vote classifier, it is necessary and sufficient that

$$\gamma_q(\mathbf{x}) \geq \gamma_c(\mathbf{x}),\ \forall c \in Y, c \neq q.$$

This might be equivalently rewritten as:

$$\mathbb{I}(\text{argmax}_c\, \gamma_c(\mathbf{x}) = q) = \mathbb{I}(\wedge_{c,c\neq q}\, \gamma_q(\mathbf{x}) \geq \gamma_c(\mathbf{x})) \tag{A.15}$$

(note that the expectation of the left-hand side which respect to $D_{|y=p}$ is $R(B_\mathfrak{Q}, p, q)$ —cf. (A.13)). Now remark that:

$$\begin{aligned}
\mathbb{I}(\wedge_{c,c\neq q}\, \gamma_q(\mathbf{x}) \geq \gamma_c(\mathbf{x})) = 1 &\Leftrightarrow \gamma_q(\mathbf{x}) - \gamma_c(\mathbf{x}),\ \forall c \in Y, c \neq q \\
&\Rightarrow \sum_{c\in Y, c\neq q} (\gamma_q(\mathbf{x}) - \gamma_c(\mathbf{x})) \geq 0 \\
&\Leftrightarrow \sum_{c\in Y, c\neq q} \gamma_q(\mathbf{x}) - \sum_{c\in Y, c\neq q} \gamma_c(\mathbf{x}) \geq 0 \\
&\Leftrightarrow (Q-1)\gamma_q(\mathbf{x}) - (1 - \gamma_q(\mathbf{x})) \geq 0 \\
&\Leftrightarrow \gamma_q(\mathbf{x}) \geq \frac{1}{Q}.
\end{aligned}$$

where we have used $\sum_{c\in Y} \gamma_c(\mathbf{x}) = 1$ in the next to last line. This means that:

$$\mathbb{I}(\wedge_{c,c\neq q}\, \gamma_q(\mathbf{x}) \geq \gamma_c(\mathbf{x})) = 1 \Rightarrow \mathbb{I}\left(\gamma_q(\mathbf{x}) \geq \frac{1}{Q}\right) = 1,$$

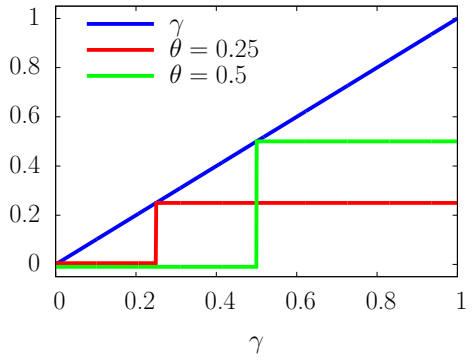



Figure A.1 – *Plot of* $\gamma \mapsto \theta \mathbb{I}(\gamma \geq \theta)$, *for* $\theta = 0.25$ *(red) and* $\theta = 0.5$ *(green). Observe that* $\gamma \geq \theta \mathbb{I}(\gamma \geq \theta)$, $\forall \theta \in [0,1]$.

from which we get:

$$\mathbb{I}(\wedge_{c,c\neq q}\, \gamma_q(\mathbf{x}) \geq \gamma_c(\mathbf{x})) \leq \mathbb{I}\left(\gamma_q(\mathbf{x}) \geq \frac{1}{Q}\right),$$

that is, by virtue of (A.15):

$$\mathbb{I}\left(\text{argmax}_c\, \gamma_c(\mathbf{x}) = q\right) \leq \mathbb{I}\left(\gamma_q(\mathbf{x}) \geq \frac{1}{Q}\right).$$

We then may use that $\gamma \geq \theta \mathbb{I}(\gamma \geq 1/Q), \forall \gamma \in [0,1], \theta \in [0,1]$, as illustrated on Figure A.1, to obtain

$$\frac{1}{Q}\mathbb{I}\left(\gamma_q(\mathbf{x}) \geq \frac{1}{Q}\right) \leq \gamma_q(\mathbf{x}) \Leftrightarrow \mathbb{I}\left(\gamma_q(\mathbf{x}) \geq \frac{1}{Q}\right) \leq Q\gamma_q(\mathbf{x}),$$

and, combining with the previous inequality:

$$\mathbb{I}\left(\text{argmax}_c\, \gamma_c(\mathbf{x}) = q\right) \leq Q\gamma_q(\mathbf{x}).$$

Taking the expectation of both sides with respect to $\mathbf{x} \sim D_{|y=p}$, we get:

$$R(B_{\mathfrak{Q}}, p, q) \leq QR(G_{\mathfrak{Q}}, p, q).$$

□

This proposition implies the following result on the confusion matrices associated to $B_{\mathfrak{Q}}$ and $G_{\mathfrak{Q}}$.

**Corollary 4** *Let* $Q \geq 2$ *be the number of class. Then* $\mathbf{C}^{B_{\mathfrak{Q}}}$ *and* $\mathbf{C}^{G_{\mathfrak{Q}}}$ *are related by the following inequality:*

$$\|\mathbf{C}^{B_{\mathfrak{Q}}}\| \leq Q\|\mathbf{C}^{G_{\mathfrak{Q}}}\|. \tag{A.16}$$

*Proof.* From the definitions of $R(G_{\mathfrak{Q}}, p, q)$ (A.12) and $R(B_{\mathfrak{Q}}, p, q)$ (A.13), we directly obtain from Proposition 3:

$$\mathbf{C}^{B_{\mathfrak{Q}}} \leq Q\mathbf{C}^{G_{\mathfrak{Q}}}. \tag{A.17}$$

We dilate $\mathbf{C}^{B_{\mathfrak{Q}}}$ and $Q\mathbf{C}^{G_{\mathfrak{Q}}}$, then (A.17) is rewritten as:

$$\mathcal{S}(\mathbf{C}^{B_{\mathfrak{Q}}}) \leq \mathcal{S}(Q\mathbf{C}^{G_{\mathfrak{Q}}}).$$

Since all component of a confusion matrix are positive, we have $0 \leq \mathcal{S}(\mathbf{C}^{B_{\mathfrak{Q}}}) \leq \mathcal{S}(Q\mathbf{C}^{G_{\mathfrak{Q}}})$. We can thus apply the property (A.6). We obtain:

$$\lambda_{\max}(\mathcal{S}(\mathbf{C}^{B_{\mathfrak{Q}}})) \leq \lambda_{\max}(\mathcal{S}(Q\mathbf{C}^{G_{\mathfrak{Q}}})). \tag{A.18}$$

Then, with property (A.4), (A.18) is rewritten as:

$$\|\mathbf{C}^{B_{\mathfrak{Q}}}\| \leq \|Q\mathbf{C}^{G_{\mathfrak{Q}}}\|.$$

Finally, by application of (A.5):

$$\|\mathbf{C}^{B_{\mathfrak{Q}}}\| \leq Q\|\mathbf{C}^{G_{\mathfrak{Q}}}\|.$$

□

## A.5 Proof of Theorem 12

This section gives the formal proof of Theorem 12. We first introduce a concentration inequality for a sum of random square matrices. This allows us to deduce the PAC-Bayes generalization bound for confusion matrices by following the same "three step process" as the one given in McAllester (2003), Seegerr (2002), Langford (2005) for the classic PAC-Bayesian bound.

### A.5.1 Concentration Inequality for the Confusion Matrix

The main result of our work is based on the following corollary of a result on the concentration inequality for a sum of self-adjoint matrices given by Tropp (2011) (see Theorem 8 in Appendix) – this theorem generalizes Hoeffding's inequality to the case self-adjoint random matrices. The purpose of the following corollary is to restate the Theorem 8 so that it carries over to matrices that are not self-adjoint. It is central to us to have such a result as the matrices we are dealing with, namely confusion matrices, are rarely symmetric.

**Corollary 5** *Consider a finite sequence $\{\mathbf{M}_i\}$ of independent, random, square matrices of order $Q$, and let $\{a_i\}$ be a sequence of fixed scalars. Assume that each random matrix satisfies $\mathbb{E}_i\mathbf{M}_i = \mathbf{0}$ and $\|\mathbf{M}_i\| \leq a_i$ almost surely.. Then, with $\sigma^2 \overset{def}{=} \sum_i a_i^2$, we have,*

$$\forall \epsilon \geq 0,\ \mathbb{P}\left\{\|\sum_i \mathbf{M}_i\| \geq \epsilon\right\} \leq 2.Q.\exp\left(\frac{-\epsilon^2}{8\sigma^2}\right). \tag{A.19}$$

*Proof.* We want to verify the hypothesis given in Theorem 8 in order to apply it.
Let $\{\mathbf{M}_i\}$ be a finite sequence of independent, random, square matrices of order $Q$ such that $\mathbb{E}_i\mathbf{M}_i = \mathbf{0}$ and let $\{a_i\}$ be a sequence of fixed scalars such that $\|\mathbf{M}_i\| \leq a_i$. We consider the sequence $\{\mathcal{S}(\mathbf{M}_i)\}$ of random self-adjoint matrices with dimension $2Q$. By the definition of the dilation, we obtain $\mathbb{E}_i\mathcal{S}(\mathbf{M}_i) = \mathbf{0}$.
From Equation (A.4), the dilation preserves the spectral information. Thus, on the one hand, we have:

$$\|\sum_i \mathbf{M}_i\| = \lambda_{\max}\left(\mathcal{S}\left(\sum_i \mathbf{M}_i\right)\right) = \lambda_{\max}\left(\sum_i \mathcal{S}(\mathbf{M}_i)\right).$$

On the other hand, we have:

$$\|\mathbf{M}_i\| = \|\mathcal{S}(\mathbf{M}_i)\| = \lambda_{\max}\big(\mathcal{S}(\mathbf{M}_i)\big) \leq a_i.$$

To assure the hypothesis $\mathcal{S}(\mathbf{M}_i)^2 \preccurlyeq \mathbf{A}_i^2$, we need to find a suitable sequence of fixed self-adjoint matrices $\{\mathbf{A}_i\}$ of dimension $2Q$ (where $\preccurlyeq$ refers to the semidefinite order on self-adjoint matrices). Indeed, it suffices to construct a diagonal matrix defined as $\lambda_{\max}\big(\mathcal{S}(\mathbf{M}_i)\big)\mathbf{Id}_{2Q}$ for ensuring $\mathcal{S}(\mathbf{M}_i)^2 \preccurlyeq \big(\lambda_{\max}\big(\mathcal{S}(\mathbf{M}_i)\big)\mathbf{Id}_{2Q}\big)^2$. More precisely, since for every $i$ we have $\lambda_{\max}\big(\mathcal{S}(\mathbf{M}_i)\big) \leq a_i$, we fix $\mathbf{A}_i$ as a diagonal matrix with $a_i$ on the diagonal, *i.e.* $\mathbf{A}_i \overset{def}{=} a_i\mathbf{Id}_{2Q}$, with $\|\sum_i \mathbf{A}_i^2\| = \sum_i a_i^2 = \sigma^2$. Finally, we can invoke Theorem 8 to obtain the concentration inequality (A.19). □

In order to make use of this corollary, we rewrite confusion matrices as sums of example-based confusion matrices. That is, for each example $(\mathbf{x}_i, y_i) \in S$, we define its empirical confusion matrix by $\mathbf{C}_i^f = (\hat{c}_{pq}(i))_{1\leq p,q\leq Q}$ as follows:

$$\forall p,q, \hat{c}_{pq}(i) \overset{def}{=} \begin{cases} 0 & \text{if } q = p \\ \dfrac{1}{m_{y_i}}\mathbb{I}(f(\mathbf{x}) = q)\mathbb{I}(y_i = p) & \text{otherwise.} \end{cases}$$

where $m_{y_i}$ is the number of examples of class $y_i \in Y$ belonging to $S$. Given an example $(\mathbf{x}_i, y_i) \in S$, the example-based confusion matrix contains at most one non zero-element when $f$ misclassifies $(\mathbf{x}_i, y_i)$. In the same way, when $f$ correctly classifies $(\mathbf{x}_i, y_i)$ then the example-based confusion matrix is equal to $\mathbf{0}$. Concretely, for every sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ and every $f \in \mathcal{F}$, our error measure is then $\mathbf{C}_S^f = \sum_{i=1}^m \mathbf{C}_i^f$. It naturally appears that we penalize only when $f$ errs.

We further introduce the random square matrices $\mathbf{C'}_i^f$:

$$\mathbf{C'}_i^f = \mathbf{C}_i^f - \mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_i^f, \tag{A.20}$$

which verify $\mathbb{E}_i\mathbf{C'}_i^f = 0$.

We have yet to find a suitable $a_i$ for a given $\mathbf{C'}_i^f$. Let $\lambda_{\max_i}$ be the maximum singular value of $\mathbf{C'}_i^f$. It is easy to verify that $\lambda_{\max_i} \leq \frac{1}{m_{y_i}}$. Thus, for all $i$ we fix $a_i$ equal to $\frac{1}{m_{y_i}}$.

Finally, with the introduced notations, Corollary 5 leads to the following concentration inequality:

$$\mathbb{P}\left\{\|\sum_{i=1}^m \mathbf{C'}_i^f\| \geq \epsilon\right\} \leq 2.Q.\exp\left(\frac{-\epsilon^2}{8\sigma^2}\right). \tag{A.21}$$

This inequality (A.21) allows us to demonstrate our Theorem 12 by following the process of McAllester (2003), Seegerr (2002), Langford (2005).

### A.5.2 "Three Step Proof" Of Our Bound

First, thanks to concentration inequality (A.21), we prove the following lemma.

**Lemma 3** *Let $Q$ be the size of $\mathbf{C}_S^f$ and $\mathbf{C'}_i^f = \mathbf{C}_i^f - \mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_i^f$ defined as in (A.20). Then the following bound holds for any $\delta \in (0, 1]$:*

$$\mathbb{P}_{S\sim\mathfrak{D}_m}\left\{\mathbb{E}_{f\sim\mathfrak{P}}\left[\exp\left(\frac{1-8\sigma^2}{8\sigma^2}\|\sum_{i=1}^m \mathbf{C'}_i^f\|^2\right)\right] \leq \frac{2Q}{8\sigma^2\delta}\right\} \geq 1-\delta$$

*Proof.* For readability reasons, we note $\mathbf{C'}^f_S = \sum_{i=1}^m \mathbf{C'}^f_i$. If $Z$ is a real valued random variable so that $\mathbb{P}(Z \geq z) \leq k\exp(-n.g(z))$ with $g(z)$ non-negative, non-decreasing and $k$ a constant, then $\mathbb{P}(\exp((n-1)g(Z)) \geq \nu) \leq \min(1, k\nu^{-n/(n-1)})$. We apply this to the concentration inequality (A.21). Choosing $g(z) = z^2$ (non-negative), $z = \epsilon$, $n = \frac{1}{8\sigma^2}$ and $k = 2Q$, we obtain the following result:

$$\mathbb{P}\left\{\exp\left(\frac{1-8\sigma^2}{8\sigma^2}\|\mathbf{C'}^f_S\|\right) \geq \nu\right\} \leq \min\left(1, 2Q\nu^{-1/(1-8\sigma^2)}\right).$$

Note that $\exp\left(\frac{1-8\sigma^2}{8\sigma^2}\|\mathbf{C'}^f_S\|\right)$ is always non-negative. Hence it allows us to compute its expectation as:

$$\begin{aligned}
&\mathbb{E}\left[\exp\left(\frac{1-8\sigma^2}{8\sigma^2}\|\mathbf{C'}^f_S\|\right)\right] \\
&= \int_0^\infty \mathbb{P}\left\{\exp\left(\frac{1-8\sigma^2}{(8\sigma^2)}\|\mathbf{C'}^f_S\|\right) \geq \nu\right\} d\nu \\
&\leq 2Q + \int_1^\infty 2Q\nu^{-1/(1-8\sigma^2)} d\nu \\
&= 2Q - 2Q\frac{1-8\sigma^2}{8\sigma^2}\left[\nu^{-8\sigma^2/(1-8\sigma^2)}\right]_1^\infty \\
&= 2Q + 2Q\frac{1-8\sigma^2}{8\sigma^2} \\
&= \frac{2Q}{8\sigma^2}.
\end{aligned}$$

For a given classifier $f \in \mathcal{F}$, we have:

$$\mathbb{E}_{S\sim\mathfrak{D}^m}\left[\exp\left(\frac{1-8\sigma^2}{8\sigma^2}\|\mathbf{C'}^f_S\|\right)\right] \leq \frac{2Q}{8\sigma^2} \tag{A.22}$$

Then, if $\mathfrak{P}$ is a probability distribution over $\mathcal{F}$, Equation (A.22) implies that:

$$\mathbb{E}_{S\sim\mathfrak{D}^m}\left[\mathbb{E}_{f\sim\mathfrak{P}}\exp\left(\frac{1-8\sigma^2}{8\sigma^2}\|\mathbf{C'}^f_S\|\right)\right] \leq \frac{2Q}{8\sigma^2} \tag{A.23}$$

Using Markov's inequality [1], we obtain the result of the lemma. □

The second step to prove Theorem 12 is to use the shift given in McAllester (2003). We recall this result in the following lemma.

**Lemma 4** (Donsker-Varadhan inequality Donsker and Varadhan (1975)) *Given the Kullback-Leibler divergence* [2] $KL(\mathfrak{Q}\|\mathfrak{P})$ *between two distributions* $\mathfrak{P}$ *and* $\mathfrak{Q}$ *and let* $g(\cdot)$ *be a function, we have:*

$$\mathbb{E}_{a\sim Q}\Big[g(b)\Big] \leq KL(\mathfrak{Q}\|\mathfrak{P}) + \ln \mathbb{E}_{x\sim P}\Big[\exp(g(b))\Big].$$

*Proof.* See McAllester (2003). □

1. see Theorem 9 in Appendix.
2. The KL-divergence is defined in Equation (A.1).

Recall that $\mathbf{C'}_S^f = \sum_{i=1}^m \mathbf{C'}_i^f$. With $g(b) = \frac{1-8\sigma^2}{8\sigma^2}b^2$ and $b = \|\mathbf{C'}_S^f\|$, Lemma 4 implies:

$$\mathbb{E}_{f\sim\mathfrak{Q}}\left[\frac{1-8\sigma^2}{8\sigma^2}\|\mathbf{C'}_S^f\|^2\right] \leq KL(\mathfrak{Q}\|\mathfrak{P}) + \ln \mathbb{E}_{f\sim\mathfrak{P}}\left[\exp\left(\frac{1-8\sigma^2}{8\sigma^2}\|\mathbf{C'}_S^f\|^2\right)\right]. \tag{A.24}$$

The last step that completes the proof of Theorem 12 consists in applying the result we obtained in Lemma 3 to Equation (A.24). Then, we have:

$$\mathbb{E}_{f\sim\mathfrak{Q}}\left[\frac{1-8\sigma^2}{8\sigma^2}\|\mathbf{C'}_S^f\|^2\right] \leq KL(\mathfrak{Q}\|\mathfrak{P}) + \ln\frac{2Q}{8\sigma^2\delta}. \tag{A.25}$$

Since $g(\cdot)$ is clearly convex, we apply Jensen's inequality [3] to (A.25). Then, with probability at least $1-\delta$ over $S$, and for every distribution $\mathfrak{Q}$ on $\mathcal{F}$, we have:

$$\left(\mathbb{E}_{f\sim\mathfrak{Q}}\|\mathbf{C'}_S^f\|\right)^2 \leq \frac{8\sigma^2}{1-8\sigma^2}\left(KL(\mathfrak{Q}\|\mathfrak{P}) + \ln\frac{2Q}{8\sigma^2\delta}\right). \tag{A.26}$$

Since $\mathbf{C'}_S^f = \sum_{i=1}^m \left[\mathbf{C}_i^f - \mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_i^f\right]$, then the bound (A.26) is quite similar to the one given in Theorem 12.

We present in the next section, the calculations leading to our PAC-Bayesian generalization bound.

### A.5.3 Simplification

We first compute the variance parameter $\sigma^2 = \sum_{i=1}^m a_i^2$. For that purpose, in Section A.5.1 we showed that for each $i \in \{1,\ldots,m\}$, we can choose $a_i = \frac{1}{m_{y_i}}$, where $y_i$ is the class of the $i$-th example and $m_{y_i}$ is the number of examples of class $y_i$. Thus we have:

$$\sigma^2 = \sum_{i=1}^m \frac{1}{m_{y_i}^2} = \sum_{y=1}^Q \sum_{i:y_i=y} \frac{1}{m_y^2} = \sum_{y=1}^Q \frac{1}{m_y}.$$

For sake of simplification of Equation (A.26) and since the term on the right side of this equation is an increasing function with respect to $\sigma^2$, we propose to upper-bound $\sigma^2$:

$$\sigma^2 = \sum_{y=1}^Q \frac{1}{m_y} \leq \frac{Q}{\min_{y=1,\ldots,Q} m_y}. \tag{A.27}$$

Let $m_- \stackrel{def}{=} \min_{y=1,\ldots,Q} m_y$, then using Equation (A.27), we obtain the following bound from Equation (A.26):

$$\left(\mathbb{E}_{f\sim\mathfrak{Q}}[\|\mathbf{C'}_S^f\|]\right)^2 \leq \frac{8Q}{m_- - 8Q}\left(KL(\mathfrak{Q}\|\mathfrak{P}) + \ln\frac{m_-}{4\delta}\right).$$

Then:

$$\mathbb{E}_{f\sim\mathfrak{Q}}[\|\mathbf{C'}_S^f\|] \leq \sqrt{\frac{8Q}{m_- - 8Q}\left(KL(\mathfrak{Q}\|\mathfrak{P}) + \ln\frac{m_-}{4\delta}\right)}. \tag{A.28}$$

3. see Theorem 10 in Appendix.

It remains to replace $\mathbf{C'}_S^f = \sum_{i=1}^m \left[\mathbf{C}_i^f - \mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_i^f\right]$. Recall that $\mathbf{C}^{G_\mathfrak{Q}} = \mathbb{E}_{f\sim\mathfrak{Q}}\mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_S^f$ and $\mathbf{C}_S^{G_\mathfrak{Q}} = \mathbb{E}_{f\sim\mathfrak{Q}}\mathbf{C}_S^f$, we obtain:

$$\begin{aligned}
\mathbb{E}_{f\sim\mathfrak{Q}}[\|\mathbf{C'}_S^f\|] &= \mathbb{E}_{f\sim\mathfrak{Q}}\left[\|\sum_{i=1}^m\left[\mathbf{C}_i^f - \mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_i^f\right]\|\right] \\
&= \mathbb{E}_{f\sim\mathfrak{Q}}\left[\|\sum_{i=1}^m\left[\mathbf{C}_i^f\right] - \sum_{i=1}^m\left[\mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_i^f\right]\|\right] \\
&= \mathbb{E}_{f\sim\mathfrak{Q}}\left[\|\mathbf{C}_S^f - \mathbb{E}_{S\sim\mathfrak{D}_m}\left[\sum_{i=1}^m\mathbf{C}_i^f\right]\|\right] \\
&= \mathbb{E}_{f\sim\mathfrak{Q}}\left[\|\mathbf{C}_S^f - \mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_S^f\|\right] \\
&\geq \|\mathbb{E}_{f\sim\mathfrak{Q}}\left[\mathbf{C}_S^f - \mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_S^f\right]\| \\
&= \|\mathbb{E}_{f\sim\mathfrak{Q}}\mathbf{C}_S^f - \mathbb{E}_{f\sim\mathfrak{Q}}\mathbb{E}_{S\sim\mathfrak{D}_m}\mathbf{C}_S^f\| \\
&= \|\mathbf{C}_S^{G_\mathfrak{Q}} - \mathbf{C}^{G_\mathfrak{Q}}\|. \qquad \text{(A.29)}
\end{aligned}$$

By substituting the left part of the inequality (A.28) with the term (A.29), we find the bound of our Theorem 12.

## A.6 Discussion and Future Work

This work gives rise to many interesting questions, among which the following ones.

Some perspectives will be focused on instantiating our bound given in Theorem 12 for specific multi-class frameworks, such as multi-class SVM Weston and Watkins (1998), Crammer and Singer (2002), Lee et al. (2004) and multi-class boosting (AdaBoost.MH/AdaBoost.MR Schapire and Singer (1999), SAMME Zhu et al. (2009), AdaBoost.MM Mukherjee and Schapire (2011)). Taking advantage of our theorem while using the confusion matrices, may allow us to derive new generalization bounds for these methods.

Additionally, we are interested in seeing how effective learning methods may be derived from the risk bound we propose. For instance, in the binary PAC-Bayes setting, the algorithm MinCq proposed by Laviolette et al. (2011) minimizes a bound depending on the first two moments of the margin of the $\mathcal{Q}$-weighted majority vote. From our Theorem 12 and with a similar study, we would like to design a new multi-class learning algorithm and observe how sound such an algorithm could be. This would probably require the derivation of a Cantelli-Tchebycheff deviation inequality in the matrix case.

Besides, it might be very interesting to see how the noncommutative/matrix concentration inequalities provided by Tropp (2011) might be of some use for other kinds of learning problem such as multi-label classification, label ranking problems or structured prediction issues.

Finally, the question of extending the present work to the analysis of algorithms learning (possibly infinite-dimensional) operators as Abernethy et al. (2009) is also very exciting.

## A.7 Conclusion

In this paper, we propose a new PAC-Bayesian generalization bound that applies in the multi-class classification setting. The originality of our contribution is that we consider the confusion matrix as an error measure. Coupled with the use of the operator norm on matrices, we are capable of providing generalization bound on the 'size' of confusion matrix (with the idea that the smaller the norm of the confusion matrix of the learned classifier, the better it is for the classification task at hand). The derivation of our result takes advantage of the concentration inequality proposed by Tropp (2011) for the sum of random self-adjoint matrices, that we directly adapt to square matrices which are not self-adjoint.

The main results are presented in Theorem 12 and Corollary 3. The bound in Theorem 12 is given on the difference between the true risk of the Gibbs classifier and its empirical error. While the one given in Corollary 3 upper-bounds the risk of the Gibbs classifier by its empirical error.

An interesting point is that our bound depends on the minimal quantity $m_-$ of training examples belonging to the same class, for a given number of classes. If this value increases, *i.e.* if we have a lot of training examples, then the empirical confusion matrix of the Gibbs classifier tends to be close to its true confusion matrix. A point worth noting is that the bound varies as $O(1/\sqrt{m_-})$, which is a typical rate in bounds not using second-order information.

The present work gives rise to a few algorithmic and theoretical questions that we have discussed in the previous section.

# Bibliography


Naoki Abe. An iterative method for multi-class cost-sensitive learning. In *In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3–11, 2004. (Cited in pages 20 and 53.)

Jacob Abernethy, Francis Bach, Theodoros Evgeniou, and Jean-Philippe Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826, 2009. (Cited in page 116.)

Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004*, volume 3201 of *Lecture Notes in Computer Science*, pages 39–50. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-23105-9. (Cited in page 20.)

Muhammad Awais, Fei Yan, Krystian Mikolajczyk, and Josef Kittler. Novel fusion methods for pattern recognition. In *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part I*, ECML PKDD'11, pages 140–155, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23779-9. (Cited in page 15.)

Maria F. Balcan, Avrim Blum, and Ke Yang. Co-Training and Expansion: Towards Bridging Theory and Practice. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 89–96. MIT Press, Cambridge, MA, 2005. (Cited in page 16.)

R. Barandela, R.M. Valdovinos, and J.S. Sanchez. New applications of ensembles of classifiers. *Pattern Analysis & Applications*, 6(3):245–256, 2003. ISSN 1433-7541. (Cited in page 20.)

Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004. ISSN 1931-0145. doi: 10.1145/1007730.1007735. URL http://doi.acm.org/10.1145/1007730.1007735. (Cited in page 19.)

Avrim B. Blum and Tom M. Mitchell. Combining labeled and unlabeled data with co-training. In *11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998. (Cited in pages 16, 49, 97, and 99.)

Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *J. Machine Learning Research*, 6:1579–1619, 2005. (Cited in page 53.)

Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159, July 1997. ISSN 0031-3203. (Cited in page 21.)

Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, August 1996. (Cited in page 4.)

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125. (Cited in page 4.)

Olivier Catoni. 4. gibbs estimators. In *Statistical Learning Theory and Stochastic Optimization*, volume 1851, pages 111–135. Springer, 2004. (Cited in page 104.)

Olivier Catoni. PAC-bayesian supervised classification: The thermodynamics of statistical learning. *ArXiv e-prints*, 2007. (Cited in pages 62 and 104.)

O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. In *JMLR Workshop and Conference Proceedings*, volume 14, pages 1–24, 2011. (Cited in page 53.)

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002. (Cited in page 18.)

Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *PKDD'03*, pages 107–119, 2003. (Cited in page 63.)

Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, 1999. (Cited in page 10.)

Corinna Cortes and Mehryar Mohri. Auc optimization vs. error rate minimization. In MIT Press, editor, *Advances in Neural Information Processing Systems (NIPS 2003)*, volume 16, Vancouver, Canada, 2004. (Cited in page 53.)

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3): 273–297, September 1995. ISSN 0885-6125. (Cited in page 3.)

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002. (Cited in pages 104 and 116.)

Mark Culp, George Michailidis, and Kjell Johnson. On multi-view learning with additive models. *Annals of Applied Statistics*, 3:292–318, 2009. (Cited in page 28.)

Virginia R. de Sa and Dana H. Ballard. Category learning through multimodality sensing. *Neural Comput.*, 10(5):1097–1117, July 1998. ISSN 0899-7667. (Cited in page 11.)

Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Mach. Learn.*, 46(1-3):225–254, March 2002. ISSN 0885-6125. (Cited in page 10.)

T. Diethe, D.R. Hardoon, and J. Shawe-Taylor. Multiview fisher discriminant analysis. Technical report, Presented at the NIPS 2008 workshop Learning from Multiple Sources, 2008. URL http://www.tomdiethe.com/research/papers/multiview_learning_with_labels.pdf. (Cited in page 13.)

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn.*, 40(2):139–157, August 2000. ISSN 0885-6125. (Cited in page 7.)

David Donsker and S. S. Varadhan. Asymptotic evaluation of certain markov process expectations for large time. *Communications on Pure and Applied Mathematics*, 28, 1975. (Cited in page 114.)

Charles Elkan. The foundations of cost-sensitive learning. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001. (Cited in page 53.)

Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1): 18–36, 2004. (Cited in pages 19 and 53.)

Benoit Favre, Dilek Hakkani-Tür, and Sebastien Cuendet. Icsiboost. http://code.google.come/p/icsiboost, 2007. (Cited in page 44.)

Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006. (Cited in page 52.)

Yinfu Feng, Jun Xiao, Yueting Zhuang, and Xiaoming Liu. Adaptive unsupervised multi-view feature selection for visual concept recognition. In KyoungMu Lee, Yasuyuki Matsushita, JamesM. Rehg, and Zhanyi Hu, editors, *Computer Vision – ACCV 2012*, volume 7724 of *Lecture Notes in Computer Science*, pages 343–357. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-37330-5. (Cited in page 13.)

Dean P. Foster, Sham M. Kakade, and Tong Zhang. Multi-view dimensionality reduction via canonical correlation analysis. Technical report, Toyota Technological Institute at Chicago, 2008. URL http://www.ttic.edu/technical_reports/ttic-tr-2008-4.pdf. (Cited in page 13.)

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL http://archive.ics.uci.edu/ml. (Cited in pages 5, 13, and 63.)

Yoav Freund. Boosting a weak learning algorithm by majority. *Inf. Comput.*, 121(2): 256–285, September 1995. ISSN 0890-5401. (Cited in pages 4 and 5.)

Yoav Freund. An adaptive version of the boost by majority algorithm. *Mach. Learn.*, 43(3):293–318, June 2001. ISSN 0885-6125. (Cited in pages 7 and 10.)

Yoav Freund. A more robust boosting algorithm. Technical report, Computer Science and Engineering, UCSD, 2009. URL http://arxiv.org/abs/0905.2138. (Cited in page 10.)

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, pages 23–37, London, UK, UK, 1995. Springer-Verlag. ISBN 3-540-59119-2. (Cited in pages 5 and 7.)

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *In Proceedings of the International Conference on Machine Learning*, pages 148–156, 1996. (Cited in pages 5, 7, and 104.)

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, August 1997. ISSN 0022-0000. (Cited in page 7.)

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, December 2003. ISSN 1532-4435. (Cited in page 10.)

J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics*, 38(2):337–407, 2000. (Cited in page 10.)

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001. (Cited in page 10.)

Mikel Galar, Alberto Fernández, Edurne Barrenechea Tartas, Humberto Bustince Sola, and Francisco Herrera. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(4):463–484, 2012. URL http://dblp.uni-trier.de/db/journals/tsmc/tsmcc42.html#GalarFTSH12. (Cited in page 20.)

Sally A. Goldman and Yan Zhou. Enhancing supervised learning with unlabeled data. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 327–334, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2. (Cited in page 16.)

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *J. Mach. Learn. Res.*, 12:2211–2268, July 2011. ISSN 1532-4435. (Cited in page 17.)

Hongyu Guo and Herna L. Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *SIGKDD Explor. Newsl.*, 6(1):30–39, June 2004. ISSN 1931-0145. (Cited in page 19.)

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, editors, *Advances in Intelligent Computing*, volume 3644 of *Lecture Notes in Computer Science*, pages 878–887. Springer Berlin Heidelberg, 2005. (Cited in page 19.)

David J. Hand and Robert J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Mach. Learn.*, 45(2):171–186, October 2001. (Cited in pages 21 and 63.)

Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284, September 2009. (Cited in pages 20 and 53.)

Haibo He, Yang Bai, E.A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328, 2008. (Cited in page 19.)

Jean-Christophe Janodet, Marc Sebban, and Henri-Maxime Suchier. Boosting classifiers built from different subsets of features. *Fundam. Inf.*, 96:89–109, January 2009. ISSN 0169-2968. (Cited in pages 11, 17, 39, 48, 49, and 99.)

Michael J. Jones and Paul Viola. Fast multi-view face detection. In *Proc. of Computer Vision and Pattern Recognition*, 2003. (Cited in page 11.)

R. Wang K. Tang and T. Chen 7-11 August 2011. Towards maximizing the area under the roc curve for multi-class classification problems. In *Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, 2011. (Cited in pages 21 and 53.)

Hachem Kadri, Stéphane Ayache, Cécile Capponi, Sokol Koço, François-Xavie Dupé, and Emilie Morvant. The multi-task learning view of multimodal data. In *Proceedings of the Asian Conference of Machine Learning (ACML 2013)*, 2013. (Cited in pages 49 and 101.)

Michael Kearns and Leslie G. Valiant. Crytographic limitations on learning boolean formulae and finite automata. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, STOC '89, pages 433–444, New York, NY, USA, 1989. ACM. ISBN 0-89791-307-8. doi: 10.1145/73007.73049. URL http://doi.acm.org/10.1145/73007.73049. (Cited in page 4.)

Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 20:226–239, 1998. (Cited in page 15.)

Sokol Koço and Cécile Capponi. A boosting approach to multiview classification with cooperation. In *European Conference on Machine Learning (ECML)*, pages 209–228, 2011. (Cited in page 28.)

Sokol Koço and Cécile Capponi. On multi-class classification through the minimization of the confusion matrix norm. *Proceedings of the 5th Asian Conference in Machine Learning*, November 2013. (Cited in page 52.)

Sokol Koço, Cécile Capponi, and Frédéric Béchet. Applying multiview learning algorithms to human-human conversation classification. In *Annual Conference of the International Speech Communication Association, Interspeech*, September 2012. (Cited in page 28.)

Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51 (2):181–207, 2003. (Cited in pages 48, 50, and 102.)

Alexandre Lacasse, François Laviolette, Mario Marchand, Pascal Germain, and Nicolas Usunier. PAC-bayes bounds for the risk of the majority vote and the variance of the Gibbs classifier. In *Adv. in Neural Processing Systems (NIPS)*, 2007. (Cited in page 104.)

Christoph H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958, 2009. (Cited in pages viii, x, 73, 87, and 88.)

Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, December 2004. ISSN 1532-4435. (Cited in pages 15, 49, and 78.)

John Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306, 2005. (Cited in pages 62, 104, 106, 112, and 113.)

John Langford and John Shawe-Taylor. PAC-bayes & margins. In *Advances in Neural Information Processing Systems 15*, pages 439–446. MIT Press, 2002. (Cited in page 104.)

John Langford, Matthias Seeger, and Nimrod Megiddo. An improved predictive accuracy bound for averaging classifiers. In *Proc. of the International Conference on Machine Learning*, pages 290–297, 2001. (Cited in page 104.)

François Laviolette, Mario Marchand, and Jean-Francis Roy. From PAC-Bayes Bounds to Quadratic Programs for Majority Votes. In *Proc. of the International Conference on Machine Learning*, June 2011. (Cited in pages 15, 104, and 116.)

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99:67–81, 2004. (Cited in pages 104 and 116.)

David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0164-8. (Cited in page 11.)

Marcus A. Maloof. Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 Workshop on Learning from Imbalanced Data Sets II*, 2003. (Cited in page 20.)

Muharram Mansoorizadeh and Nasrollah Moghaddam Charkari. Multimodal information fusion application to human emotion recognition from face and speech. *Multimedia Tools and Applications*, 49(2):277–297, 2010. (Cited in page 28.)

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. *Boosting algorithms as gradient descent in function space*, volume 11. NIPS, 1999. (Cited in pages 10 and 32.)

Francesco Masulli and Sushmita Mitra. Natural computing methods in bioinformatics: A survey. *Information Fusion*, 10(3):211–216, july 2009. (Cited in page 28.)

David A. McAllester. Some PAC-bayesian theorems. *Machine Learning*, 37:355–363, 1999a. (Cited in pages 62 and 104.)

David A. McAllester. PAC-bayesian model averaging. In *Proceedings of the annual conference on Computational learning theory (COLT)*, pages 164–170, 1999b. (Cited in page 104.)

David A. McAllester. Simplified PAC-bayesian margin bounds. In *Proc. of the annual conference on Computational learning theory (COLT)*, pages 203–215, 2003. (Cited in pages 105, 106, 112, 113, and 114.)

Emilie Morvant, Sokol Koço, and Liva Ralaivola. PAC-Bayesian Generalization Bound on Confusion Matrix for Multi-Class Classification. In *International Conference on Machine Learning*, pages 815–822, 2012. (Cited in pages 5, 52, 54, 62, 97, and 103.)

Indraneel Mukherjee and Robert E. Schapire. A theory of multiclass boosting. *CoRR*, abs/1108.2989, 2011. (Cited in pages 8, 9, 10, 31, 35, 41, 50, 52, 54, 58, 59, 66, 104, and 116.)

Ion Muslea and Craig A. Knoblock. Active learning with multiple views. *J. Artif. Intell. Res. (JAIR)*, 27:203–233, 2006. (Cited in pages 16 and 97.)

Richard Nock and Patrice Lefaucheur. A robust boosting algorithm. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Machine Learning: ECML 2002*, volume 2430 of *Lecture Notes in Computer Science*, pages 319–331. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-44036-9. (Cited in page 10.)

David W. Opitz and Jude W. Shavlik. Generating Accurate and Diverse Members of a Neural-Network Ensemble. *Neural Information Processing Systems*, 8:535–541, 1996. (Cited in page 15.)

V.I. Paulsen. *Completely bounded maps and operator algebras*. Cambridge studies in advanced mathematics. Cambridge University Press, 2002. (Cited in page 105.)

Jing Peng, Costin Barbu, Guna Seetharaman, Wei Fan, Xian Wu, and Kannappan Palaniappan. Shareboost: boosting for multi-view learning with performance guarantees. In *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part II*, ECML PKDD'11, pages 597–612, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23782-9. (Cited in page 49.)

J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986. ISSN 0885-6125. (Cited in page 3.)

Liva Ralaivola. Confusion-based online learning and a passive-aggressive scheme. In *Neural Information Processing Systems Conference*, 2012. (Cited in pages 5, 52, 54, 56, 96, and 101.)

Amir Saffari, Christian Leistner, Martin Godec, and Horst Bischof. Robust multi-view boosting with priors. In *Proceedings of the 11th European conference on computer vision conference on Computer vision: Part III*, ECCV'10, pages 776–789, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15557-X, 978-3-642-15557-4. (Cited in page 49.)

Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, July 1990. ISSN 0885-6125. doi: 10.1023/A:1022648800760. URL http://dx.doi.org/10.1023/A:1022648800760. (Cited in pages 4 and 5.)

Robert E. Schapire. Advances in boosting. In *UAI*, pages 446–452, 2002. (Cited in page 8.)

Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012. ISBN 0262017180, 9780262017183. (Cited in page 6.)

Robert E. Schapire and Yoav Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999. (Cited in pages 8, 104, and 116.)

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998. ISSN 0090-5364. (Cited in pages 38 and 40.)

Matthias Seegerr. PAC-bayesian generalization error bounds for gaussian process classification. *Journal of Machine Learning Research*, 3:233–269, 2002. (Cited in pages 104, 106, 112, and 113.)

Vikas Sindhwani and Partha Niyogi. A co-regularized approach to semi-supervised learning with multiple views. In *Proceedings of the ICML Workshop on Learning with Multiple Views*, 2005. (Cited in page 16.)

Cees Snoek, Marcel Worring, and Arnold Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 399–402, New York, NY, USA, 2005. ACM. ISBN 1-59593-044-2. (Cited in page 16.)

Karthik Sridharan and Sham M. Kakade. An information theoretic framework for multi-view learning. In *Annual Conference on Computational Learning Theory*, pages 403–414, 2008. (Cited in page 28.)

Jerzy Stefanowski and Szymon Wilk. Selective pre-processing of imbalanced data for improving classification performance. In *Proceedings of the 10th international conference on Data Warehousing and Knowledge Discovery*, DaWaK '08, pages 283–292, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-85835-5. (Cited in page 19.)

Shiliang Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, pages 1–8, 2013. ISSN 0941-0643. (Cited in page 17.)

Y. Sun, M.S. Kamel, and Y. Wang. Boosting for learning multiple classes with imbalanced class distribution. In *In 2006 IEEE International Conference on Data Mining, HongKong*, pages 592–602, 2006. (Cited in page 63.)

Yanmin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recogn.*, 40(12): 3358–3378, December 2007. ISSN 0031-3203. (Cited in pages 19 and 68.)

Feng Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *ICCV*, pages 1–8, 2007. (Cited in page 16.)

Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. Unsupervised feature selection for multi-view data in social media. In *SDM*, 2013. (Cited in page 13.)

K.M. Ting. "a comparative study of cost-sensitive boosting algorithms". In *Int'l Conf. Machine Learning*, pages 983–990, 2000. (Cited in pages 19, 53, and 68.)

Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, pages 1–46, 2011. (Cited in pages 105, 106, 112, 116, and 117.)

L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984. ISSN 0001-0782. doi: 10.1145/1968.1972. URL http://doi.acm.org/10.1145/1968.1972. (Cited in pages 3, 4, and 11.)

L.J.P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review, 2008. URL http://www.iai.uni-bonn.de/~jz/dimensionality_reduction_a_comparative_review.pdf. (Cited in page 13.)

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998. ISBN 0471030031. URL http://www.worldcat.org/isbn/0471030031. (Cited in page 39.)

B.X. Wang and N. Japkowicz. Imbalanced data set learning with synthetic samples. In *Proc. IRIS Machine Learning Workshop*, 2004. (Cited in page 19.)

Huanjing Wang, Taghi M. Khoshgoftaar, and Amri Napolitano. Software measurement data reduction using ensemble techniques. *Neurocomputing*, 92:124 – 132, 2012. (Cited in page 53.)

Shuo Wang and Xin Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *Computational Intelligence and Data Mining, 2009. CIDM '09. IEEE Symposium on*, pages 324–331, 2009. (Cited in page 20.)

Shuo Wang and Xin Yao. Multiclass imbalance problems: Analysis and potential solutions. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42 (4):1119–1130, 2012. (Cited in pages ix, 20, 63, and 64.)

Shuo Wang, Huanhuan Chen, and Xin Yao. Negative correlation learning for classification ensembles. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8, 2010. (Cited in page 19.)

Wei Wang and Zhi hua Zhou. Multi-view active learning in the non-realizable case. In J. Lafferty, C. Williams, J. Shawe-taylor, R.s. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2388–2396. NIPS, 2010. URL http://books.nips.cc/papers/files/nips23/NIPS2010_0787.pdf. (Cited in page 16.)

Wei Wang and Zhi-Hua Zhou. On multi-view active learning and the combination with semi-supervised learning. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1152–1159, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390301. URL http://doi.acm.org/10.1145/1390156.1390301. (Cited in page 16.)

Jason Weston and Chris Watkins. Multi-class support vector machines, 1998. (Cited in pages 104 and 116.)

Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, December 2007. ISSN 0219-1377. (Cited in page 7.)

Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-486-3. (Cited in page 13.)

Shipeng Yu, Balaji Krishnapuram, Rómer Rosales, and R. Bharat Rao. Bayesian co-training. *J. Mach. Learn. Res.*, 12:2649–2680, November 2011. ISSN 1532-4435. (Cited in page 16.)

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *SIGIR*, pages 271–278, 2007. (Cited in page 53.)

Xiaoyu Zhang, Jian Cheng, Changsheng Xu, Hanqing Lu, and Songde Ma. Multi-view multi-label active learning for image classification. In *ICME*, pages 258–261. IEEE, 2009. ISBN 978-1-4244-4291-1. (Cited in page 16.)

Zhi-Hua Zhou and Xu-Ying Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010. (Cited in pages 53 and 57.)

Ji Zhu, Hui Zou, Saharon Rosset, and Trevor Hastie. Multi-class adaboost, 2009. (Cited in pages 10, 104, and 116.)

**Titre** Méthodes ensemblistes pour des problèmes de classification multi-vues et multi-classes avec déséquilibres

**Résumé** De nos jours, dans plusieurs domaines, tels que la bio-informatique ou le multimédia, les données peuvent être représentées par plusieurs ensembles d'attributs, appelés des *vues*. Ainsi, en multimédia, une vidéo peut être représenté par sa bande son ou les images qui la composent. Bien que représentant les mêmes objets, chaque vue est plus ou moins adaptée à une tâche d'apprentissage donnée. Par exemple, dans le cas de la classification des vidéos musicales selon leurs genres, la vue *son* est plus adaptée que la vue *image*. Pour une tâche de classification donnée, nous distinguons deux types de vues : les vues fortes sont celles adaptées à la tâche, les vues faibles sont adaptées à une (petite) partie de la tâche ; en classification multi-classes, chaque vue peut s'avérer forte pour reconnaître une classe, et faible pour reconnaître d'autres classes : une telle vue est dite déséquilibrée. Les travaux présentés dans cette thèse s'inscrivent dans le cadre de l'apprentissage supervisé et ont pour but de traiter les questions d'apprentissage multi-vue dans le cas des vues fortes, faibles et déséquilibrées. La première contribution de cette thèse est un algorithme d'apprentissage multi-vues théoriquement fondé sur le cadre de boosting multi-classes utilisé par AdaBoost.MM. Le but de cet algorithme est d'améliorer les performances des classifieurs appris sur les vues fortes en utilisant des informations contenues dans les vues faibles. Comparé aux méthodes existantes, la nouveauté de notre approche demeure dans l'utilisation de la coopération entre les vues afin de trouver la vue la plus adaptée pour chaque exemple. La seconde partie de cette thèse concerne la mise en place d'un cadre général pour les méthodes d'apprentissage de classes déséquilibrées (certaines classes sont plus représentées que les autres). Ce cadre consiste à utiliser la norme de la matrice de confusion comme mesure d'erreur pour un classifieur donné. Dans ce cadre, nous proposons une extension de AdaBoost.MM permettant de prendre en compte des classes déséquilibrées. Dans la troisième partie, nous traitons le problème des vues déséquilibrées en combinant notre approche des classes déséquilibrées et la coopération entre les vues mise en place pour appréhender la classification multi-vues. Contrairement à la coopération de la première méthode, utilisée pour trouver la meilleure vue pour chaque exemple, dans ce cas, la coopération permet de trouver la meilleure vue pour chaque classe. Autrement dit, la coopération passe de l'espace d'entrée à celui de sortie. Cela nous permet de dériver différentes méthodes d'apprentissage permettant de combiner plusieurs classifieurs appris sur les vues. Parmi ces méthodes, nous proposons une méthode de boosting proche de la première méthode. Afin de tester les méthodes sur des données réelles, nous nous intéressons au problème de classification d'appels téléphoniques, qui a fait l'objet du projet ANR DECODA. Ainsi chaque partie traite différentes facettes du problème. La première partie présente le problème en tant que problème multi-vues (sacs de mots et de concepts, prosodie, mesures de l'interaction entre les locuteurs, etc.): est-ce que la séparation des vues, traitées par coopération, améliore les performances d'une fusion précoce de toutes ces vues? Dans la seconde partie, nous considérons le problème des classes déséquilibrées (par exemple, il y a beaucoup plus d'appels concernant les *itinéraires* que les *procès verbaux*). La troisième partie regroupe les aspects multi-vues et les déséquilibres entre classes.

**Mots-clés** apprentissage automatique, apprentissage supervisé, apprentissage multi-vues, vues déséquilibrées, méthodes ensemblistes, boosting, coopération entre vues, matrices de confusion, classes déséquilibrées

**Title** Tackling the uneven views problem with cooperation based ensemble learning methods

**Abstract** Nowadays, in many fields, such as bioinformatics or multimedia, data may be described using different sets of features, also called *views*. For instance, in multimedia, a video may be represented by the audio or the images contained therein. Even though they represent the same object, each view is more or less adapted for a given task. For example, if the goal is to classify musical videos by genre, then the *audio* view is more adapted than the *image* one. For a given classification task, we distinguish two types of views: *strong* views, which are suited for the task, and *weak* views suited for a (small) part of the task; in multi-class learning, a view can be *strong* with respect to some (few) classes and *weak* for the rest of the classes: these are *imbalanced* views. The works presented in this thesis fall in the supervised learning setting and their aim is to address the problem of multi-view learning under strong, weak and imbalanced views, regrouped under the notion of *uneven views*. The first contribution of this thesis is a multi-view learning algorithm based on the same framework as AdaBoost.MM. The goal of this algorithm is to improve the performances of the classifiers learnt only on the strong views, through the use of information contained in the weaker views. Compared to existing methods, the novelty of our approach resides in promoting the cooperation between the views, so that each example can be processed by the most appropriate view. The second part of this thesis proposes a unifying framework for imbalanced classes supervised methods (some of the classes are more represented than others). The novelty of this framework consists in using the norm of the confusion matrix (for a given classifier) as an error measure. Based on this framework, we proposed an extension of AdaBoost.MM allowing to take into consideration the imbalance between classes. In the third part of this thesis, we tackle the uneven views problem through the combination of the imbalanced classes framework and the between-views cooperation used to take advantage of the multiple views. Contrary to the cooperation in the first part, used to find the most suitable view for each example, in this case, the cooperation is employed to find the most suitable view(s) for each *class*. That is, the cooperation is transposed from the input space to the output one, allowing to deriver several ensemble based learning method for combining classifier coming from the various views. Amidst these method, we propose a boosting algorithm similar to the one introduced in the first part. In order to test the proposed methods on real-world data, we consider the task of phone calls classifications, which constitutes the subject of the ANR DECODA project. Each part of this thesis deals with different aspects of the problem. In the first part, we deal with the multi-view aspect of dialogs (bag of words, prosody, interaction between speakers, etc.), in order to address the question of wether installing some cooperation between the views, leads to better performance than early fusion methods. The imbalance classes aspect of the data is the main focus of the second part (for instance, there are more phone calls for *itineraries* than there are for *fines*). The third part regroups the multi-view and imbalanced classes facets.

**Keywords** machine learning, supervised learning, multi-view learning, uneven views, ensemble methods, boosting, between-views cooperation, confusion matrix, imbalanced classes