

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

UNIVERSIDAD POLITÉCNICA DE MADRID

PHD THESIS

ARTIFICIAL INTELLIGENCE

NEW DEFINITIONS OF MARGIN FOR MULTI-CLASS BOOSTING ALGORITHMS

AUTHOR: Antonio Fernández Baldera

ADVISORS: Luis Baumela Molina, Jose Miguel Buenaposada Biencinto

JULY, 2015

New definitions of margin for multi-class Boosting algorithms

Antonio Fernández Baldera

July, 2015

Agradecimientos

Durante el tiempo que lleva concluir unos estudios de doctorado uno llega a sentir el apoyo de tantas personas que resulta difícil incluirlas en pocas líneas. En cualquier caso, trataré de hacer un buen ejercicio de memoria para recogerlos a todos en esta dedicatoria.

En primer lugar me veo en la obligación de agradecer a Luis Baumela todo el esfuerzo (y paciencia) que ha dedicado en mi formación y que ha derivado en la consecución de la presente tesis. Gracias por haber estado ahí ayudándome en tantos aspectos: proponiendo ideas, cuidando la manera de redactar, enseñando a resolver trámites, facilitando material, escuchando propuestas, orientando objetivos, etc. Ciertamente es imposible no aprender con una persona como él trabajando a tu lado. De verdad, muchas gracias.

Igualmente, no puedo dejar de agradecer a Jose Miguel Buenaposada por todos sus comentarios, tanto teóricos como experimentales. Qué fácil es mantener el entusiasmo por una idea cuando se tiene un compañero como él. En verdad sobra decir que parte de los objetivos alcanzados en este documento se han conseguido gracias a sus consejos. Muchísimas gracias.

Mis agradecimientos también a mis compañeros de laboratorio: Juan, Yadira, Pablo, David y Kendrick; por los buenos ratos que hemos llegado a pasar. Fuera del grupo, gracias también a Monse y a Elena, por amenizar los momentos del café. Agradecer de igual modo a Antonio Valdés por aportar su visión matemática a nuestros seminarios y a Aleix Martínez por haberme hecho un hueco en su extraordinario grupo.

Gracias a toda mi familia, como no, por haber estado siempre ahí. Gracias por hacerme sentir más confiado que nunca en mi potencial académico, especialmente después de tantos años de formación.

Por otro lado, gracias a todos los amigos que durante este tiempo me han permitido compartir tan buenos momentos e igualmente han sabido estar ahí cuando les he necesitado. Gracias por vuestro apoyo. A la gente de mi pueblo: Rocha, Galea, Jose Luis y Charly; a mis compadres de Badajoz: Felix y Javi; a mis excompañeros del INE: Alberto, Paz y Cristina; a los amigos de San Sebastián de los Reyes: Álvaro, Sergio y Jose; a mis excompañeros del máster: Arturo, Diego, Ernesto, Victor, Ghislain, Pablo y Raúl; a mi gente de Las Rozas: Cristina, Rosa y Carlos; y, como no, a mis amigos en Columbus: Fabián, Felipe y Adriana (y toda su familia). Igualmente, gracias a tantos otros amigos que no recojo en estas líneas pero que también me han hecho sentir afortunado durante estos años.

Como no podía ser de otro modo, mis más cariñosos agradecimientos a Aili Cutie Báez. Sobran los motivos.

Aparte, creo más que conveniente dedicar también mi agradecimiento al proyecto TIN2008-06815-C02-02 del Ministerio de Economía y Competitividad. Sin dicho proyecto no hubiera sido posible que esta investigación se llevara a cabo.

Abstract

The family of Boosting algorithms represents a type of classification and regression approach that has shown to be very effective in Computer Vision problems. Such is the case of detection, tracking and recognition of faces, people, deformable objects and actions. The first and most popular algorithm, AdaBoost, was introduced in the context of binary classification. Since then, many works have been proposed to extend it to the more general multi-class, multi-label, cost-sensitive, etc... domains. Our interest is centered in extending AdaBoost to two problems in the multi-class field, considering it a first step for upcoming generalizations.

In this dissertation we propose two Boosting algorithms for multi-class classification based on new generalizations of the concept of margin. The first of them, PIBoost, is conceived to tackle the multi-class problem by solving many binary sub-problems. We use a vectorial codification to represent class labels and a multi-class exponential loss function to evaluate classifier responses. This representation produces a set of margin values that provide a range of penalties for failures and rewards for successes. The stagewise optimization of this model introduces an asymmetric Boosting procedure whose costs depend on the number of classes separated by each weak-learner. In this way the Boosting procedure takes into account class imbalances when building the ensemble. The resulting algorithm is a well grounded method that canonically extends the original AdaBoost.

The second algorithm proposed, BAdaCost, is conceived for multi-class problems endowed with a cost matrix. Motivated by the few cost-sensitive extensions of AdaBoost to the multi-class field, we propose a new margin that, in turn, yields a new loss function appropriate for evaluating costs. Since BAdaCost generalizes SAMME, Cost-Sensitive AdaBoost and PIBoost algorithms, we consider our algorithm as a canonical extension of AdaBoost to this kind of problems. We additionally suggest a simple procedure to compute cost matrices that improve the performance of Boosting in standard and unbalanced problems.

A set of experiments is carried out to demonstrate the effectiveness of both methods against other relevant Boosting algorithms in their respective areas. In the experiments we resort to benchmark data sets used in the Machine Learning community, firstly for minimizing classification errors and secondly for minimizing costs. In addition, we successfully applied BAdaCost to a segmentation task, a particular problem in presence of imbalanced data. We conclude the thesis justifying the horizon of future improvements encompassed in our framework, due to its applicability and theoretical flexibility.

Keywords:
Machine Learning, AdaBoost, Multi-class Boosting, Margin-based classifiers, Cost-sensitive learning, Imbalanced data.

Resumen

La familia de algoritmos de Boosting son un tipo de técnicas de clasificación y regresión que han demostrado ser muy eficaces en problemas de Visión Computacional. Tal es el caso de los problemas de detección, de seguimiento o bien de reconocimiento de caras, personas, objetos deformables y acciones. El primer y más popular algoritmo de Boosting, AdaBoost, fue concebido para problemas binarios. Desde entonces, muchas han sido las propuestas que han aparecido con objeto de trasladarlo a otros dominios más generales: multiclase, multilabel, con costes, etc. Nuestro interés se centra en extender AdaBoost al terreno de la clasificación multiclase, considerándolo como un primer paso para posteriores ampliaciones.

En la presente tesis proponemos dos algoritmos de Boosting para problemas multiclase basados en nuevas derivaciones del concepto margen. El primero de ellos, PIBoost, está concebido para abordar el problema descomponiéndolo en subproblemas binarios. Por un lado, usamos una codificación vectorial para representar etiquetas y, por otro, utilizamos la función de pérdida exponencial multiclase para evaluar las respuestas. Esta codificación produce un conjunto de valores margen que conllevan un rango de penalizaciones en caso de fallo y recompensas en caso de acierto. La optimización iterativa del modelo genera un proceso de Boosting asimétrico cuyos costes dependen del número de etiquetas separadas por cada clasificador débil. De este modo nuestro algoritmo de Boosting tiene en cuenta el desbalanceo debido a las clases a la hora de construir el clasificador. El resultado es un método bien fundamentado que extiende de manera canónica al AdaBoost original.

El segundo algoritmo propuesto, BAdaCost, está concebido para problemas multiclase dotados de una matriz de costes. Motivados por los escasos trabajos dedicados a generalizar AdaBoost al terreno multiclase con costes, hemos propuesto un nuevo concepto de margen que, a su vez, permite derivar una función de pérdida adecuada para evaluar costes. Consideramos nuestro algoritmo como la extensión más canónica de AdaBoost para este tipo de problemas, ya que generaliza a los algoritmos SAMME, Cost-Sensitive AdaBoost y PIBoost. Por otro lado, sugerimos un simple procedimiento para calcular matrices de coste adecuadas para mejorar el rendimiento de Boosting a la hora de abordar problemas estándar y problemas con datos desbalanceados.

Una serie de experimentos nos sirven para demostrar la efectividad de ambos métodos frente a otros conocidos algoritmos de Boosting multiclase en sus respectivas áreas. En dichos experimentos se usan bases de datos de referencia en el área de Machine Learning, en primer lugar para minimizar errores y en segundo lugar para minimizar costes. Además, hemos podido aplicar BAdaCost con éxito a un proceso de segmentación, un caso particular de problema con datos desbalanceados. Concluimos justificando el horizonte de futuro que encierra el marco de trabajo que presentamos, tanto por su aplicabilidad como por su flexibilidad teórica.

Palabras clave:

Aprendizaje Automático, AdaBoost, Boosting Multiclase, Clasificadores basados en Margen, Aprendizaje con costes, Datos desbalanceados.

Contents

1 Introduction **1**

1.1 Motivation . . . 2

1.2 Main Contributions . . . 2

1.3 Structure of the Thesis . . . 3

2 Background on Boosting **5**

2.1 Background on Machine Learning . . . 5

2.1.1 Supervised Classification Problems . . . 7

2.2 Binary Boosting: AdaBoost . . . 8

2.2.1 Understanding AdaBoost . . . 9

2.2.2 Statistical View of Boosting . . . 11

2.3 Multi-class Boosting . . . 13

2.3.1 Boosting algorithms based on binary weak-learners . . . 13

2.3.2 Boosting algorithms based on vectorial encoding . . . 16

2.4 Cost-sensitive binary Boosting . . . 20

2.5 Other perspectives of Boosting . . . 23

3 Partially Informative Boosting **25**

3.1 Multi-class margin extension . . . 26

3.2 PIBoost . . . 28

3.2.1 AdaBoost as a special case of PIBoost . . . 31

3.2.2 Asymmetric treatment of partial information . . . 32

3.2.3 Common sense pattern . . . 32

3.3 Related work . . . 33

3.4 Experiments . . . 35

4 Multi-class Cost-sensitive Boosting **41**

4.1 Cost-sensitive multi-class Boosting . . . 41

4.1.1 Previous works . . . 43

4.1.2 New margin for cost-sensitive classification 46
4.2 BAdaCost: Boosting Adapted for Cost-matrix 47
4.2.1 Direct generalizations 49
4.3 Experiments 50
4.3.1 Cost matrix construction 50
4.3.2 Minimizing costs: UCI repository 51
4.3.3 Unbalanced Data: Synapse and Mitochondria segmentation 53

5 Conclusions **57**
5.1 Future work 58
5.1.1 New theoretical scopes 58
5.1.2 Other scopes of supervised learning 59

A Proofs **61**
A.1 Proof of expression (3.3) 61
A.2 Proof of Lemma 1 61
A.3 Proof of Lemma 2 63
A.4 Proof of Corollary 1 64
A.5 Proof of Corollary 2 65
A.6 Proof of Corollary 3 66

List of Figures

3.1 Values of the Exponential Loss Function over margins, z, for a classification problem with 4-classes. Possible margin values are obtained taking into account the expression (3.5) for $s = 1$ and $s = 2$. 28

3.2 Margin vectors for a problem with three classes. Left figure presents the set of vectors Y. Right plot presents the set $\hat{Y}$. 31

3.3 Plots comparing the performances of Boosting algorithms. In the vertical axis we display the error rate. In the horizontal axis we display the number of weak-learners fitted for each algorithm. 38

3.4 Diagram of the Nemenyi test. The average rank for each method is marked on the segment. We show critical differences for both $\alpha = 0.05$ and $\alpha = 0.1$ significance level at the top. We group with thick blue line algorithms with no significantly different performance. 39

3.5 Plot comparing the performances of Boosting algorithms for the *Amazon* data base. In the vertical axis we display the error rate. In the horizontal axis we display the number of weak-learners fitted for each algorithm. 40

4.1 Comparison of ranks through the Bonferroni-Dunn test. BAdaCost's average rank is taken as reference. Algorithms significantly worse than our method for a significance level of 0.10 are unified with a blue line. 53

4.2 Example of a segmented image. In **b)**, green pixels belong to mitochondria while red ones belong to synapses. Figure **c)** indicates an estimation. 54

4.3 Brain images experiment with a heavily unbalanced data set. Training and testing error rates, along the iterations, for each algorithm are shown. 55

List of Tables

3.1 Cost Matrix associated to a PIBoost's separator of a set S with $s = |S|$ classes. 32

3.2 An example of encoding matrix for PIBoost's weak learners when $K = 4$ and $\mathcal{G} = \{\text{all single labels}\} \cup \{\text{all pairs of labels}\}$. 33

3.3 Comparison of the main properties of ECOC-based algorithms and PIBoost. $\mu_m(l)$ denotes the coloring function $\mu_m : L \to \{\pm 1\}$ at iteration m. R denotes the length of "code-words". In AdaBoost.OC, $\bar{I}_m(\mathbf{x})$ indicates $I\left(h_m(\mathbf{x}) = \mu_m(l)\right)$. 34

3.4 Summary of selected UCI data sets 36

3.5 Number of iterations considered for each Boosting algorithm. The first column displays the data base name with the number of classes in parenthesis. Columns two to six display the number of iterations of each algorithm. For PIBoost(2) the number of separators per iteration appears inside brackets. The last column displays the number of weak-learners used for each data base. 37

3.6 Error rates of GentleBoost, AdaBoost.MH, SAMME, PIBoost(1) and PIBoost(2) algorithms for each data set in table 3.4. Standard deviations appear inside parentheses in 10^{-4} scale. Bold values represent the best result achieved for each data base. 40

3.7 *P*-values corresponding to Wilcoxon matched-pairs signed-ranks test. 40

4.1 Summary of selected UCI data sets 52

4.2 Classification cost rates of Ada.C2M1, MultiBoost, L_p-CSB, and BAdaCost algorithms for each data set after 100 iterations. Standard deviations appear inside parentheses in 10^{-4} scale. Bold values represent the best result achieved for each data base. 53

4.3 Error rates of the five algorithms after the last iteration. 55

Chapter 1

Introduction

The emergence of the personal computer and its increasing computing power has brought a new scope of applications to our lives. The more computational power the closer the chance to transfer human reasoning to artificial environments. This issue derived inevitably on a field of Science: *Artificial Intelligence* (AI). There are many disciplines included in AI such as Robotic Systems, Mecatronics, Ontology, Bio-Informatics, Computer Vision, Data Mining, Pattern Recognition, Machine Learning, etc. The last three have been guided by statistical improvements and have received an enormous degree of attention in the last three decades. The contents of this dissertation are set within the context of Machine Learning and Pattern Recognition. Both disciplines will be considered as synonymous hereafter.

When introducing Machine Learning a first question always arises: what should be understood by *learning*? This is an extraordinary complex question, almost as difficult as to define the concept of *intelligence*. Such endeavour could take up the whole thesis, so we will bear in mind the following simple definition: *learning is the act of acquiring new knowledge by synthesizing information*. It is surely an incomplete definition since for humans one should add the fact of modifying existing knowledge. Moreover, behaviours and skills are also included in the process of learning. In any case we find this definition good enough to fill the gap between human learning and Machine Learning. This field of AI encompasses situations where an automatic decision must be taken wisely taking into account some restrictions. In other words, its aim is "teaching machines" to take reasonable decisions. We will pay attention only to this perspective of learning: situations where one has to select the right option based on a set of restrictions. This understanding of choosing "right decisions" leads to the concept of *classification problem*. Roughly speaking, we could say that the structure of a classification problem is as simple as follows. Choose the "best" answer from a finite set of discrete options, based on some input data and some sample choices.

Several well established techniques, grounded in different statistical foundations, have been proposed to solve it. Some examples are: Support Vector Machines, Bayesian Networks, Neural Networks, and ensemble algorithms. The latter being the scope of Boosting, the topic of this thesis.

1.1 Motivation

Boosting algorithms are learning schemes that produce an accurate or *strong classifier* by combining a set of simple base prediction rules or *weak-learners*. Their popularity is based not only on the fact that it is often much easier to devise a simple but inaccurate prediction rule than building a highly accurate classifier, but also because of the successful practical results and good theoretical properties of the algorithms. This philosophy of learning has found applications in many fields of Science, but specially in Computer Vision where there is a plethora of works grounded in them [16, 90, 42, 11, 54]. The most well-known Boosting algorithm, AdaBoost, was introduced in the context of two-class (binary) classification. It works in an iterative way. First a weight distribution is defined over the training set. Then, at each iteration, the best weak-learner according to the weight distribution is selected and combined with the previously selected weak-learners to form the strong classifier. Weights are updated to decrease the importance of correctly classified samples, so the algorithm tends to concentrate on the "difficult" examples.

Classification problems with more than two possible labels (multi-class problems) are very common in practice. When the nature of a task of this kind suggests decomposing it into binary subproblems, then it may be quite convenient to use Boosting [80, 78, 17, 3, 35, 95]. Although many Boosting algorithms have been proposed to address this issue [80, 78, 35, 3], none of them can be considered a canonical generalization of the original AdaBoost [26] in the sense of evaluating binary information under a multi-class loss function. In this thesis we provide a canonical extension of AdaBoost based on the well known statistical interpretation of Boosting [29]. The resulting method links to a common sense pattern, namely "the fact of discarding options in order to select the correct one". A pattern that, in turn, brings this computational process closer to human reasoning.

Furthermore, the multi-class classification problem at hand may present different costs for misclassifications. Usually this information can be encoded using a cost matrix. The inclusion of a cost matrix can be considered either the goal of the problem, if we are interested in optimizing a cost-based loss function, or a tool for other purposes. In the second case, it turns out helpful for addressing problems with label-dependent unbalanced data, ordinal problems[1] or problems with hard-to-fit decision boundaries. Cost-sensitive classification has been exploited for problems with two labels [94, 93, 60, 23, 33, 50, 89, 86, 63], but not much research has been devoted to problems with more than two [58, 102, 86]. This lack of study motivated us to develop a Boosting algorithm that encloses the essence of the most relevant generalizations of AdaBoost to both multi-class and cost-sensitive perspectives.

The algorithms we introduce in this thesis, *PIBoost* and *BAdaCost*, address respectively the problems of multi-class Boosting and multi-class Boosting with costs. They may be used to solve many relevant problems such as multi-class and minimum risk classification, object detection, image segmentation, etc.

1.2 Main Contributions

In the present thesis we focus on the concept of margin as cornerstone for the exponential loss function. The core message of the dissertation is described below:

[1] Multi-class problems coming from the discretization of a continuous variable.

Hypothesis: *The use of suitable margin vectors yields multi-class Boosting algorithms able to manage binary information or costs in a canonical way via the exponential loss function.*

This idea will be developed properly throughout the chapters. Here we summarize our main contributions:

- We propose two definitions of margin vectors. Firstly we introduce margin vectors suited for multi-class problems based on binary information. Secondly our proposal deals with cost-sensitive problems. Given a cost matrix for a multi-class classification problem we introduce another set of margin vectors. Both types of vectors are introduced jointly with an exponential loss function.

- Based on the first extension of margin, we introduce a new multi-class Boosting algorithm, *PIBoost* [24]. We decompose the classification problem into binary sub-problems that are evaluated with a multi-class loss function. The margin values obtained with the new encoding produce a broad range of penalties and rewards. The stagewise optimization of this model introduces an asymmetric Boosting procedure whose costs depend on the number of classes separated by each weak-learner. We think of PIBoost as a canonical extension of AdaBoost to the multi-class case using binary weak learners. With our proposal we schedule a framework that allows managing complex problems by translating them into the binary case in the purest Boosting fashion.

- Based on the second extension of margin we introduce a cost-sensitive multi-class Boosting algorithm, *BAdaCost*. We consider BAdaCost a canonical generalization of AdaBoost to the multi-class cost-sensitive case using multi-class weak learners. In fact we generalize SAMME [112] and Cost-sensitive AdaBoost [63], two direct extensions of AdaBoost for multi-class and binary cost-sensitive problems respectively. These results are discussed in Chapter 4.

1.3 Structure of the Thesis

The remainder of the thesis is organized as follows:

- **Chapter 2** introduces Boosting. Firstly we describe some basics on Machine Learning and then we present the AdaBoost algorithm from the two most common perspectives of the process, namely: *error-bounding* and *statistical*. We include a summary of multi-class Boosting methods, highlighting those derived upon the concept of margin, followed by a summary of binary cost-sensitive Boosting algorithms. Other interesting perspectives of Boosting are also included.

- **Chapter 3** presents our algorithm PIBoost for multi-class problems. Here we introduce the new set of margin vectors on which we ground our algorithm. We compare PIBoost with previous Boosting algorithms based on binary information. Experiments showing the performance of PIBoost against other relevant Boosting algorithms are included.

- **Chapter 4** describes our multi-class cost-sensitive algorithm BAdaCost. Again, we introduce a set of margin vectors from which we derive our algorithm. We explain BAdaCost's

structure in detail and compare it with previous relevant works in the area. Finally a set of experiments shows the applicability of BAdaCost under different perspectives.

- **Chapter 5** summarizes the most relevant conclusions of our work jointly with new lines of ongoing research.
- In the **Appendix** at the end of the document we provide the proofs of the results in the thesis.

Chapter 2

Background on Boosting

Most learning paradigms are based on acquiring a single model that yields a powerful classifier after training. During this process it is usual to require high computational cost to produce accurate results (SVM, Bayesian Networks, Deep Learning, etc.). If lucky, the process can be parallelized and then the computational burden is distributed.

When such capabilities are not at hand it would be desirable to have some mechanism able to produce good classifications based on a combination of *not so good*, but usually faster, ones. In most cases we can build simple rules of thumb, called weak predictions, that partially solve the problem by analyzing a small group of predictive variables. Hence, we may wonder: is it possible to use weak predictions to build a powerful classifier in a step-wise fashion? Besides, can we improve it by learning from previous mistakes? If so, can we do it without losing previous gains in accuracy? The Boosting philosophy of learning answers these questions.

This chapter serves as guide to introduce Boosting and some of its most important extensions. It is organized as follows. In a first section we review some basics related to the scope of supervised learning. In particular, we describe our problems of interest and the concept of meta-classifier. Section 2.2 is devoted to introduce the origin of Boosting jointly with AdaBoost, the most important algorithm in the field. A comprehensive description of the method is carried out, paying special attention to its statistical interpretation. In section 2.3 we review the most relevant extensions of AdaBoost to multi-class problems. In this case we will focus on those developed upon the concept of *margin*. Section 2.4 is devoted to review the most relevant generalizations proposed to tackle cost-sensitive problems. Finally, a short summary of other interesting topics covered by Boosting closes the chapter in section 2.5.

2.1 Background on Machine Learning

There is a glossary of primal concepts that should be known beforehand to frame Boosting and our algorithms conveniently. Let us describe these elements jointly with our terminology for classification problems. Firstly, we will assume without loss of generality that the domain, X, of *predictive variables* (features) of a problem is a subset of $\mathbb{R}^D$. In general X is not required to be a vectorial subspace of $\mathbb{R}^D$. In fact, depending on the nature of the problem, some operations may not even make sense. Nonetheless, it is usual for X to be endowed with structure of vectorial space. Once the variables are given, we call *dimensionality* to the number of them, i.e. the minimum value D for which $X \subseteq \mathbb{R}^D$. The term *class* (objective variable) will refer

to the qualitative or numeric feature that is the goal of the prediction. In the following we will take into account problems where the class variable can be encoded with a finite number of values, called *labels*. Hence, regression problems are out of the scope of the thesis. The set of labels, L, should be *exhaustive*. In other words, one should find in data each and every of these values and no one else is needed. We will assume that the *mutually-exclusive* property is satisfied, which implies that labels are well defined, i.e. no semantic overlapping should arise. An *instance*, $(\mathbf{x}, l)$, is hoped to have just one label. In subsection 2.1.1, we will comment a classification problem that does not meet this property. In the following, there will be times where the concepts of class and label will be used with the same meaning of the latter. In those situations the context will yield no ambiguities. Finally, a classifier (learner, hypothesis) will be any function $H : X \to L$ that assigns a label (and only one) to each instance $\mathbf{x} \in X$. In some cases the set L could be replaced by $\mathcal{P}(L)$, parts of L, or $\{S, S^C\}$ for a subset $S \subset L$.

Besides, the broad set of methods included in the world of Machine Learning can be divided into many categories according to different perspectives. Here we summarize some of them, emphasizing the area of our interest. Based on the availability of knowledge about the objective variable we can set a first division: *Unsupervised learning* (cluster analysis), when there is no knowledge about labels in data; *Semi-supervised Learning*: when some instances are unlabelled; *Supervised Learning*, when there is complete knowledge about the class in data. The algorithms we present lie in the latter.

A second important distinction comes from the necessity of estimating a priori probabilities to apply a method. Given an observation, $\mathbf{x}$, the Bayes' rule for predicting a label l is $P(l|\mathbf{x}) = P(\mathbf{x}|l)P(l)P(\mathbf{x})^{-1}$, which in turn is proportional to $P(\mathbf{x}|l)P(l)$. One can estimate these values either directly or using the above formula, that implies estimating $P(\mathbf{x}|l)$ and $P(l)$. This second option needs to fit a probability distribution for each label and to estimate the a priori probabilities. Algorithms tackling the classification problem in this fashion are called *generative* while those that fit $P(l|\mathbf{x})$ directly are called *discriminative*, which is the case of Boosting algorithms.

An additional classification can be established depending on the theoretical nature. It is usual to call *paradigm* to every approach that takes basis on a different conception of the learning process. They have particular motivations: bio-inspired, probabilistic, geometrical, "case-based", etc. Popular paradigms are: Neural Networks, Lazy Learners (K-NN and its variants), Bayesian Classifiers (Naive Bayes, TAN, etc), Support Vector Machines, Logistic Regression, Classification Trees and Rule-based algorithms. Some of these methodologies are supported by theories about universal approximations, i.e. functions that theoretically can approach any objective function up to a specified degree of error. Boosting algorithms neither are a paradigm per se nor can be included in any paradigm. Rather they are settled in the group of *meta-classifiers* or, as better known today, *ensemble learning*.

Let us describe some essential ideas about meta-classifiers. For a complete introduction see L. Kuncheva's book [47]. Roughly speaking, we define a meta-classifier as any *algorithm that builds a classification rule based on the results provided by various classifiers*. Following an initial intuition, one should expect the classifiers to be heterogeneous in order to obtain a good final decision using an ensemble. It could be proved theoretically that this combination can be specially suited to reduce either the general classification error bias or its associated variance [7]. This heterogeneity does not imply poor predictions for the compounding learners. Moreover, it is desirable to have a reasonably good accuracy for each of them. It could also be expected to combine a large number of classifiers, thus an efficient type of base algorithms should be used. In the same way it would be desirable to guarantee, if possible, the management

of learners specialized in sub-domains of the data.

To commit a meta-classifier one can use the same data in every base classifier (which is the usual situation) or develop a particular classifier for every single variable and then derive a global decision. The latter is often the case when working with magnitudes from different contexts: weight, length, pressure, temperature, sound intensity, etc. Taking into account the structure of the meta-classifier, which will depend on the type of problem, we can organize the compounding classifiers in *line*, in *parallel* or following a *hierarchic structure*. Finally, meta-classifiers can also be divided into two categories depending on whether they use the same paradigm to generate base classifiers or different. We are interested in the first case. Important families of algorithms using the same base paradigm are: *Bagging* [8], *Randomness*-derived algorithms (for instance, *Random Forests* [9]), *Cascade-shaped classifiers* [100] and finally the ones derived applying *Boosting* [28, 29].

We complete the section describing the supervised problems of our interest. For more information about the above topics we recommend the books of J. Friedman et al. [30], R. Duda et al. [19], A. J. Izenman [41], C. M. Bishop [7], and I. H. Witten et al. [104].

2.1.1 Supervised Classification Problems

The supervised classification problems of our interest are divided into three groups: *binary*, *multi-class* and *multi-label*. Each group generalizes its predecessors. These classification problems will be supposed free of costs. Although the algorithms proposed in this thesis are conceived for multi-class problems, we include the multi-label case bearing in mind the strong connections it has with our multi-class proposal and other previous works on multi-class classification.

In first place, binary classification problems are the simplest. The class variable presents two possible labels, usually denoted by $L = \{-1, 1\}$, $\{1, 0\}$ or $L = \{1, 2\}$.

Secondly, multi-class problems differ from the previous case in the fact that L has more than two elements, usually denoted with natural numbers $L = \{1, 2, \ldots, K\}$. Although its name suggests the opposite in this kind of classification problems there is only one class variable.

Finally, multi-labels problems have also a finite set of labels $L = \{1, 2, \ldots, K\}$, but every instance $\mathbf{x}$ has an associate subset $L_{\mathbf{x}} \subseteq L$ of labels, which justifies the name. In other words, instances can present more than a single label. Thus, this kind of classification is accomplished through a function $H : X \to \mathcal{P}(L)$, where $\mathcal{P}(L)$ is the *Label Powerset* of the problem. There may be instances for which no labels or all labels are assigned. Let us consider the following illustrative example.

INSTANCE	LABEL1	LABEL2	LABEL3	LABEL4
1	X	X		
2				X
3	X		X	X
4		X	X	
5	X			

With the aim of simplifying multi-label problems we explain below a couple of strategies widely used. These processes are also considered to tackle multi-class problems, as we highlight in section 2.3.

1. *Binary Relevance*. The idea is to fit a binary classifier for each label (presence/absence) and then map the result on $\mathcal{P}(L)$. This implies transforming the original data set into K theoretically independent data sets. For example, Label 3 in the above table would become:

INSTANCE	LABEL3
1	0
2	0
3	1
4	1
5	0

2. *Extending the data base*. The idea behind this transformation is simple. Every instance is repeated K times and then a new feature, with each of the possible labels, is added. In other words, each repetition has the shape "feature-label". Then the new class variable becomes binary just by encoding the presence/absence of a label with $+1/-1$ respectively. Thus there will be an amount of $N \cdot K$ observations of the kind $(\mathbf{x}_n, j, \pm 1)$; with $1 \leq n \leq N$ and $1 \leq j \leq K$. The first instance in the example gives rise to:

INSTANCE	LABEL	PRESENCE
1	1	1
1	2	1
1	3	-1
1	4	-1

2.2 Binary Boosting: AdaBoost

Let us introduce this methodology starting from its origin. Boosting is a technique proposed by R. Schapire [77] at the end of the eighties. The main idea in his work was to commit the efficiency of many poor classifiers (*weak learners*) into a unique powerful classifier (*strong learner*). Roughly speaking, we can define Boosting as *a general methodology of converting rough rules of thumb into a high quality prediction rule*.

Schapire's conception of the problem finds its basis on the work of L. Valiant [97] introduced in 1984 about the *probably approximately correct* model of learning (*PAC* learning). The implicit goal of this venue of research was to combine the computational power available at the time with the most recent theories about learnability. Computational learning theory was then conceived as a complex branch of Computer Science where statistics, inductive inference and information theory were mixed with a novel ingredient, the *computational complexity*. At that date, Valiant's work on PAC learning was considered an essential step for developing an adequate mathematical framework where both statistical and computational views of learning had place.

Successive works of M. Kearns and L. Valiant [44, 45] kept open the question of whether a weak learning algorithm (let us say, one with accuracy slightly better than random guessing) allows to be "boosted" to a strong learning algorithm (one with arbitrarily high accuracy). The most important theorem on this topic is due to Schapire [77] and came to solve the problem. His result states the equivalence between both types of learnabilities. Specifically, Schapire proved that *a class of target functions is strongly PAC-learnable if and only if it is weakly learnable*. It

is worth mentioning that in this genesis of Boosting there were no algorithms defined as such. Rather there were only theoretical results describing, under fixed conditions, how a classifier improves its accuracy after a reweigthing of instances (which is the essence of Boosting).

A few years later Y. Freund and R. Schapire [28] defined their seminal algorithm based on this reweighting technique, *AdaBoost*, which stands for *Adaptive Boosting* and whose pseudo-code is shown in Algorithm 1. By far this is the most important and referenced Boosting algorithm in the literature. Originally designed to tackle binary problems, AdaBoost has been a cornerstone for subsequent derivations that reached for multi-class, multi-label, semi-supervised and cost-sensitive problems.

Algorithm 1 : AdaBoost

1- Initialize the weight vector $\mathbf{W}$ with uniform distribution $w(n) = 1/N\ ,\ n = 1, \ldots, N$.
2- For $m = 1$ to M:
(a) Fit a classifier $f_m(\mathbf{x})$ to the training data using weights $\mathbf{W}$.
(b) Compute weighted error: $Err_m = \sum_{n=1}^{N} w(n) I\left(f_m(\mathbf{x}_n) \neq y_n\right)$.
(c) Compute $\alpha_m = \frac{1}{2}\log\left((1 - Err_m)/Err_m\right)$.
(d) Update weights $w(n) \leftarrow w(n)\exp\left(-\alpha_m y_n f_m(\mathbf{x}_n)\right)\ ,\ n = 1, \ldots, N$.
(e) Re-normalize $\mathbf{W}$.
3- Output Final Classifier: $H(\mathbf{x}) = sign\left(\sum_{m=1}^{M} \alpha_m f_m(\mathbf{x})\right)$

Let X and $L = \{+1, -1\}$ be the domain and set of labels, respectively, of the problem at hand. Maintaining the classic notation for binary problems, we will denote real labels with y instead of l. AdaBoost creates a strong classifier, H, whose output is the $sign$ of a linear combination of weak classifiers:

$$F(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m f_m(\mathbf{x})\ . \tag{2.1}$$

In other words, $H(\mathbf{x}) = sign(F(\mathbf{x})) = sign(\sum_{m=1}^{M} \alpha_m f_m(\mathbf{x}))$, where $\alpha_m \in \mathbb{R}^+$ and $f_m : X \to L, \forall\ m$. Usually all of these single functions are expected to belong to a specific type of hypothesis $\mathcal{H}$ (for instance, classification trees with a restriction on the depth). The shape of the final classifier is exactly a weighted majority vote from a set of M classifiers. Since $F(\mathbf{x}) \in \mathbb{R}$, the value of its module $|F(\mathbf{x})|$ should be understood as a degree of confidence for classifying with $sign(F(\mathbf{x}))$. That is why the values provided by F are usually called *confidence-rated* predictions. In the language of Boosting every classifier f_m included in expression 2.1 is called a *weak learner* while the linear combination F is called the *committee*.

2.2.1 Understanding AdaBoost

Let us explain briefly how this iterative process works. For each iteration m of AdaBoost, a pair (f_m, α_m) is calculated and added to the model in a greedy[1] fashion. On the one hand f_m is fitted taking into account the weight associated to each instance in such a way that those with largest weights have priority of classification. Therefore, every weak classifier focusses on hard instances according to $\mathbf{W}$. On the other hand, the constant α_m is a positive value that measures the goodness of f_m. The larger the value the most reliable its associated classifier. Then appears

[1] Elements included in previous iterations remain unaffected.

the essential idea behind AdaBoost, the reweighting process carried out over instances. Before defining it we must introduce the *Exponential Loss Function*:

$$\mathcal{L}(y, F(\mathbf{x})) := \exp(-yF(\mathbf{x})) \,. \tag{2.2}$$

This function is a key concept for the development of Boosting algorithms as we will show in subsequent sections. For a certain labeled instance $(\mathbf{x}, y)$ and given a confidence rated classifier F, it computes the value $z := yF(\mathbf{x})$, usually called *Margin*, and then applies $\exp$ to $-z$. Hence the exponential loss function can be defined over margin values: $\mathcal{L}(z) = \exp(-z)$. It is clear that the more negative the value of margin the larger the loss incurred, while the more positive the closer to zero.

AdaBoost's reweighting process computes the values of this loss function when the pair (α_m, f_m) is added to the current model F_{m-1}, i.e.

$$w_m = \exp(-yF_m(\mathbf{x})) = \exp(-y(F_{m-1}(\mathbf{x}) + \alpha_m f_m(\mathbf{x}))) \,, \tag{2.3}$$

which in turn can be recursively depicted as follows:

$$w_m = \exp\left(-y\sum_{r=1}^{m}\alpha_r f_r(\mathbf{x})\right) = w_{m-1}\exp(-y\alpha_m f_m(\mathbf{x})) \,, \tag{2.4}$$

that is quite more compact and efficient. Additionally, the weight vector is normalized, $\mathbf{W} = \mathbf{W}/Z_m$, where $Z_m = \sum_{n=1}^{N} w_m(n)$ is the normalization constant. These weights will be taken into account in the next iteration. Specifically, the objective of the next classifier f is to minimize the weighted error, i.e. $Err(f) = \sum_{n=1}^{N} w(n)I(f(\mathbf{x}_n) \neq y_n)$. So, when computing it, there are two ways of using $\mathbf{W}$:

1. *Directly*: If the set of hypothesis $\mathcal{H}$ allows weights in its computations.
2. *By sampling*: Since $\mathbf{W}$ is a distribution of probability on data, it can be used for sampling data from which we can learn a weak classifier from $\mathcal{H}$.

In both cases, every new weak learner is forced to focus on hard instances (those poorly classifed in previous rounds). Once the result of its weighted error Err is known, the value of α associated to this classifier is

$$\alpha = \frac{1}{2}\log\left(\frac{1 - Err}{Err}\right) \,. \tag{2.5}$$

The process can be repeated as many times as iterations M are specified.

Schapire's proposal kept a theoretical justification based on an uniform bounding of the classification error. Specifically, if we denote $Err_m = 1/2 - \gamma_m$ the classification error of the weak learner in iteration m, then the training error of the classifier globally fitted, H, is bounded in the following way:

$$Err(H) \leq \prod_{m=1}^{M} 2\sqrt{Err_m(1 - Err_m)} \leq \prod_{m=1}^{M}\sqrt{1 - 4\gamma_m^2} \leq \exp\left(-2\sum_{m=1}^{M}\gamma_m^2\right) . \tag{2.6}$$

Thus, for a constant γ satisfying $0 < \gamma \leq \gamma_m$, $\forall\ m$, we get:

$$Err(H) \leq \exp(-2M\gamma^2) \,, \tag{2.7}$$

which tends to zero as the number of iterations M increases. Expression (2.7) justifies the part *Adaptive* in AdaBoost, since γ and M do not need to be given for the algorithm to adapt to the problem. Proving the first inequality in (2.6) is not difficult. One just has to realize that the $0|1$-error loss defined in terms of the margin, $Err_{0|1}(z) = (1 - sign(z))/2$, is upperly bounded by the exponential $\mathcal{L}(z) = \exp(-z)$. Therefore

$$Err(H) = \frac{1}{N}\sum_{n=1}^{N} I(y_n \neq H(\mathbf{x}_n)) = \frac{1}{N}\sum_{n=1}^{N} I(y_n F(\mathbf{x}_n) < 0)$$

$$\leq \frac{1}{N}\sum_{n=1}^{N} \exp(-y_n F(\mathbf{x}_n)) = \sum_{n=1}^{N} w_M(n) \cdot \prod_{m=1}^{M} Z_m = \prod_{m=1}^{M} Z_m \, .$$

Where the second last equality follows from substituting (2.5) in the normalization factor

$$Z_m = \sum_{n=1}^{N} w_{m-1}(n) \exp(-\alpha_m y_n f_m(\mathbf{x}_n)).$$

A more complex reasoning has to be applied to describe a bound for the generalization error. This analysis was carried out by Y. Freund and R. Schapire [28]. They justified that

$$GeneralErr(H) \leq \hat{P}(z \leq \sigma) + \tilde{O}\left(\frac{\sqrt{D_{VC}/N}}{\sigma}\right) \; ; \tag{2.8}$$

where: $\hat{P}$ is the empirical probability, N is the number of instances, and D_{VC} is the Vapnik-Chervonenkis dimension of the problem. Later, Schapire et al. [79] proposed a better bound in terms of the concept of margin but independent from M. In this context, margins are exactly a "measure of confidence" in the prediction (their sign). Specifically he proved that AdaBoost tends to increase the margin of the training data and, as consequence, the generalization error decreases.

It is worth mentioning that at the end of the nineties AdaBoost was becoming so popular, specially due to its surprising accuracy in high dimensions, that made other famous meta-classifiers like Leo Breiman's *Bagging* algorithm [8] take a second place. However, the beginning of the present century saw the emergence of an important meta-classifier also derived by L. Breiman, the *Random Forests* [9], that have received a degree of research comparable to Boosting, specially for Computer Vision problems.

AdaBoost allows an alternative interpretation that became quite useful for justifying its good properties. This different point of view has a statistical background that may be more understandable to newcomers. Next subsection is devoted to it.

2.2.2 Statistical View of Boosting

Since its beginning, Boosting has always been object of study because of its apparently resistance to overfiting. It was never well understood how such an iterative process improves its generalization error iteration after iteration even when its training error is zero. But at the end of the nineties Friedman, Hastie and Tibshirani's work [29] came to shed some light on the matter, although their arguments were not completely satisfactory [66]. In their work the authors

proved that AdaBoost can be obtained by fitting an additive model [37], i.e. a model with the shape of expression (2.1), whose goal is to minimize the expected exponential loss.

Specifically they proved that Schapire's method builds an additive logistic regression via Newton-like updates for minimizing such expected loss. They also derived the *Real AdaBoost*, an analogous version of AdaBoost for confidence-rated weak learners free of constants α. Furthermore, they propose *LogitBoost* and *GentleBoost*, two algorithms that resort to Newton steps for updating the additive model with regard to the binomial log-likelihood and the exponential loss, respectively.

Let us explain briefly how the pair (α_m, f_m) is estimated in the m-th iteration under this statistical point of view. Firstly, the optimal parameters have to satisfy

$$(\alpha_m, f_m) = \arg\min_{\alpha,f} \sum_{n=1}^{N} \exp\left[-y_n(F_{m-1}(\mathbf{x}_n) + \alpha f(\mathbf{x}_n))\right] \; ; \tag{2.9}$$

that, again, can be written in terms of weigths $w_{m-1}(n) = \exp(-y_n F_{m-1}(\mathbf{x}_n))$ as follows

$$(\alpha_m, f_m) = \arg\min_{\alpha,f} \sum_{n=1}^{N} w_{m-1}(n) \exp\left(-\alpha y_n f(\mathbf{x}_n)\right) \; . \tag{2.10}$$

Now we can calculate each parameter separately. Let $\alpha > 0$ be a pre-specified value, to characterize f we just have to rewrite the objective function conveniently:

$$\sum_{n=1}^{N} w_{m-1}(n) \exp\left(-\alpha y_n f(\mathbf{x}_n)\right) =$$

$$e^{-\alpha} \sum_{f(\mathbf{x}_n)=y_n}^{N} w_{m-1}(n) + e^{\alpha} \sum_{f(\mathbf{x}_n)\neq y_n}^{N} w_{m-1}(n) =$$

$$\left(e^{\alpha} - e^{-\alpha}\right) \sum_{n=1}^{N} w_{m-1}(n) I(f(\mathbf{x}_n) \neq y_n) + e^{-\alpha} \underbrace{\sum_{n=1}^{N} w_{m-1}(n)}_{1} \; .$$

Therefore, independently of α, the optimal weak learner minimizes the weighted error. Now, let us assume f_m be known (hence its weighted error Err too). The above expression becomes

$$\left(e^{\alpha} - e^{-\alpha}\right) Err + e^{-\alpha} \, , \tag{2.11}$$

that is a convex function w.r.t α. So differentiating and equating to zero one obtains

$$\alpha_m = \frac{1}{2} \log\left(\frac{1 - Err}{Err}\right) \, , \tag{2.12}$$

exactly the same value of (2.5).

We will bear in mind this for our derivations. It will serve to introduce our definition of *canonical extension* for Boosting algorithms. Let $\mathbf{P}_1$ and $\mathbf{P}_2$ be two types of classification problems in such a way that $\mathbf{P}_1$ becomes $\mathbf{P}_2$ when a set of restrictions is satisfied. Assume that $\mathcal{A}$ and $\mathcal{B}$ are Boosting algorithms developed to solve $\mathbf{P}_1$ and $\mathbf{P}_2$, respectively. Assume also that both algorithms are derived from different loss functions following the statistical interpretation of Boosting. We define the concept of canonical extension (generalization) as follows:

"Let ($\mathcal{A}$,$\mathcal{B}$) be a pair of Boosting algorithms defined as above. $\mathcal{A}$ is said to extend canonically $\mathcal{B}$ if the loss function in $\mathcal{A}$ restricted to the framework of $\mathcal{B}$ leads to update the additive model with the same elements ($G_m(\mathbf{x})$,α_m) fitted following $\mathcal{B}$. In other words, the restrictions in $\mathcal{A}$ given by the second framework yields the algorithm $\mathcal{B}$".

2.3 Multi-class Boosting

Almost jointly with the emergence of AdaBoost new extensions to multi-class problems were proposed. There is a large number of Boosting algorithms dealing with this type of classification. For ease of cataloging, we divide them into two groups: algorithms that *decompose the problem into binary subproblems* (thus, using binary weak learners) and algorithms that *work simultaneously with all the labels* (using multi-class weak learners or computing a posteriori probabilities at the same iteration). The following subsections discuss each approach. The Boosting algorithm that we introduce in the next chapter may be positioned between both groups.

Hereafter, we will maintain the notation of section 2.1.1 for multi-class problems. The set of labels will be $L = \{1, 2, \ldots, K\}$. Instances will be denoted again $(\mathbf{x}, l)$.

2.3.1 Boosting algorithms based on binary weak-learners

Completing Freund and Schapire's contribution [28] for binary problems, the authors proposed in [26] two extensions to multi-class problems. We start discussing the second one, AdaBoost.M2. This algorithm proceeds extending the datasets K-times, as discussed in section 2.1.1 for multi-label problems, and then computing binary weak classifiers of the shape $h : X \times L \to \{0, 1\}$. Following AdaBoost's essence, a normalized distribution of weights, $\mathbf{W}$, is used, but this time with a matrix shape, $\mathbf{W} \in [0, 1]^{N \times K}$. The main novelty of the method was the inclusion of errors based on a *pseudo-loss* function

$$\bar{E}rr(h) = \frac{1}{2} \sum_{(n,k) \in B} w(n, k)(1 - h(x_n, l_n) + h(x_n, k)) \,, \tag{2.13}$$

where $B = \{(n, k) | n = 1, \ldots, N \ ; \ k \neq l_n\}$. This type of losses penalizes simultaneously hard instances and hard labels. The pseudo-loss is combined with the binary exponential loss function to update weights, while α constants are computed substituting $\bar{E}rr$ in (2.5).

Years later, the idea behind AdaBoost.M2 was brought back by Shapire and Singer in [80], where three algorithms for multi-label problems were proposed. The first of them, AdaBoost.MH, enables a direct application to multi-class problems, becoming one of the most popular methods for this task among the proposed at the date. We show its pseudo-code in Algorithm 2. AdaBoost.MH addresses the multi-class problem extending the data base just as introduced in 2.1.1. The main difference between it and AdaBoost.M2 is the use of the Hamming loss for measuring errors instead of a pseudo-loss. Analogously AdaBoost.MR was also proposed, in this case using a ranking loss to measure multi-label accuracy.

The third algorithm in [80], AdaBoost.MO, is more complex than AdaBoost.MH since the set of labels, L, is mapped on $\mathcal{P}(\hat{L})$ for an undefined set of labels, $\hat{L}$, with R elements. Therefore, via $\lambda : L \to \mathcal{P}(\hat{L})$, each label l has an associated "codeword" $\lambda_l \in \{\pm 1\}^R$ in such

Algorithm 2 : AdaBoost.MH

1- Initialize the weight matrix $\mathbf{W}$ with uniform distribution $w(n,k) = 1/KN$, for $n = 1,\ldots,N; k = 1,\ldots K$.
2- For $m = 1$ to M:
(a) Fit binary classifier $h_m : X \times L \rightarrow \{-1,+1\}$ using weights $\mathbf{W}$.
(b) Compute α_m.
(c) Update weight vector $w(n,l) \leftarrow w(n,l)\exp\left(-\alpha_m y_l h_m(\mathbf{x}_n,l)\right)$, for $n = 1,\ldots,N; l = 1,\ldots K$.
(d) Re-normalize $\mathbf{W}$.
3- Output Final Classifier:
Multi-label. $H(\mathbf{x},l) = sign\left(\sum_{m=1}^{M}\alpha_m h_m(\mathbf{x},l)\right)$.
Multi-class. $H(\mathbf{x}) = \arg\max_k \sum_{m=1}^{M}\alpha_m h_m(\mathbf{x},k)$.

a way that a binary classification can be performed on the extended set $\{(\mathbf{x}_n,\hat{l})|\ \hat{l} = 1,\ldots,R\}$ just as stated for AdaBoost.MH. AdaBoost.MO belongs to a new stream of multi-class Boosting based on the *Error Correcting Output Codes* philosophy (ECOC). Let us describe briefly the main ideas behind this perspective of learning.

The ECOC methodology was introduced by Dietterich and Bakiri [17] in the nineties. It served as an alternative approach to reduce a multi-class problem to a set of R binary ones. The key point of this approach lies in using a particular encoding to represent the subset of labels object of classification. So, considered the r-th task, one generates a weak-learner $h_r : X \rightarrow \{+1,-1\}$. The presence/absence of a group of labels on an instance is encoded by a column vector belonging to $\{-1,+1\}^K$, where $+1$ indicates presence of the objective labels of h_r. It is usual to use a *coloring* function, μ, for assigning the presence or absence of a set of labels in the data. The resulting assignment becomes the real labels for its associated binary subproblem. The composition of all column vectors generated in this fashion produces a $(K \times R)$-codification matrix, in which the l-th row serves as a *codeword* associated to label l. When classifying a new instance one has to compute all the answers of the weak learners and compare the resulting $(1 \times R)$-vector with each codeword. The decision rule consists in choosing the nearest class to the result (under certain measure like the Hamming distance). This solution justifies matrix encodings as a practical and intuitive tool for building strong multi-class classifiers using binary weak learners.

Based on this idea, and jointly with AdaBoost.MO [80], two new Boosting algorithms appeared, AdaBoost.OC [78] and AdaBoost.ECC [35]. These two algorithms are probably the most relevant ones for this insight of multi-class Boosting. We show their respective pseudo-codes in Algorithm 3 and 4. We must point out that AdaBoost.OC is also grounded on the M2-version proposed by Schapire, but this time completed with an ECOC perspective.

Here we briefly compare these three ECOC-based methods:

- With regard to the *instance weights*, they have in common a normalized matrix $\mathbf{W} \in \mathbb{R}^{N\times R}$ used at each iteration. In particular, for AdaBoost.OC and AdaBoost.ECC, $R = K$ and the actual weights, $\mathbf{D} \in \mathbb{R}^N$, for fitting weak learners are computed based on $\mathbf{W}$ and the m-th selected coloring, μ_m.

- Besides this, the three algorithms add just one *voting constant*, α, per iteration to accompany the decisions of its respective weak learner. In AdaBoost.ECC the constant $g_m(\mathbf{x}_n)$

Algorithm 3 : AdaBoost.OC

1- Initialize the weight matrix $\mathbf{W}$ with uniform distribution $w(n,k) = 1/N(K-1)$, if $k \neq l_n$, and $w(n,l_n) = 0\ ,\ n = 1,\ldots,N$.
2- For $m = 1$ to M:
(a) Compute coloring function $\mu_m : L \to \{0,1\}$.
(b) Compute $D_m(n) = Z_m^{-1} \sum_{k\in L} w(n,k) I(\mu_m(l_n) \neq \mu_m(k))$,
where Z_m is a normalization constant.
(c) Fit a classifier $h_m : X \to \{0,1\}$ to the data $\{(\mathbf{x}, \mu_m(l))\}$ weighted according to $\mathbf{D}_m$.
(d) Define $\tilde{h}_m(\mathbf{x}_n)) := \{k \in L : h_m(\mathbf{x}_n) = \mu_m(k)\}$.
(e) Compute $Err_m = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K} w(n,k)(\ I(l_n \notin \tilde{h}_m(\mathbf{x}_n)) + I(k \in \tilde{h}_m(\mathbf{x}_n))\ )$,
(f) Compute $\alpha_m = \frac{1}{2}\log\left((1-Err_m)/Err_m\right)$.
(g) Update weights $w(n,k) \leftarrow w(n,k)\exp\left[\alpha_m(\ I(l_n \notin \tilde{h}_m(\mathbf{x}_n)) + I(k \in \tilde{h}_m(\mathbf{x}_n))\ )\right]$.
(h) Re-normalize $\mathbf{W}$.
3- Output Final Classifier: $H(\mathbf{x}) = \arg\max_k \sum_{m=1}^{M} \alpha_m I\left(h_m(\mathbf{x}) = \mu_m(l)\right)$.

supports the possibility of two instance-dependent values:

$$\begin{aligned}
\alpha_m &= \frac{1}{2}\ln\left(\frac{\sum_{n:h_m(\mathbf{x}_n)=\mu_m(l_n)=1} D_m(n)}{\sum_{n:h_m(\mathbf{x}_n)=1\neq\mu_m(l_n)} D_m(n)}\right)\ ,\ \text{if } h_m(\mathbf{x}_n) = +1\ ; \\
\beta_m &= \frac{1}{2}\ln\left(\frac{\sum_{n:h_m(\mathbf{x}_n)=\mu_m(l_n)=-1} D_m(n)}{\sum_{n:h_m(\mathbf{x}_n)=-1\neq\mu_m(l_n)} D_m(n)}\right)\ ,\ \text{if } h_m(\mathbf{x}_n) = -1\ .
\end{aligned} \tag{2.14}$$

- The *loss function* applied for updating weights in all three algorithms comes from a derivation of the binary exponential loss function with non-vectorial arguments (for instance, a pseudo-loss in the case of AdaBoost.OC).

- Another important issue is the *shape of the final classifier*. On the one hand, the final decision for AdaBoost.OC and AdaBoost.ECC admits a translation into a K-dimensional function $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\ldots,f_K(\mathbf{x}))^\top$ whose maximum coordinate is selected as response. On the other hand, AdaBoost.MO proposes two options based on the final response $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^R$ and the $(K \times R)$-matrix of codewords. One can select the row that measures the closest distance to the response or resort to a confidence-rated prediction such as:

$$\arg\min_{l} \sum_{\hat{l}\in\hat{L}} \exp\left(-\lambda_l(\hat{l}) f(\mathbf{x},\hat{l})\right)\ , \tag{2.15}$$

that is the option recommended by the authors.

- Finally, with regard to the *number of weak learners* computed at each iteration, the three algorithms compute just one weak learner (jointly with its voting constant α). AdaBoost.OC and AdaBoost.ECC train a weak classifier associated to the coloring μ_m, while AdaBoost.MO trains a weak learner for the binary problem generated by the extended data base $\{(\mathbf{x}_n,\hat{l})|\ \hat{l} = 1,\ldots,R\}$ with labels $\{\lambda_{l_n}(\hat{l})|\ \hat{l} = 1,\ldots,R\}$.

We will come back to these issues in next chapter, when introducing our multi-class algorithm.

Algorithm 4 : AdaBoost.ECC

1- Initialize the weight matrix $\mathbf{W}$ with uniform distribution $w(n,k) = 1/N(K-1)$, if $k \neq l_n$, and $w(n,l_n) = 0\ ,\ n = 1,\ldots,N$.
2- For $m = 1$ to M:
(a) Compute coloring function $\mu_m : L \to \{-1,+1\}$.
(b) Compute $D_m(n) = Z_m^{-1} \sum_{k \in L} w(n,k) I(\mu_m(l_n) \neq \mu_m(k))$,
where Z_m is a normalization constant.
(c) Fit a binary classifier $h_m : X \to \{-1,+1\}$ to the training data using weights $\mathbf{D}_m$.
(d) Compute α_m and β_m following (2.14).
(e) Compute $g_m(\mathbf{x}) = \alpha_m I(h_m(\mathbf{x}) = +1) - \beta_m I(h_m(\mathbf{x}) = -1)$.
(f) Update weights $w(n,k) \leftarrow w(n,k) \exp\left[\frac{1}{2} g_m(\mathbf{x}_n)(\mu_m(k) - \mu_m(l_n))\right]$,
(g) Re-normalize $\mathbf{W}$.
3- Output Final Classifier: $H(\mathbf{x}) = \arg\max_k \sum_{m=1}^{M} g_m(\mathbf{x}) \mu_m(k)$.

2.3.2 Boosting algorithms based on vectorial encoding

Grouped in a second block, we encompass those multi-class algorithm that manage all labels simultaneously at each iteration (using multi-class weak learners or estimating directly a posteriori probabilities). Here we include the first algorithm proposed by Freund and Schapire in [26], AdaBoost.M1. The essence of this multi-class generalization of AdaBoost lies in using pure multi-class weak learners while maintaining the same structure of the original algorithm. The main drawback in AdaBoost.M1 is the need for "strong learners", i.e. hypothesis that can achieve accuracy of at least 50%, a requirement that may be too strong when the number of labels is high. Despite the lack of theory supporting this method, we must clarify that it is very common to consider AdaBoost.M1 as the first Boosting algorithm. This is due to its direct translation to AdaBoost for the binary case. A second approach came with the multi-class version of LogitBoost appeared jointly with the binary one [29]. As its counterpart for two labels, it estimates separately the probability of belonging to each label based on a multi-logit parametrization.

The most interesting works under our point of view are those grounded on a vectorial insight. A successful way to generalize the symmetry of class-label representation in the binary case to the multi-class case is using a set of vector-valued class codes that represent the correspondence between the label set $L = \{1,\ldots,K\}$ and a collection of vectors $Y = \{\mathbf{y}_1,\ldots,\mathbf{y}_K\}$, where vector $\mathbf{y}_l$ has a value 1 in the l-th coordinate and $\frac{-1}{K-1}$ elsewhere. So, if $l = 1$, the code vector representing class 1 is $\mathbf{y}_1 = \left(1, \frac{-1}{K-1}, \ldots, \frac{-1}{K-1}\right)^\top$. It is immediate to see the equivalence between classifiers $H(\mathbf{x})$ defined over L and classifiers $\mathbf{f}(\mathbf{x})$ defined over Y:

$$H(\mathbf{x}) = l \in L \ \Leftrightarrow\ \mathbf{f}(\mathbf{x}) = \mathbf{y}_l \in Y\ . \tag{2.16}$$

In the remainder of the thesis we will use capital letters (H, G or T) for denoting classifiers with target set L. On the other hand, classifiers having as codomain a set of vectors, like Y, will be denoted with small bold letters ($\mathbf{f}$ or $\mathbf{g}$).

This codification was first introduced by Lee, Lin and Wahba [51] for extending the binary Support Vector Machine to the multi-class case. More recently H. Zou, J. Zhu and T. Hastie [114] generalized the concept of binary margin to the multi-class case using a related vectorial codification in which a K-vector $\mathbf{y}$ is said to be a *margin vector* if it satisfies the *sum-to-zero* condition $\sum_{k=1}^{K} y(k) = 0$. In other words, $\mathbf{y}^\top \mathbf{1} = 0$, where $\mathbf{1}$ denotes a vector

of ones. This sum-to-zero condition reflects the implicit nature of the response in classification problems in which each y_n takes one and only one value from a set of labels. Margin vectors are useful for multi-class classification problems for other reasons. One of them comes directly from the sum-to-zero property. It is known that, in general, every vectorial classifier $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x}))^\top$ has a direct translation into a posteriori probabilities $P(l = k \mid \mathbf{x}), \forall k \in L$, via the Multi-class Logistic Regression Function (*MLRF*)

$$P(l = k \mid \mathbf{x}) = \frac{\exp(f_k(\mathbf{x}))}{\sum_{i=1}^{K} \exp(f_i(\mathbf{x}))} \,. \tag{2.17}$$

It is clear that a function $\mathbf{f}(\mathbf{x})$ produces the same a posteriori probabilities than $\mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \alpha(\mathbf{x}) \cdot \mathbf{1}$, where $\alpha(\mathbf{x})$ is a real valued function and $\mathbf{1}$ is a K-dimensional vector of ones. Such is the case when, for example, $\alpha(\mathbf{x}) = -f_K(\mathbf{x})$. Using margin vectors we do not have to concern about this issue[2].

Using this codification, J. Zhu, H. Zou, S. Rosset and T. Hastie [112] generalized the original AdaBoost to multi-class problems under a statistical point of view. This work has been a cornerstone for subsequent derivations. Let us describe the main elements upon which the algorithm is grounded. Firstly, the binary margin applied in AdaBoost, $z = yf(\mathbf{x})$, is replaced by the *Multi-class Vectorial Margin*, that is defined as a scalar product

$$z := \mathbf{y}^\top \mathbf{f}(\mathbf{x}) \,. \tag{2.18}$$

The essence of the margin approach resides in maintaining negative/positive values of the margin when a classifier has respectively a failure/success. That is, if $\mathbf{y}, \mathbf{f}(\mathbf{x}) \in Y$; the margin $z = \mathbf{y}^\top \mathbf{f}(\mathbf{x})$ satisfies: $z > 0 \Leftrightarrow \mathbf{y} = \mathbf{f}(\mathbf{x})$, and $z < 0 \Leftrightarrow \mathbf{y} \neq \mathbf{f}(\mathbf{x})$. Note that, again, this definition of margin serves as a compact way to specify numerically hits and mistakes of classification. It is straightforward that the only two possible values of the margin when $\mathbf{y}, \mathbf{f}(\mathbf{x}) \in Y$ are:

$$z = \mathbf{y}^\top \mathbf{f}(\mathbf{x}) = \begin{cases} \frac{K}{(K-1)} & \text{if } \mathbf{f}(\mathbf{x}) = \mathbf{y} \,, \\ \frac{-K}{(K-1)^2} & \text{if } \mathbf{f}(\mathbf{x}) \neq \mathbf{y} \end{cases} \tag{2.19}$$

Bearing this definition in mind, the *Multi-class Exponential Loss Function* is:

$$\mathcal{L}\left(\mathbf{y}, \mathbf{f}(\mathbf{x})\right) := \exp\left(-\frac{1}{K}\mathbf{y}^\top \mathbf{f}(\mathbf{x})\right) = \exp\left(-\frac{1}{K} z\right) \,. \tag{2.20}$$

As the reader may guess, the presence of the constant $1/K$ is important but not determinant for the proper behaviour of the loss function. We will see later how it simplifies some calculations. An interesting property of this loss function (that justifies the addition of the constant $1/K$) comes from the following result:

$$\begin{aligned} \mathcal{L}\left(\mathbf{y}, \mathbf{f}(\mathbf{x})\right) &= \exp\left(-\frac{1}{K}\sum_{k=1}^{K} y(k) f_k(\mathbf{x})\right) = \left(\prod_{k=1}^{K} \exp\left(-y(k) f_k(\mathbf{x})\right)\right)^{1/K} \\ &= \sqrt[K]{\prod_{k=1}^{K} \exp\left(-y(k) f_k(\mathbf{x})\right)} \,. \end{aligned} \tag{2.21}$$

[2]Over the set of functions $\mathcal{F} = \left\{\mathbf{f} : \mathbf{x} \mapsto \mathbb{R}^K\right\}$ we can define the equivalence relation $\mathbf{f}(\mathbf{x}) \sim \mathbf{g}(\mathbf{x}) \Leftrightarrow \exists \alpha : \mathbf{x} \mapsto \mathbb{R} \mid \mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \alpha(\mathbf{x})\mathbf{1}$. Then margin functions become representatives of equivalence classes.

Hence this multi-class loss function is a *geometric mean* of the binary exponential loss function applied to each pair of coordinates of $(\mathbf{y}, \mathbf{f}(\mathbf{x}))$ (i.e. component-wise margins).

Moreover, this loss function is *Fisher-consistent* [114]. This property is defined as follows:

"A loss function $\mathcal{L}$ is said to be Fisher-consistent if, $\forall \mathbf{x} \in X$ (set of full measure), the optimization problem $\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} \mathcal{L}(P_{L|X}, \mathbf{f}(\mathbf{x}))$, with $\mathbf{f}$ belonging to the hyperplane of margin vectors, has an unique solution and, in addition, $\arg\max_j \hat{f}_j(\mathbf{x}) = \arg\max_j P(l = j|\mathbf{x})$".

Roughly speaking the Fisher-consistent condition says that if we were provided with infinite samples, we could recover the exact Bayes rule by minimizing losses of the kind. This fact makes them suitable for fitting multi-class classifiers with guarantees. H. Zou, J. Zhu and T. Hastie introduced a theoretical basis for margin vectors and Fisher-consistent loss functions [114]. Given a classification function expressed in terms of a margin vector $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x}))^\top$, they defined the multi-class margin for an instance $(\mathbf{x}, l)$ as the coordinate $f_l(\mathbf{x})$. Consequently, a binary loss function can be used for evaluating multi-class decisions. Although this generalization is adequate to derive algorithms like *AdaBoost.ML* and the *Multi-category GentleBoost* we believe this definition of margin does not exploit the utility of vectorial encodings for labels.

In the case of the multi-class exponential loss (2.20), it can be proved that the population minimizer:

$$\arg\min_{\mathbf{f}} E_{Y|X=\mathbf{x}} \left[\mathcal{L}\left(Y, \mathbf{f}(\mathbf{x})\right) \right] ,$$

corresponds to the multi-class Bayes optimal classification rule [112]:

$$\arg\max_k f_k(\mathbf{x}) = \arg\max_k P\left(Y = k|\mathbf{x}\right).$$

Other loss functions, such as the *logit* or L_2, share this property and may also be used for building Boosting algorithms. Similarly, Saberian and Vaconcelos justified that other set of margin vectors could have been used for representing labels [74], and therefore to develop equivalent algorithms. Their work also proposes an interesting definition of a multi-class margin for label k, $z_k := 1/2 \left[\mathbf{y}_k^\top \mathbf{f}(\mathbf{x}) - \max_{j \neq k} \mathbf{y}_j^\top \mathbf{f}(\mathbf{x}) \right]$. Using it, they justify the classification criteria $H(\mathbf{x}) = \arg\max_k z_k$.

Having defined the above concepts, namely *multi-class vectorial margin* and *multi-class exponential loss*, we can come back to the work of Zhu et al. [112]. Their proposed algorithm, SAMME[3] (Stage-wise Additive Modeling using a Multi-class Exponential loss function), resorts to the multi-class exponential loss for evaluating classifications encoded with margin vectors when real labels are encoded likewise. The expected loss is then minimized using a stage-wise additive gradient descent approach. Its pseudo-code is shown in Algorithm 5. It is quite interesting how the resulting algorithm only differs from AdaBoost (see both pseudo-codes) in step **2.**(c) that now is $\alpha_m = \log\left((1 - Err_m)/Err_m\right) + \log(K - 1)$. Step **3.** is not specially different since

$$H(\mathbf{x}) = \arg\max_k \sum_{m=1}^{M} \alpha_m I\left(T_m(\mathbf{x}) = k\right) = \arg\max_k f_k(\mathbf{x}) ,$$

where $\mathbf{f}(x) = \sum_{m=1}^{M} \alpha_m \mathbf{g}_m(\mathbf{x})$. Moreover, it is an easy exercise to prove that the above classification rule is equivalent to assigning the maximum margin (2.18), $H(\mathbf{x}) = \arg\max_k \mathbf{y}_k^\top \mathbf{f}(\mathbf{x})$,

[3] A curious name for an algorithm that is essentially the *same* as the AdaBoost.*ME* proposed in a technical report [113] developed previously by the authors of [114] following a different margin-based insight.

which links with the perspective defined in [74]. In the same way it is straightforward to verify that AdaBoost becomes a particular case when $K = 2$, what makes us think that SAMME is a canonical generalization of AdaBoost using multi-class weak-learners.

Algorithm 5 : SAMME

1- Initialize the weight vector $\mathbf{W}$ with uniform distribution $w(n) = 1/N\ ,\ n = 1, \ldots, N$.
2- For $m = 1$ to M:
(a) Fit a multi-class classifier $T_m(\mathbf{x})$ to the training data using weights $\mathbf{W}$.
(b) Compute weighted error: $Err_m = \sum_{n=1}^{N} w(n) I\left(T_m(\mathbf{x}_n) \neq l_n\right)$.
(c) Compute $\alpha_m = \log\left((1 - Err_m)/Err_m\right) + \log(K - 1)$.
(d) Update weight vector $w(n) \leftarrow w(n) \exp\left(\alpha_m I\left(T_m(\mathbf{x}_n) \neq l_n\right)\right)$, $n = 1, \ldots, N$.
(e) Re-normalize $\mathbf{W}$.
3- Output Final Classifier: $H(\mathbf{x}) = \arg\max_k \sum_{m=1}^{M} \alpha_m I\left(T_m(\mathbf{x}) = k\right)$.

It is worth pointing out also the impact of the above works [112, 114], jointly with [20], on the multi-class field of Boosting. For instance, J.Huang et al. [39] proposed GAMBLE (Gentle Adaptative Multi-class Boosting Learning), an algorithm that takes advantage of the same vectorial labels and loss function of SAMME, for one side, and the same type of weak learners and structure of GentleBoost, for another. The resulting multi-class Boosting algorithm is merged with an active learning methodology to scale up to large data sets.

To finish the section, let us describe the Multi-category GentleBoost algorithm introduced in [114]. This method is a prominent example of multi-class algorithm conceived to address a coordinate-wise fit of the margin vector $\mathbf{f}(\mathbf{x})$.See its pseudo-code in Algorithm 6. Multi-category GentleBoost resorts to the exponential loss (applied to the margin $z = f_l(\mathbf{x})$) to build the vectorial additive model. It works as follows. A vectorial function $\mathbf{h}(\mathbf{x}) \in \mathbb{R}^K$ is initialized to zero in each of its coordinates. The iterative process updates $\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) + \mathbf{g}(\mathbf{x})$ aiming to find the direction, $\mathbf{g}(\mathbf{x})$, that makes the empirical risk

$$\frac{1}{N} \sum_{n=1}^{N} \exp\left(-h_{l_n}(\mathbf{x}_n) + \frac{1}{K} \sum_{k=1}^{K} h_k(\mathbf{x}_n)\right) \tag{2.22}$$

decrease most. This is accomplished by considering the expansion of (2.22) to the second order and then simplifying the Hessian considering only its diagonal. Doing so, it can be verified that the optimal j-th coordinate in $\mathbf{g}(\mathbf{x})$ minimizes

$$-\sum_{n=1}^{N} g_j(\mathbf{x}_n) z_{nj} \exp\left(-f_{l_n}(\mathbf{x}_n)\right) + \frac{1}{2} \sum_{n=1}^{N} g_j^2(\mathbf{x}_n) z_{nj}^2 \exp\left(-f_{l_n}(\mathbf{x}_n)\right)\ , \tag{2.23}$$

where $z_{nj} = -1/K + I(l_n = j)$ and $\mathbf{f}(\mathbf{x})$ is the margin vector corresponding to the model already fitted (i.e $\mathbf{h}(\mathbf{x})$ mapped on the margin hyperplane). A simple way to solve it is calculating a regression function by weighted least-squares of the target variable z_{nj}^{-1} to $\mathbf{x}_n$ with weights $w(n, j) = z_{nj}^2 \exp(-f_{l_n}(\mathbf{x}_n))$. Finally, $\mathbf{h}$ is projected to the hyperplane of margin vector in order to classify or compute a posteriori probabilities if required. It is easy to verify that Multi-category GentleBoost extends canonically its binary counterpart GentleBoost [29].

Algorithm 6 : Multi-category GentleBoost

1- Initialize the weight vector $\mathbf{W}$ with constant distribution $w(n) = 1\ ,\ n = 1, \ldots, N$.
2- Initialize coordinates $h_k(\mathbf{x}) = 0$, for $k = 1, \ldots, K$.
3- For $m = 1$ to M:
(a) For $k = 1$ to K:
* Let $z_n := -1/K + I(l_n = k)$. Compute $w^*(n) = z_n^2 \exp(-f_{l_n}(\mathbf{x}_n))$.
* Fit a regressor $g_k(\mathbf{x})$ by least-squares of variable z_n^{-1} to $\mathbf{x}_n$ weighted with $w^*(n)$.
* Update $h_k(\mathbf{x}) = h_k(\mathbf{x}) + g_k(\mathbf{x})$.
(b) Compute $\mathbf{f}(\mathbf{x})$, the projection of $\mathbf{h}(\mathbf{x})$ into the margin hyperplane.
(c) Compute $w(n) = \exp(-f_{l_n}(\mathbf{x}_n))$, $n = 1, \ldots, N$.
4- Output Final Classifier: $H(\mathbf{x}) = \arg\max_k f_k(\mathbf{x})$.

2.4 Cost-sensitive binary Boosting

A second mandatory extension for AdaBoost comes with the chance of having different costs for misclassifications. Assume that the binary classification problem at hand is provided with a (2×2)-cost matrix

$$\mathbf{C} = \begin{pmatrix} C(1,1) & C(1,2) \\ C(2,1) & C(2,2) \end{pmatrix} \tag{2.24}$$

with no negatives real values. Here, rows are referred to real labels while columns indicate predicted labels.

Since its beginning, AdaBoost has received a notable degree of attention in order to adapt its good properties to a cost-sensitive structure like the above. It is very common to set the diagonal equal to zero, i.e. no costs for correct classifications. We will justify this consideration in section 4.1. For ease of notation we will use C_1 and C_2 to denote the constants $C(1,2)$ and $C(2,1)$, respectively.

Initial attempts to generalize AdaBoost in this fashion came essentially from heuristic changes on specific parts of the pseudo-code. Such is the case of CSB0, CSB1 and CSB2 [94, 93]. Their respective reweighting schemes for the n-th instance are given by:

$$w_{m+1}(n) = \begin{cases} w_m(n) & \text{if } f_m(\mathbf{x}_n) = y_n \\ C_{y_n} w_m(n) & \text{if } f_m(\mathbf{x}_n) \neq y_n \end{cases} , \tag{2.25}$$

$$w_{m+1}(n) = \begin{cases} w_m(n) \exp(-y_n f_m(\mathbf{x}_n)) & \text{if } f_m(\mathbf{x}_n) = y_n \\ C_{y_n} w_m(n) \exp(-y_n f_m(\mathbf{x}_n)) & \text{if } f_m(\mathbf{x}_n) \neq y_n \end{cases} , \tag{2.26}$$

$$w_{m+1}(n) = \begin{cases} w_m(n) \exp(-\alpha_m y_n f_m(\mathbf{x}_n)) & \text{if } f_m(\mathbf{x}_n) = y_n \\ C_{y_n} w_m(n) \exp(-\alpha_m y_n f_m(\mathbf{x}_n)) & \text{if } f_m(\mathbf{x}_n) \neq y_n \end{cases} . \tag{2.27}$$

The common structure of the three algorithms is shown in Algorithm 7.

Another example came with AdaCost, the algorithm proposed by W. Fan et al. [23]. It was developed guided by a weighing rule that includes a particular cost and a margin-dependent function, $\beta(n)$, in the argument of the exponential loss function. So, for an instance $(\mathbf{x}_n, y_n)$, one computes

$$w_{m+1}(n) = w_m(n) \exp(-\alpha_m y_n f_m(\mathbf{x}_n) \beta(n)) . \tag{2.28}$$

In practice, the authors select $\beta(n) = 1/2(1 - z_n C_{y_n})$, where $C(y_n)$ is the cost incurred for the real label y_n and z_n is the associated margin[4] in iteration m. Similarly, AsymBoost [101] is also

[4]Remember the equivalence between labels $\{1, -1\}$, $\{1, 0\}$ and $\{1, 2\}$ for binary problems.

Algorithm 7 :CSB

1- Initialize the weight vector $\mathbf{W}$ with $w(n) = C_{y_n}/Z_0$, $n = 1, \ldots, N$; where Z_0 is a normalization constant.
2- For $m = 1$ to M:
(a) For $f \in Pool$
* Compute weighted error of f, i.e. $Err = \sum_{n=1}^{N} w(n) I\left(f(\mathbf{x}_n) \neq y_n\right)$.
end for
(b) Select f_m, the weak hypothesis with minimum weighted error.
(c) Compute $\alpha_m = \log\left((1 - Err_m)/Err_m\right)$.
(d) Update weight vector following either (2.25), (2.26) or (2.27).
(e) Re-normalize $\mathbf{W}$.
3- Output Final Classifier:
$H(\mathbf{x}) = sign \sum_{m=1}^{M} \alpha_m f_m(\mathbf{x})(C_1 I(f_m(\mathbf{x}) = 1) + C_2 I(f_m(\mathbf{x}) = -1))$

based on a reweighting scheme:

$$w_{m+1}(n) = w_m(n) \exp(-\alpha_m y_n f_m(\mathbf{x}_n))(C_1/C_2)^{y_n/2m} . \tag{2.29}$$

This choice seems to be non optimal due to its dependence of the current iteration m, that differs from the "adaptive" property of the AdaBoost algorithm.

In the same way, Y.Sun et al. [89, 87] proposed another three ways to update weights in a cost-sensitive fashion:

1. $w_{m+1}(n) = w_m(n) \exp(-\alpha_m C_{y_n} y_n f_m(\mathbf{x}_n))$
2. $w_{m+1}(n) = w_m(n) C_{y_n} \exp(-\alpha_m y_n f_m(\mathbf{x}_n))$
3. $w_{m+1}(n) = w_m(n) C_{y_n} \exp(-\alpha_m C_{y_n} y_n f_m(\mathbf{x}_n))$

No formal reason is given to include costs "inside" the exponential loss, "outside" the exponential loss, and "in both places" jointly. Each reweighting scheme yields, in turn, its cost-sensitive extension of AdaBoost, namely AdaC1, AdaC2 and AdaC3. Algorithm 8 shows AdaC2's pseudo-code. We will come back to it in chapter 4.

Algorithm 8 :AdaC2

1- Initialize the weight vector $\mathbf{W}$ with uniform distribution $w(n) = 1/N$, $n = 1, \ldots, N$.
2- For $m = 1$ to M:
(a) Fit a classifier $f_m(\mathbf{x})$ to the training data using weights $\mathbf{W}$.
(b) Compute weighted error: $Err_m = \sum_{n=1}^{N} w(n) I\left(f_m(\mathbf{x}_n) \neq y_n\right)$.
(c) Compute $\alpha_m = \log\left((1 - Err_m)/Err_m\right)$.
(d) Update weight vector $w(n) \leftarrow w(n) C_{y_n} \exp\left(\alpha_m I\left(f_m(\mathbf{x}_n) \neq y_n\right)\right), n = 1, \ldots, N$.
(e) Re-normalize $\mathbf{W}$.
3- Output Final Classifier: $H(\mathbf{x}) = sign \sum_{m=1}^{M} \alpha_m f_m(\mathbf{x})$

More recently, two works came to shed light on the cost-sensitive capability of Boosting from a more formal point of view. On the one hand Masnadi-Shirazi and Vasconcelos' work [63] (see also their previous paper [60]) may be considered a canonical extension of AdaBoost to

this field. Beside other three methods for binary cost-sensitive problems, this paper introduces *Cost-Sensitive AdaBoost* (CS-AdaBoost). The core idea behind the algorithm is substituting the original exponential loss function by a cost-dependent derivative:

$$\mathcal{L}_{CS-Ada}(y, F(\mathbf{x})) = I(y=1)\exp\left(-yC_1F(\mathbf{x})\right) + I(y=-1)\exp\left(-yC_2F(\mathbf{x})\right) \ . \quad (2.30)$$

It is clear that it becomes AdaBoost when a $0|1$-cost matrix is used. CS-Adaboost is then derived by fitting an additive model whose objective is to minimize the expected loss. Algorithm 9 shows the pseudo-code. Like previous approaches, the algorithm needs to have a pool of available weak learners from which to select the optimal one in each iteration, jointly with the optimal step β. For a candidate weak learner, $g(\mathbf{x})$, they compute two constants summing up the weighted errors associated to instances with the same label, b for label 1 and d for label -1. Then they calculate β by finding the only real solution to the following equation:

$$2C_1 b\cosh\left(\beta C_1\right) + 2C_2 d\cosh\left(\beta C_2\right) = T_1C_1\mathrm{e}^{-\beta C_1} + T_2C_2\mathrm{e}^{-\beta C_2} \ . \quad (2.31)$$

Finally, the pair $(g(\mathbf{x}), \beta)$ minimizing $\mathcal{L}_{CS-Ada}(y, F(\mathbf{x}) + \beta g(\mathbf{x}))$ is added to the model. A particularity of CS-AdaBoost lies in the initial weigthing of instances, that attempts to distribute the sum of positive and negative weights evenly ($1/2$ for each group of weights). See **1-** in Algorithm 9 for $N_1 = \sum_{n=1}^{N} I(y_n = 1)$ and $N_{-1} = \sum_{n=1}^{N} I(y_n = -1)$.

Algorithm 9 :Cost-Sensitive AdaBoost

1- Initialize the weight vector $\mathbf{W}$ with distribution $w(n) = 1/2N_{y_n}\ ,\ n = 1, \ldots, N$.
2- For $m = 1$ to M:
(a) Calculate constants: $T_1 = \sum_{y_n=1}^{N} w(n)$, $T_2 = \sum_{y_n=-1}^{N} w(n) = 1 - T_1$.
(b) For $f \in Pool$:
* Calculate: $b = \sum_{y_n=1}^{N} w(n) I(y_n \neq f(\mathbf{x}_n))$, $d = \sum_{y_n=-1}^{N} w(n) I(y_n \neq f(\mathbf{x}_n))$.
* Calculate β, the solution to equation (2.31).
* Compute the weighted error using $\sum_{n=1}^{N} \mathcal{L}_{CS-Ada}(y_n, F_{m-1}(\mathbf{x}_n) + \beta f(\mathbf{x}_n))$.
end for
(c) Select the pair (f_m, β_m) of minimum weighted error.
(d) Update weights: $w(n) \leftarrow w(n) \cdot L_{CS-Ada}(y_n, \beta_m f_m(\mathbf{x}_n))$, $n = 1, \ldots, N$.
(e) Re-normalize $\mathbf{W}$.
3- Output Final Classifier: $H(\mathbf{x}) = sign \sum_{m=1}^{M} \beta_m f_m(\mathbf{x})$.

On the other hand Landesa-Vazquez and Alba-Castro's work [49] discuss the effect of an initial non-uniform weighing of instances to endow AdaBoost with a cost-sensitive behaviour. The resulting method, Cost-Generalized AdaBoost, takes advantage of the remainder of the AdaBoost's original structure.

Beside these approaches, we must point out that probably the most intuitive way for applying AdaBoost to cost-sensitive problems came with Viola and Jones' work for detection [100]. The authors basically maintained the original algorithm and introduce a threshold, θ, to bias the committed response in favour of the most costly label. Hence, the resulting variation has the shape

$$H(\mathbf{x}) = sign\left(F(\mathbf{x}) - \theta\right) = sign\left(\sum_{m=1}^{M} \alpha_m f_m(\mathbf{x}) - \theta\right) \ . \quad (2.32)$$

Obviously, the threshold is manipulated over a validation set to asses the required quality of detection. If an actual cost matrix is given then θ is easily calculated:

$$F_C(\mathbf{x}) = \log \frac{P(Y=1|\mathbf{x})C_1}{P(Y=-1|\mathbf{x})C_2} = \log \frac{P(Y=1|\mathbf{x})}{P(Y=-1|\mathbf{x})} - \log \frac{C_2}{C_1} = F(\mathbf{x}) - \theta \,. \tag{2.33}$$

It is also easy to verify that this way of proceeding is equivalent to change the a priori probabilities from a generative point of view:

$$F_C(\mathbf{x}) = \log \frac{P(Y=1|\mathbf{x})C_1}{P(Y=-1|\mathbf{x})C_2} = \log \frac{P(\mathbf{x}|Y=1)P(Y=1)C_1}{P(\mathbf{x}|Y=-1)P(Y=-1)C_2} \,, \tag{2.34}$$

since the ratio of a prioris $P(Y=1)/P(Y=-1)$ is directly corrected by C_1/C_2. Moving the cost-insensitve decision boundary is a suboptimal strategy because the are no guarantees of fitting the true cost-sensitive decision boundary, as was pointed out in [63].

2.5 Other perspectives of Boosting

So far we have only considered the extensions of AdaBoost most closely related to our work in this thesis, namely: multi-class and binary cost-sensitive problems. We close the chapter giving a short overview of other interesting perspectives driven by Boosting in the world of Machine Learning. The second part of the section is devoted to comment the presence of Boosting in some Computer Vision tasks. Here is a short list of topics where Boosting has allowed new venues of research:

- *New loss functions*. Some recent advances in Boosting came from developing a loss function that allows a desired property. Such is the case of the SavageBoost [62] and TangentBoost [61] algorithms. For both methods the authors derived a loss function specially designed to prevent the effect of outliers in the classification. Another examples can be found in [10, 80, 34, 71].

- *Semi-supervised learning*. Problems with unlabeled instances are also present in the scope of Boosting. Two recent works in this area are [59], where the SemiBoost algorithm is introduced, and [13], where a margin-based cost function is regularized in order to be optimized in a supervised way. For readers interested in this field we also recommend [85, 53, 75].

- *Entropy projection*. Kivinen and Warmuth's work [46] discovered properties between consecutive weight vectors $\{\mathbf{W}_m, \mathbf{W}_{m+1}\}$, understood as probability distributions. Specifically the new destribution, $\mathbf{W}_{m+1}$, is the closest to the previous one, via the relative entropy, from those belonging to the orthogonal hyperlane of weight vectors. See [84] for another perspective in which the Lagrange dual problems of some Boosting derivations are proven to be entropy maximization problems.

- *Regresion*. Obviously the world of regresion has also received attention in the Boosting literature. We highlight GradientBoost [31], the algorithm developed by J.Friedman as a reference in the area. Another relevant derivation is the above mentioned GentleBoost [29, 114].

- *Game theory*. In the work [27] the authors explained the connection between Boosting and game theory. Specifically they described AdaBoost in terms of a 2-players game where one of them fits a weak learner based on a given weight vector while the second player receives the weighted error and computes the α value with which derive the next vector of weights to the first player.

- *Mahalanobis distance*. Informative distances, like Mahalanobis, have also been used in the stage-wise optimization performed by Boosting. The most representative works in the area were introduced by C. Shen et al. [82, 83]. The idea behind their proposal is the use of "diferences between Mahalanobis distances", $d^2_M(i,j) - d^2_M(i,k)$, as argument (margin) for the exponential loss function. Specifically, the objective of the minimization is a sum of as many exponetial losses as restrictions $d_M(i,j) > d_M(i,k)$ are required among point triplets, (i,j,k). Following this insight they derived the *MetricBoost* algorithm.

- *Condition of Boostability*. This terminology is referred to works that address the conditions for a Boosting algorithm to have guarantees of convergence. We highlight the paper of I. Mukherjee and R. Schapire [67], in which both strong and weak conditions of "boostability" are given for multi-class Boosting algorithms. These conditions are evaluated on previous relevant works. An alternative analysis of the convergence of Boosting under several loss functions is devoted to M. Telgarsky [92].

- *Relationship to other paradigms*. Since Boosting is a particular case of meta-classifier, it was a matter of time combining its structure with other relevant paradigms. Such is the case of DeepBoost [14], a recent work where strong learners are allowed as basis without losing quality of fit. The method is open to be combined with deep decision trees, Support Vector Machines or even Neural Networks. For an interesting comparative with Bagging with respect to robustness see S. Rosset's work [73].

Taking aside the above topics, Boosting has shown to be an excellent tool for many problems in the area of Computer Vision. Some of them require modifications in the original formulation of the algorithms to provide optimal results. That is the case of detection problems, where the shape of the strong learner calculated with AdaBoost is pruned conveniently to acquire a cascade-shaped structure. In case of *face detection* we must highlight the above mentioned works of P. Viola and M. Jones [101, 100]. It has also been successfully used for recognizing texts [12, 22], deriving object detectors efficiently [95, 99, 52], and labelling images [43, 108, 105]. Moreover, Boosting has became a very useful strategy for feature selection in Computer Vision problems [96, 48].

Other interesting application of Boosting in Computer Vision is to identify personal characteristics from low resolution pictures of faces. Such is the case, for instance, of gender recognition. Here we must point out S. Baluja y H. Rowley's work [5], in which AdaBoost uses simply comparisons of gray-level between pairs of pixels to obtain significantly good results. See [107] for another approach, in this case based on second order discriminant analysis updated iteration after iteration.

Chapter 3

Partially Informative Boosting

In section 2.3.1 we have discussed some multi-class Boosting algorithms based on binary weak learners, that essentially separate the set of classes into two groups. None of them is a proper extension of AdaBoost in the sense of taking advantage of the exponential loss function in a pure multi-class fashion. This is exactly the root of our theoretical improvement. Can we transfer partial responses to the multi-class field maintaining this property? So far we have discussed the important role of margin for binary and multi-class Boosting. Here we extend this concept to manage binary sub-problems properly and, hence, to answer the above question.

In this chapter we introduce a multi-class generalization of AdaBoost that uses ideas present in previous works. We use binary weak-learners to separate groups of classes, like [3, 78, 80], and a margin-based exponential loss function with a vectorial encoding like [51, 112, 39]. However, the final result is new. To model the uncertainty in the classification provided by each weak-learner we use different vectorial encodings for representing class labels and classifier responses. This codification yields an asymmetry in the evaluation of classifier performance that produces different margin values depending on the number of classes separated by each weak-learner. Thus, at each Boosting iteration, the sample weight distribution is updated as usually according to the performance of the weak-learner, but also, depending on the number of classes in each group. In this way our Boosting approach takes into account both, the uncertainty in the classification of a sample in a group of classes, and the imbalance in the number of classes separated by the weak-learner [87, 38]. Specifically, we decompose the problem into binary subproblems whose goal is to separate a set of labels from the rest and then we encode every response using a new set of margin vectors in a way that the multi-class exponential loss function can be applied. The resulting algorithm is called *PIBoost*, which stands for Partially Informative Boosting, reflecting the idea that the Boosting process collects partial information about classification provided by each weak-learner. PIBoost is well grounded theoretically and provides significantly good results. We consider it, perhaps, the only canonical extension of AdaBoost based on binary weak learners.

The chapter is organized as follows. Next section is devoted to introduce our new set of margin vectors jointly with the loss function. Section 3.2 describes PIBoost in detail. Here we pay attention to Lemma 1, the main result upon which the algorithm is based. Paragraphs showing PIBoost's relationships with AdaBoost and CS-AdaBoost are also included. We also point out how PIBoost implies a pattern of common sense when taking decisions. In section 3.3 we compare the main points of our algorithm with those of the ECOC-based algorithms commented in section 2.3.1. Finally, we devote section 3.4 to discuss experiments where the accuracy of PIBoost against other relevant multi-class Boosting methods is analyzed.

3.1 Multi-class margin extension

We saw in section 2.3.2 how the use of margin vectors for encoding labels induces a natural generalization of binary classification, yielding margins that derive multi-class algorithms. In this section we introduce a new multi-class margin expansion. Similarly to [51, 114, 112, 74, 39] we use margin vectors to represent multi-class membership, i.e. real labels. However, in our proposal, data labels and those estimated by a classifier will not be defined on the same set of vectors. Our margin vectors will produce, for each iteration of the algorithm, different margin values for each sample, depending on the number of classes separated by the weak-learner. This fact is related to the asymmetry produced in the classification when the number of classes separated by a weak learner is different on each side and to the "difficulty" or information content of that classification.

Remember that the essence of the margin approach resides in maintaining negative/positive values of the margin when a classifier has respectively a failure/success. That is, if $\mathbf{y}, \mathbf{f}(\mathbf{x}) \in Y$, the margin $z = \mathbf{y}^\top \mathbf{f}(\mathbf{x})$ satisfies: $z > 0 \Leftrightarrow \mathbf{y} = \mathbf{f}(\mathbf{x})$ and $z < 0 \Leftrightarrow \mathbf{y} \neq \mathbf{f}(\mathbf{x})$. We extend the set Y by allowing that each $\mathbf{y}_l$ may also take a negative value, that can be interpreted as a fair vote for *any label but the l-th*. This vector encodes the uncertainty in the response of the classifier, by evenly dividing the evidence among all labels but the l-th. It provides the smallest amount of information about the classification of an instance; i.e. a negative classification, the instance does not belong to class l but to any other. Our goal is to build a Boosting algorithm that combines both positive and negative weak responses into a *strong decision*.

Following this intuition we introduce new margin vectors through fixing a group of s-labels, $S \in \mathcal{P}(L)$, and defining $\mathbf{y}^S$ in the following way:

$$\mathbf{y}^S = \left(y_1^S, \ldots, y_K^S\right)^\top \text{ with } y_i^S := \begin{cases} \frac{1}{s} & \text{if } i \in S \\ \frac{-1}{K-s} & \text{if } i \notin S \end{cases} \tag{3.1}$$

It is straightforward that any $\mathbf{y}^S$ is a margin vector [51, 114]. In addition, if S^c is the complementary set of $S \in \mathcal{P}(L)$, then $\mathbf{y}^{S^c} = -\mathbf{y}^S$. Let $\hat{Y}$ be the whole set of vectors obtained in this fashion. We want to use $\hat{Y}$ as target set, that is: $\mathbf{f} : X \to \hat{Y}$, but under a binary perspective. The difference with respect to other approaches using similar codification [51, 114, 112] is that the correspondence defined in (2.16) is broken. In particular, weak-learners will take values in $\left\{\mathbf{y}^S, -\mathbf{y}^S\right\}$ rather than the whole set $\hat{Y}$. The combination of answers obtained by the Boosting algorithm will provide complete information over $\hat{Y}$. So now the correspondence for each weak-learner is binary

$$F^S(\mathbf{x}) = \pm 1 \;\Leftrightarrow\; \mathbf{f}^S(\mathbf{x}) = \pm \mathbf{y}^S \,, \tag{3.2}$$

where $F^S : X \to \{+1, -1\}$ is a classifier that recognizes the presence $(+1)$ or absence (-1) of the group of labels S in the data.

We propose a multi-class margin for evaluating the answer given by $\mathbf{f}^S(\mathbf{x})$. Data labels always belong to Y but predicted ones, $\mathbf{f}^S(\mathbf{x})$, belong to $\hat{Y}$. In consequence, depending on $s = |S|$, we have four possible margin values

$$z = \mathbf{y}^\top \mathbf{f}^S(\mathbf{x}) = \begin{cases} \pm \frac{K}{s(K-1)} & \text{if } \mathbf{y} \in S \\ \pm \frac{K}{(K-s)(K-1)} & \text{if } \mathbf{y} \notin S \end{cases} \tag{3.3}$$

where the sign is positive/negative if the partial classification is correct/incorrect. The derivations of the above expressions are in the Appendix A.1.

We use the multi-class exponential loss just as it was introduced in 2.3.2 to evaluate these margins (3.3):

$$\mathcal{L}\left(\mathbf{y}, \mathbf{f}^S(\mathbf{x})\right) = \exp\left(\frac{-1}{K}\mathbf{y}^\top \mathbf{f}^S(\mathbf{x})\right) . \tag{3.4}$$

In consequence, the above vectorial codification of labels will produce different degrees of punishes and rewards depending on the number of classes separated by the weak-learner. Assume that we fix a set of classes, S, and an associated weak-learner that separates them from the rest, $\mathbf{f}^S(\mathbf{x})$. We may also assume that $|S| \leq K/2$, since if $|S| > K/2$ we can choose $S' = S^c$ and then $|S'| \leq K/2$. The failure or success of $\mathbf{f}^S(\mathbf{x})$ in classifying an instance $\mathbf{x}$ with label $l \in S$ will have a larger margin than when classifying an instance with label $l \in S^c$. The margins in (3.3) provide the following rewards and punishes when used in conjunction with the exponential loss (3.4)

$$\mathcal{L}\left(\mathbf{y}, \mathbf{f}^S(\mathbf{x})\right) = \begin{cases} \exp\left(\frac{\mp 1}{s(K-1)}\right) & \text{if } \mathbf{y} \in S \\ \exp\left(\frac{\mp 1}{(K-s)(K-1)}\right) & \text{if } \mathbf{y} \notin S. \end{cases} \tag{3.5}$$

In dealing with the class imbalance problem, the losses produced in (3.5) reflect the fact that the importance of instances in S is higher than those in S^c, since S is a smaller set. Hence, the cost of miss-classifying an instance in S outweighs that of classifying one in S^c [87]. This fact may also be intuitively interpreted in terms of the "difficulty" or amount of information provided by a classification. Classifying a sample in S provides more information, or, following the usual intuition behind Boosting, is more "difficult" than the classification of an instance in S^c, since S^c has a broader range of possible labels. The smaller the set S the more "difficult" or informative will be the result of the classification of an instance in it.

We can further illustrate this idea with an example. Assume that we work on a classification problem with $K = 5$ classes. We may select $S_1 = \{1\}$ and $S_2 = \{1, 2\}$ as two possible sets of labels to be learned by our weak-learners. Samples in S_1 should be the more important than those in S_1^C or in S_2, since S_1 has the smallest class cardinality. Similarly, in general, it is easier to recognize data in S_2 than in S_1, since the latter is smaller; i.e. classifying a sample in S_1 provides more information than in S_2. Encoding labels with vectors from Y we will have the following margin values and losses:

$$z = \mathbf{y}^\top \mathbf{f}^{S_1}(\mathbf{x}) = \begin{cases} \pm 5/4 \\ \pm 5/16 \end{cases} \Rightarrow \mathcal{L}(\mathbf{y}, \mathbf{f}^{S_1}) = \begin{cases} e^{\pm 1/4} = \{0.77, 1.28\} & \mathbf{y} \in S_1 \\ e^{\pm 1/16} = \{0.93, 1.06\} & \mathbf{y} \in S_1^c \end{cases}$$

$$z = \mathbf{y}^\top \mathbf{f}^{S_2}(\mathbf{x}) = \begin{cases} \pm 5/8 \\ \pm 5/12 \end{cases} \Rightarrow \mathcal{L}(\mathbf{y}, \mathbf{f}^{S_2}) = \begin{cases} e^{\pm 1/8} = \{0.88, 1.13\} & \mathbf{y} \in S_2 \\ e^{\pm 1/12} = \{0.92, 1.08\} & \mathbf{y} \in S_2^c \end{cases}$$

Everything we say about instances in S_1 will be the most rewarded or penalized in the problem, since S_1 is the smallest class set. Set S_2 is the second smallest, in consequence classification in that set will produce the second largest rewards and penalties. Similarly, we "say more" excluding an instance from $S_2 = \{1, 2\}$ than from $S_1 = \{1\}$, since S_2^c is smaller than S_1^c. In consequence, rewards and penalties for samples classified in S_2^c will be slightly larger than those in S_1^c. In Fig. 3.1 we display the loss values for the separators associated to the sets S_1 and S_2.

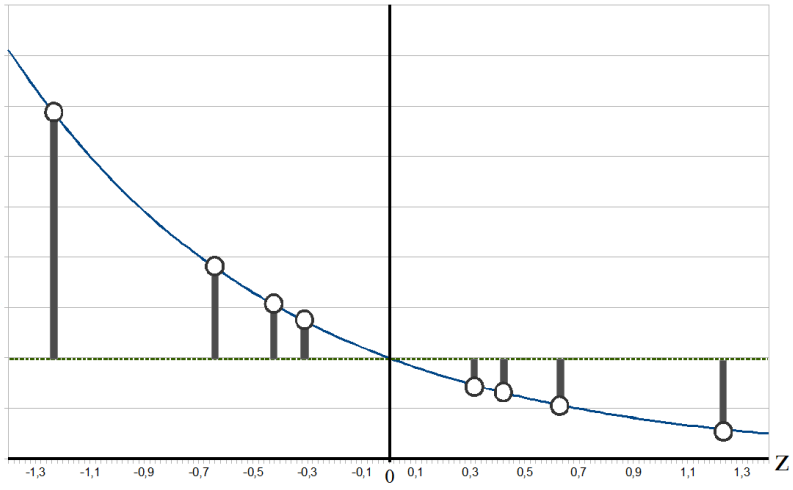


Figure 3.1: Values of the Exponential Loss Function over margins, z, for a classification problem with 4-classes. Possible margin values are obtained taking into account the expression (3.5) for $s = 1$ and $s = 2$.

3.2 PIBoost

In this section we present the structure of PIBoost [24], whose pseudo-code we show in Algorithm 10. At each Boosting iteration we fit as many weak-learners as groups of labels, $\mathcal{G} \subset \mathcal{P}(L)$, are considered. The aim of each weak-learner is to separate its associated labels from the rest and persevere in this task iteration after iteration. That is the reason why we call them *separators*. A weight vector $\mathbf{W}^S$ is associated to the separator of set S.

For each set $S \in \mathcal{G}$, with $s = |S|$, PIBoost builds a stage-wise additive model [37] of the form $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$ (where super-index ^{S} is omitted for ease of notation). In step **2** of the algorithm we estimate constant β and function $\mathbf{g}(\mathbf{x})$ for each label and iteration. The following Lemma solves the problem of finding these parameters.

Lemma 1. *Given an additive model* $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$ *associated to a set of labels,* $S \in \mathcal{G}$*, the solution to*

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg\min_{\beta, \mathbf{g}(\mathbf{x})} \sum_{n=1}^{N} \exp\left(\frac{-1}{K} \mathbf{y}_n^\top \left(\mathbf{f}_{m-1}(\mathbf{x}_n) + \beta \mathbf{g}(\mathbf{x}_n)\right)\right) \tag{3.6}$$

is obtained in the following way:

- *Given* $\beta > 0$*, the optimal weak learner is:*

$$\mathbf{g}_m = \arg\min_{\mathbf{g}} B_1 \sum_{l_n \in S}^{N} w(n) I\left(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0\right) + B_2 \sum_{l_n \notin S}^{N} w(n) I\left(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0\right) \ ,$$

with $B_1 = \left[\exp(\frac{\beta}{s(K-1)}) - \exp(\frac{-\beta}{s(K-1)})\right]$, $B_2 = \left[\exp(\frac{\beta}{(K-s)(K-1)}) - \exp(\frac{-\beta}{(K-s)(K-1)})\right]$.

- *Given a learner* $\mathbf{g}(\mathbf{x})$*, the optimal constant is* $\beta_m = s(K-s)(K-1)\log R$,

where R is the only real positive root of the polynomial

$$P_m(x) = E1(K-s)x^{2(K-s)} + sE2x^K - s(A_2 - E2)x^{(K-2s)} - (K-s)(A_1 - E1) \quad (3.7)$$

and the constants involved in both expresions are defined as follows: $A_1 = \sum_{l_n \in S} w(n)$, $A_2 = \sum_{l_n \notin S} w(n)$ *(i.e.* $A_1 + A_2 = 1$ *),* $E1 = \sum_{l_n \in S} w(n) I(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0)$ *,* $E2 = \sum_{l_n \notin S} w(n) I(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0)$*, and* $\boldsymbol{W}_{m-1} = \{w(n)\}$ *is the weight vector of iteration m-1.*

The proof of this result is in Appendix A.2. As can be seen in the Lemma, the optmization of $\mathbf{g}_m(\mathbf{x})$ and β_m in 3.6 depend on each other. An efficient strategy to solve this problem is to iteratively optimize for one of the variables, assuming known the other. We have considered two ways to proceed: *1)* compute an initial $\mathbf{g}_m$ fixing an initial β_m (1, for simplicity) and; *2)* compute an initial $\mathbf{g}_m$ assuming $B_1 = B_2$. In both cases we have empirically confirmed that the results obtained with sevaral iterations in this process are not significantly better than those in the first iteration. Hence, in Algorithm 10, we introduce the method by assuming the second option ($B_1 = B_2$) and making no sub-iterations to obtain the optimal pair ($\mathbf{g}_m(\mathbf{x})$,β_m), which is the procedure selected for our experiments. With this assumption, the optimal weak learner is calculated according to

$$\mathbf{g}_m = \arg\min_{\mathbf{g}} \sum_{n=1}^{N} w(n) I\left(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0\right) , \quad (3.8)$$

that is an efficient and practical criteria.

Lemma 1 justifies steps **2:(a)**, **(c)**[1] , **(d)** and **(e)** in our pseudo-code. In case $\mathbf{y} \in S$, the update rule **2:(f)** follows from:

$$\begin{aligned} w^S(n) &= w^S(n) \exp\left(\frac{-1}{K} \mathbf{y}_n^\top \beta \mathbf{f}^S(\mathbf{x}_n)\right) \\ &= w^S(n) \exp\left(\frac{-1}{K} s(K-s)(K-1) \log\left(R_m^S\right) \frac{\pm K}{s(K-1)}\right) \\ &= w^S(n) \exp\left(\mp (K-s) \log\left(R_m^S\right)\right) = w^S(n) \left(R_m^S\right)^{\mp(K-s)} \end{aligned}$$

The case $\mathbf{y} \notin S$ provides an analogous expression.

Note in (3.7) that the root will be zero only if $A_1 = E1$, what implies $\beta = -\infty$. This possibility should be considered explicitly in the implementation. An interesting case with closed form solution for the polynomial occurs when K is an even number. Any separator of $s = K/2$ labels yields a simpler formula in (3.7):

$$P(x) = \underbrace{(E1 + E2)}_{Err} x^K - (A_1 - E1 + A_2 - E2) = Err x^K - (1 - Err) ,$$

that can be easily solved with a closed-form solution, $x = \sqrt[K]{\frac{1-Err}{Err}}$, which in turn provides the value

$$\beta = \frac{K(K-1)}{4} \log \frac{1-Err}{Err} = s \frac{K-1}{2} \log \frac{1-Err}{Err}. \quad (3.9)$$

[1] In expression $l_n \notin G_m^S(\mathbf{x})$, the set $G_m^S(\mathbf{x})$ must be understood as $\{G_m^S(\mathbf{x}) = +1\} \equiv S$ and $\{G_m^S(\mathbf{x}) = -1\} \equiv S^C$.

Algorithm 10 : PIBoost

1- Initialize weight vectors $w^S(n) = 1/N$; with $n = 1, \ldots, N$ and $S \in \mathcal{G} \subset \mathcal{P}(L)$.

2- For $m = 1$ until the number of iterations M and for each $S \in \mathcal{G}$:

(a) Fit a binary classifier $G_m^S(\mathbf{x})$ over training data with respect to its corresponding w^S.

(b) Translate $G_m^S(\mathbf{x})$ into $\mathbf{g}_m^S : X \to \hat{Y}$.

(c) Compute 2 types of errors associated with $G_m^S(\mathbf{x})$

$$E1_{S,m} = \sum_{l_n \in S} w_n^S I\left(l_n \notin G_m^S(\mathbf{x}_n)\right) \ , \ E2_{S,m} = \sum_{l_n \notin S} w_n^S I\left(l_n \notin G_m^S(\mathbf{x}_n)\right)$$

(d) Calculate R_m^S, the only positive root of the polynomial $P_m^S(x)$ defined in (3.7).

(e) Calculate $\beta_m^S = s(K-s)(K-1)\log\left(R_m^S\right)$.

(f) Update weights as follows (sign +/- depends on the failure/hit of G_m^S):

- If $l_n \in S$ then $w^S(n) = w^S(n)\left(R_m^S\right)^{\pm(K-s)}$
- If $l_n \notin S$ then $w^S(n) = w^S(n)\left(R_m^S\right)^{\pm s}$,

(g) Re-normalize weight vectors.

3- Output Final Classifier: $H(\mathbf{x}) = \arg\max_k f_k(\mathbf{x})$, where $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_K(\mathbf{x}))^\top = \sum_{m=1}^M \sum_{S \in \mathcal{G}} \beta_m^S \mathbf{g}_m^S(\mathbf{x})$.

The shape of the final classifier is easy and intuitive to interpret. The vectorial function built during the process collects in each k-coordinate information that can be understood as a *degree of confidence* for classifying sample $\mathbf{x}$ into class k. The classification rule assigns the label with highest value in its coordinate. This criterion has a geometrical interpretation provided by the codification of labels as K-dimensional vectors. Since the set $\hat{Y}$ contains margin vectors, the process of selecting the most probable one is carried out on the hyperplane orthogonal to $\mathbf{1} = (1, \ldots, 1)^\top$ (see Fig. 3.2). So, we build our decision on a subspace of $\mathbb{R}^K$ free of *total indifference* about labels. It means, that the final vector $\mathbf{f}(\mathbf{x})$ built during the process will usually present a dominant coordinate that represents the selected label. Ties between labels will only appear in degenerate cases. The plot on the right in Fig. 3.2 shows the set of pairs of vectors $\hat{Y}$ defined by our extension, whereas on the left are shown the set of vectors Y used in [51, 112]. Although the spanned gray hyperplane is the same, we exploit every binary answer in such a way that the negation of a class is directly translated into a new vector that provides positive evidence for the complementary set of classes in the final composition, $\mathbf{f}(\mathbf{x})$. The inner product of class labels $\mathbf{y} \in Y$ and classifier predictions, $\mathbf{f}(\mathbf{x}) \in \hat{Y}$, $\mathbf{y}^\top \mathbf{f}(\mathbf{x})$ produces a set of asymmetric margin values in such a way that, as described in section 3.1, all successes and failures do not have the same importance. Problems with four or more classes are more difficult to be shown graphically but allow richer sets of margin vectors.

The second key idea in PIBoost is that we can build a better classifier when collecting information from positive and negative classifications in $\hat{Y}$ than when using only the positive classifications in the set Y. Each weak-learner, or separator, $\mathbf{g}^S$, acts as a partial expert of the problem that provides us with a clue about *what is the label* of $\mathbf{x}$. Note here that when a weak-learner classifies $\mathbf{x}$ as belonging to a set of classes, the value of its associated step β, that depends on the success rate of the weak-learner, is evenly distributed among the classes in the set. In the same way, the bet will be used to evenly reduce the confidence on coordinates

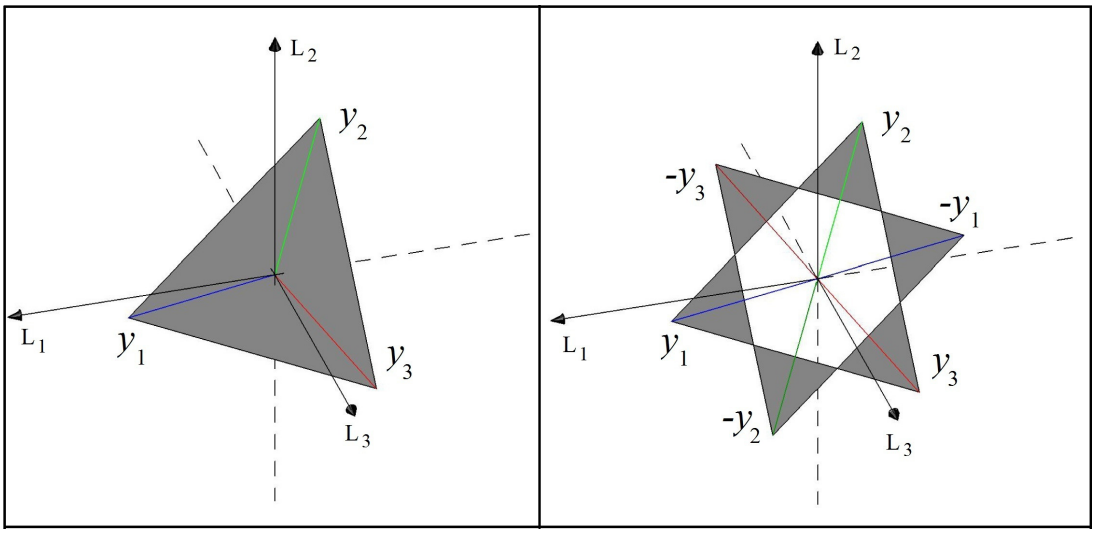


Figure 3.2: Margin vectors for a problem with three classes. Left figure presents the set of vectors Y. Right plot presents the set $\hat{Y}$.

corresponding to non-selected classes. This balance inside selected and discarded classes is reflected in a margin value with a sensible multi-class interpretation. In other words, every answer obtained by a separator is directly translated into multi-class information in a fair way.

3.2.1 AdaBoost as a special case of PIBoost

At this point we can verify that PIBoost applied to a two-class problem is equivalent to AdaBoost. In this case we only need to fit one classifier at each iteration[2]. Thus there will be only one weight vector to be updated and only one group of β constants.

It is also easy to match the expression of parameter β computed in PIBoost with the value of α computed in AdaBoost just by realizing that, fixed an iteration whose index we omit, the polynomial in step **2 - (d)** is

$$P(x) = (E1 + E2)\, x^2 - (A_1 - E1 + A_2 - E2) = Err \cdot x^2 - (1 - Err) \ .$$

Solving this expression we get $R = \left(\frac{1-Err}{Err}\right)^{1/2}$, thus $\beta = \frac{1}{2}\log\left(\frac{1-Err}{Err}\right)$. What indeed is the value of α in AdaBoost. It also could be verified substituting $K = 2$ and $s = 1$ in expression (3.9).

Finally, it is straightforward that the final decisions are equivalent. If we transform AdaBoost's labels, $L = \{+1, -1\}$, into PIBoost's, $L' = \{1, 2\}$, we get that classification rule $H(\mathbf{x}) = sign \sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})$ turns into $\hat{H}(\mathbf{x}) = \arg\max_k f_k(\mathbf{x})$, where

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})) = \sum_{m=1}^{M} \beta_m \mathbf{g}_m(\mathbf{x}) \ .$$

[2]Separating the first class from the second is equivalent to separating the second from the first and, of course, there are no more possibilities.

3.2.2 Asymmetric treatment of partial information

Our codification of class labels and classifier responses produces different margin values. This asymmetry in evaluating successes and failures in the classification may also be interpreted as a form of asymmetric Boosting. As such it is directly related to the Cost-Sensitive AdaBoost in [63].

Using the cost matrix defined in Table 3.1, we can relate the PIBoost algorithm with the Cost-Sensitive AdaBoost [63]. If we denote $b = E1_S$, $d = E2_S$, $T_+ = A_1$, $T_- = A_2$ then the polynomial (3.7), $P^S(x)$, solved at each PIBoost iteration to compute the optimal step, β_m, along the direction of largest descent $\mathbf{g}_m(\mathbf{x})$ is equivalent to the following $\cosh$-depending expression used in the Cost-Sensitive AdaBoost to estimate the same parameter [63]

$$2C_1 b \cosh(C_1\alpha) + 2C_2 d \cosh(C_2\alpha) = C_1 T_+ e^{-C_1\alpha} + C_2 \cdot T_- e^{-C_2\alpha} , \qquad (3.10)$$

where the costs $\{C_1, C_2\}$ are the non-zero values in Table 3.1. In consequence, PIBoost is

Real \ Pred.	S	S^c
S	0	$\frac{1}{s(K-1)}$
S^c	$\frac{1}{(K-1)(K-s)}$	0

Table 3.1: Cost Matrix associated to a PIBoost's separator of a set S with $s = |S|$ classes.

a Boosting algorithm that combines a set of cost-sensitive binary weak-learners whose costs depend on the number of classes separated by each weak-learner.

See the work of I. Landesa-Vazquez and J.L. Alba-Castro [50] for a better understanding of equation (3.10). In their proposal a double base[3] analysis of the asymmetries is discussed. Moreover, they also resort to a polynomial with the same shape of (3.7) to find the optimal constant β added at each iteration. We find quite interesting the way they decompose the equation in order to speed up finding the solution of the polynomial in their resulting algorithm *AdaBoostDB*.

3.2.3 Common sense pattern

We must emphasize that PIBoost's structure links with a pattern of common sense. In fact we apply this philosophy in our everyday life when we try to guess something discarding possibilities.

Let us illustrate this with an example. Assume that a boy knows that his favourite pen has been stolen in his classroom. Even though he probably thinks of a suspect he also has the chance to ask each classmate what he knows about the issue. Perhaps doing so he will collect a pool of useful answers of the kind: "I think it was Jimmy", "I am sure it was not a girl", "I just know that it was neither me not Victoria", "I would suspect of Martin and his group of friends", etc. It is clear that none of these answers drives the boy to a final conclusion (at least, they should not) but they form a set of clues quite useful.

Supposing that no more information is available it seems that an immediate strategy could be to sum up those answers weighted by the degree of confidence associated to those questioned.

[3] Such doble base comes from the two arguments of $\cosh$ appearing in (3.10).

Thus combining all that information our protagonist should have one suspect at the end of the working day. It's easy to find similarities between such a situation and the structure of PIBoost: the answer of each friend can be seen as a weak-learner G_m^S, the level of credibility (or trust) associated to each is our β_m, while the iteration value m can be thought as a measure of time in the relationship with the classmates.

3.3 Related work

Here we discuss some relationships between the ECOC-based algorithms commented in section 2.3.1 and PIBoost. Table 3.3 summarizes the main properties of the four algorithms, extending the comparative presented in that section.

Firstly, the loss function applied for updating weights in AdaBoost.OC relies on the exponential loss with a pseudo-loss as argument, while AdaBoost.MO and AdaBoost.ECC use an exponential loss function with binary arguments. In section 3.1 we have highlighted the importance of using a *pure* multi-class loss function for achieving different margin values, hence penalizing binary failures into a real multi-class context. With our particular treatment for binary sub-problems we extend AdaBoost in a more natural way, because PIBoost can be seen as a group of several binary AdaBoost *well tied* via the multi-class exponential loss function and where every partial answer is well suited for the original multi-class problem. It is not necessary to manage all instance weights linked as one when later a binary loss or a pseudo-loss (that cares of "failures" due to classes over-selected in the coloring function μ_m) will be used.

Besides, the resulting structure of PIBoost is similar to the $\{\pm 1\}$-matrix of ECOC algorithms, except for the presence of fractions. At each iteration of PIBoost there is a block of $|\mathcal{G}|$-response vectors that, grouped as columns, form a $K \times |\mathcal{G}|$-matrix similar to $|\mathcal{G}|$-weak learners of ECOC-based algorithms. Table 3.2 shows the case of a problem with four labels when $\mathcal{G}$ consists of all single labels in union with all pairs of labels. However, in our approach, fractions let us make an even distribution of evidence among the classes in a set, whereas in the ECOC philosophy every binary sub-problem has the same importance for the final count.

Label 1	**Label 2**	**Label 3**	**Label 4**	**Label 1-2**	**Label 1-3**	**Label 1-4**
1	-1/3	-1/3	-1/3	1/2	1/2	1/2
-1/3	1	-1/3	-1/3	1/2	-1/2	-1/2
-1/3	-1/3	1	-1/3	-1/2	1/2	-1/2
-1/3	-1/3	-1/3	1	-1/2	-1/2	1/2

Table 3.2: An example of encoding matrix for PIBoost's weak learners when $K = 4$ and $\mathcal{G} = \{\text{all single labels}\} \cup \{\text{all pairs of labels}\}$.

With respect to margin values, E.L. Allwein, R. Schapire and Y. Singer [3] discussed the strong connections between ECOC-based algorithms and margin theory. The framework developed by the authors provides a new perception of multi-class algorithms based on binary subproblems, unifying the most relevant ones by using matrix encodings with values in $\{+1, 0, -1\}$. A deeper analysis of this fact for AdaBoost.MO, ECC and OC can be found in [91]. Although this development is broad enough to cover the most popular multi-class Boosting methods, we find no reason in using binary margin values, $z_{(k,r)} = l(k,r) f_r(\mathbf{x})$, for measuring the quality of each bit $f_r(\mathbf{x})$ in a predicted-codeword $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_R(\mathbf{x}))$, where $l(k,r)$ is the real value associated to label k for the r-th binary subclassifier. The same binary loss

function can be applied to the resulting value. Our conception of margin values for multi-class problems based on vectorial encodings is richer and provides a broad range of values as was exposed in section 3.1.

So far we have highlighted three essential points, namely: the use of different loss functions, the fact of handling the evidence for or against a set of labels evenly, and our conception of margin; but there are others features distinguishing PIBoost from ECOC-based Boosting algorithms. We describe them briefly. Since we develop independent blocks of weak learners, we endow each sub-model with its own weight vector, $\mathbf{W}^S$, instead of using a whole weight matrix $\mathbf{W} \in \mathbb{R}^{N\times R}$ satisfying $\sum_{n=1}\sum_{r=1} w(n,r) = 1$ (with $R = K$ for the .OC and .ECC variants). Our way of proceeding lets each separator persevere independently, thus providing responses uncorrelated with the rest of separators. Another important aspect is the shape of the final decision rule. Both AdaBoost.OC and AdaBoost.ECC build a K-dimensional function $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x}))^\top$ whose maximum coordinate is selected as response. We argued that this particularity is shared by our algorithm, but we gave a geometric meaning to this way of summarizing information. In fact, it is easy to prove that our rule is equivalent to choosing the vector in Y that provides maximum margin, i.e. $\arg\max_k \mathbf{y}_k^\top \mathbf{f}(\mathbf{x})$. With regard to the amount of weak learners computed at each iteration, we believe that setting blocks of separators at each iteration there is no need for computing coloring functions, that build a single separator based on random patterns [80] and, consequently, does not ensure an uniform label covering[4]. In the same way, we find interesting the use of "code-words" for denoting labels but we think that, once the final hypothesys is commited, it is more intuitive to select the coordinate with maximum value than resort to a metric for measuring the closeness between labels and the obtained final row. Jointly with this last observation, remark that PIBoost, unlike ECOC-based algorithms, computes one value β_m^S per set, S, and iteration, m. In other words, at each iteration $|\mathcal{G}|$ constants are calculated.

Issue	**AdaBoost.MO**	**AdaBoost.OC**	**AdaBoost.ECC**	**PIBoost**
Weights for instances	$\mathbf{W} \in \mathbb{R}^{N\times R}$ for training	$\mathbf{W} \in \mathbb{R}^{N\times K}$ and $\mathbf{D} \in \mathbb{R}^N$ for training	$\mathbf{W} \in \mathbb{R}^{N\times K}$ and $\mathbf{D} \in \mathbb{R}^N$ for training	A vector $\mathbf{W}^S$ per set $S \in \mathcal{G}$
Constants α_m per iteration	One α_m	One α_m	One $g_m(\mathbf{x})$	One β_m^S per set $S \in \mathcal{G}$
Loss function	Binary Exponential	Pseudo-loss Exponential	Binary Exponential	Multi-class Exponential
Final Classifier	Expression (2.15)	$\arg_{l\in L}\max \sum_m^M \alpha_m \bar{I}_m(\mathbf{x})$	$\arg_{l\in L}\max \sum_m^M g_m(\mathbf{x})\mu_m(l)$	Max. coordinate of $\sum_{m,S\in\mathcal{G}}^M \beta_m^S \mathbf{g}_m^S(\mathbf{x})$
W. Learners per iteration	1	1	1	$\lvert\mathcal{G}\rvert$

Table 3.3: Comparison of the main properties of ECOC-based algorithms and PIBoost. $\mu_m(l)$ denotes the coloring function $\mu_m : L \to \{\pm 1\}$ at iteration m. R denotes the length of "code-words". In AdaBoost.OC, $\bar{I}_m(\mathbf{x})$ indicates $I\left(h_m(\mathbf{x}) = \mu_m(l)\right)$.

At this point one can suspect that, independently of the selected group $\mathcal{G}$, PIBoost needs too much computational load at each iteration. This is partially true because, as was said above,

[4]We assume that, when working with PIBoost, the user will select a group G such that, at least, contains all single labels $\{\{k\} \mid k \in L\}$.

separators receive a particular treatment similar to several binary AdaBoost linked via the multi-class exponential loss function. Does such a schedule pay the way when efficiency is required? We postpone our answer to next section, where the quality of PIBoost is revealed even for few iterations.

We must emphasize that PIBoost is not the only multi-class Boosting algorithm using labels separators asymmetrically. As we said in section 2.3.1, AdaBoost.ECC [35] presents different voting weights $g_m(\mathbf{x})$ ($= \alpha_m$ or $-\beta_m$) depending on the class assigned to $(\mathbf{x}, l)$ by the weak learner $h_m(\mathbf{x})$, see (2.14). That is, fixed an iteration m, the algorithm groups the instances into two sets: $G_+ = \{(\mathbf{x}, l) \mid h_m(\mathbf{x}) = +1\}$ and $G_- = \{(\mathbf{x}, l) \mid h_m(\mathbf{x}) = -1\}$. Then the two possible voting constants are independently computed using the same weight vector for instances, $\mathbf{D}_m$. Namely:

$$\alpha_m = \frac{1}{2}\ln\left(\frac{\sum_{\{Hits\ in\ G_+\}} D_m(n)}{\sum_{\{Failures\ in\ G_+\}} D_m(n)}\right)\ ;\ -\beta_m = \frac{-1}{2}\ln\left(\frac{\sum_{\{Hits\ in\ G_-\}} D_m(n)}{\sum_{\{Failures\ in\ G_-\}} D_m(n)}\right)\ .$$

Using these values the global weights for instances $w_{m+1}(n, l)$ are updated via the binary exponential loss function. We provide a similar handling to each separator, but asymmetries *between the number of labels to be separated* are considered inherently in our margin vectors. For PIBoost, after training a new weak learner, we take into account two types of errors (and two constants) that will yield just one value β_m^S once its associated polynomial $P^S(x)$ is solved.

To complete this section we find convenient to discuss briefly the important work of A. Torralba, K.P. Murphy and W.T. Freeman [95] for multi-label problems, where the *JointBoost* algorithm is proposed, and its similarity to PIBoost's structure. Their algorithm takes advantage of the information obtained from binary subproblems focussed on separating groups of labels, which links to our point of view for the multi-class problem. JointBoost is designed to detect different possible objects in images by sharing features. This implies having a set of K labels representing the objects but also an extra label for denoting the background. In consequence the problem is composed by $K + 1$ labels non-equally important. A set of K confidence-rated predictors is fitted by adding separators in the following way: the optimal weak learners for separating groups of labels are computed, then the best of them (after evaluating a variant of weighted error) is added to those predictors whose labels are separated in the chosen group of labels. For a 3 objects problem, the fitted models have the shape:

$$\begin{aligned} H_1(\mathbf{x}) &= G_{1,2,3}(\mathbf{x}) + G_{1,2}(\mathbf{x}) + G_{1,3}(\mathbf{x}) + G_1(\mathbf{x}) \\ H_2(\mathbf{x}) &= G_{1,2,3}(\mathbf{x}) + G_{1,2}(\mathbf{x}) + G_{2,3}(\mathbf{x}) + G_2(\mathbf{x}) \\ H_3(\mathbf{x}) &= G_{1,2,3}(\mathbf{x}) + G_{1,3}(\mathbf{x}) + G_{3,3}(\mathbf{x}) + G_3(\mathbf{x}) \end{aligned} \tag{3.11}$$

Which clearly is different from PIBoost's perspective. In addition, the way JointBoost computes weak learners and the use of weighted squared errors to compare separators are also elements that differenciate this algorithm from PIBoost. It is worth mentioning that tackling multi-label problems in this binary fashion is a well known strategy for developing algorithms in the area, see [4, 43, 108, 105, 12, 22].

3.4 Experiments

Our goal in this section is to evaluate and compare the performance of PIBoost. We have selected fourteen data sets from the UCI repository: *CarEvaluation*, *Chess*, *CNAE9*, *Isolet*, *Multifeatures*, *Nursery*,*OptDigits*, *PageBlocks*, *PenDigits*, *SatImage*, *Segmentation*, *Vehicle*, *Vowel*

and *WaveForm*. They have different numbers of input variables (6 to 856), classes (3 to 26) and instances (846 to 28056), and represent a wide spectrum of types of problems. Although some data sets have separate training and test sets, we use both of them together, so the performance for each algorithm can be evaluated using cross-validation. Table 3.4 shows a summary of the main features of the data bases[5]. For comparison purposes we select three well-known

Data set	**Variables**	**Labels**	**Instances**
CarEvaluation	6	4	1728
Chess	6	18	28056
CNAE9	856	9	1080
Isolet	617	26	7797
Multifeatures	649	10	2000
Nursery	8	5	12960
OptDigits	64	10	5620
PageBlocks	10	5	5473
PenDigits	16	10	10992
SatImage	36	7	6435
Segmentation	19	7	2310
Vehicle	18	4	846
Vowel	10	11	990
Waveform	21	3	5000

Table 3.4: Summary of selected UCI data sets

multi-class Boosting algorithms. AdaBoost.MH [80] is perhaps the most prominent example of multi-class classifier with binary weak-learners. Similarly, SAMME [112] is, under our perspective, the most well known canonical multi-class algorithms with multi-class weak-learners. Finally, Multi-category GentleBoost [114] is probably the most accurate method that treats labels separately at each iteration. We display their respective pseudo-codes in Algorithm 2, 5 and 6.

Selecting a weak-learner that provides a fair comparison among different Boosting algorithms is important at this point. SAMME requires multi-class weak-learners while, on the other hand, AdaBoost.MH and PIBoost can use even simple stump-like classifiers. Besides, Multi-category GentleBoost requires the use of regression over continuous variables for computing its weak-learners. We select classification trees as weak-learners for the first three algorithms and regression trees for the latter.

For classification trees the following growing schedule is adopted. Each tree grows splitting impure nodes that present more than $\bar{N}/K$ instances (where $\bar{N}$ is the number of samples selected for fitting the tree), so this value is a lower bound for splitting. We find good results for the sample size parameter when $\bar{N} < 0.4 \cdot N$, where N is the training data size. In particular we fix $\bar{N} = 0.1 \cdot N$ for all data sets. In the case of regression trees the growing pattern is similar but the bound of $\bar{N}/K$ instances for splitting produces poor results. Here more complex trees achieve better performance. In particular when the minimum bound for splitting is $\bar{N}/2K$ instances we got excellent error rates. A pruning process is carried out too in both types of trees.

[5]The original Nursery data set has 2 instances labeled with second class. We omit them in order to apply cross-validation properly.

We experiment with two variants of PIBoost. The first one takes all single labels, $\mathcal{G} = \{\{k\}|\ k \in L\}$, as group of sets to separate while the second one, more complex, takes all single labels plus all pairs of labels, $\mathcal{G}' = \mathcal{G} \cup \{\{k,l\}|\ k \neq l\ ;\ k,l \in L\}$. We must emphasize the importance of selecting a good group of separators in achieving the best performance. Depending on the number of classes, selecting an appropriate set G is a problem on itself. Knowledge of the dependencies among labels sets will certainly help in designing a good set of separators. We leave this issue as future work.

For the experiments we fix a number of iterations that depends on the algorithm and the number of labels of each data set. Since the five algorithms considered in this section fit a different number of weak-learners at each iteration, we select the number of iterations for each one in such a way that all experiments have the same number of weak-learners (see Table 3.5). Remember that, when a data set presents K-labels, PIBoost(2) fits $\binom{K}{2} + K$ separators per iteration while PIBoost(1) and GentleBoost fit only K. Besides SAMME and AdaBoost.MH fit one weak-learner per iteration. In Fig. 3.3 we plot the performance of all five algorithms. The splitting criterion for classification trees is the Gini index.

Data set	GentleBoost	AdaBoost.MH	SAMME	PIBoost(1)	PIBoost(2)	#WL
CarEvaluation (4)	70	280	280	70	40 [7]	280
Chess (18)	95	1710	1710	95	10 [171]	1710
CNAE9 (9)	100	900	900	100	20 [45]	900
Isolet (26)	135	3510	3510	135	10 [351]	3510
Multifeatures (10)	110	1100	1100	110	20 [55]	1100
Nursery (5)	120	600	600	120	40 [15]	600
OptDigits (10)	110	1100	1100	110	20 [55]	1100
PageBlocks (5)	120	600	600	120	40 [15]	600
PenDigits (10)	110	1100	1100	110	20 [55]	1100
SatImage (7)	80	560	560	80	20 [28]	560
Segmentation (7)	80	560	560	80	20 [28]	560
Vehicle (4)	70	280	280	70	40 [7]	280
Vowel (11)	120	1320	1320	120	20 [66]	1320
Waveform (3)	40	120	120	40	40 [3]	120

Table 3.5: Number of iterations considered for each Boosting algorithm. The first column displays the data base name with the number of classes in parenthesis. Columns two to six display the number of iterations of each algorithm. For PIBoost(2) the number of separators per iteration appears inside brackets. The last column displays the number of weak-learners used for each data base.

The performance of a classifier corresponds to that achieved at the last iteration, combining all learned weak classifiers. We evaluate the performance of the algorithms using 5-fold cross-validation. Table 3.6 shows these values and their standard deviations. As can be seen, PIBoost (with its two variants) outperforms the rest of methods in many data sets. Once the algorithms are ranked by accuracy we use the Friedman test to asses whether the performance differences are statistically significant [15]. As was expected the null hypothesis (all algorithms have the same quality) is rejected with a p-value < 0.01. Hence we carry out a post-hoc analysis. We use the Nemenyi test to group the algorithms that present insignificant difference [15]. Figure 3.4 shows the result of the test for both $\alpha = 0.05$ and $\alpha = 0.1$ significance level. Summarizing, PIBoost(1) can be considered as good as PIBoost(2) and also as good as the rest of algorithms, but PIBoost(2) is significantly better than the latter. In addition, we use the Wilcoxon matched-pairs signed-ranks test to asses the statistical significance of the performance in comparisons

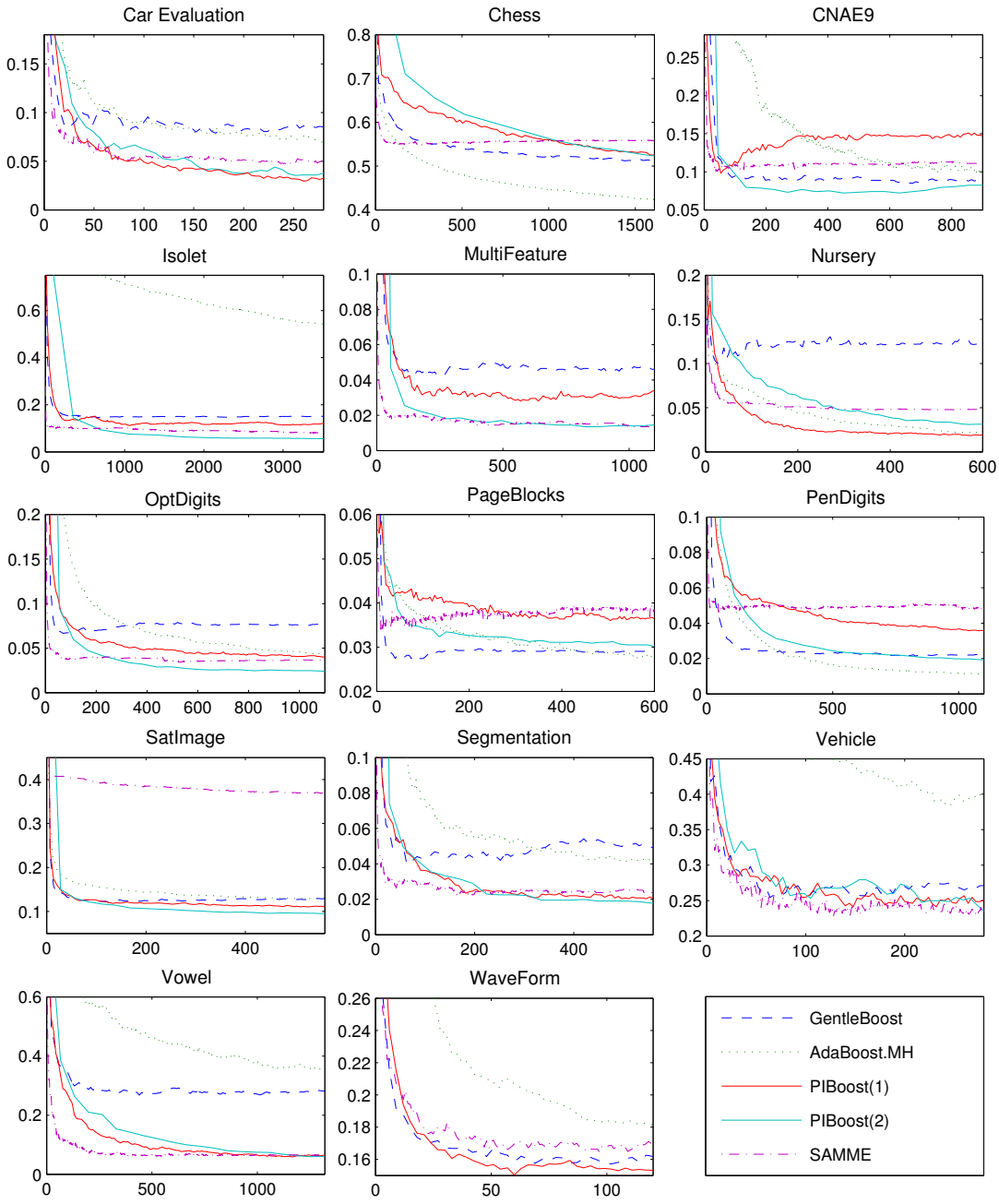


Figure 3.3: Plots comparing the performances of Boosting algorithms. In the vertical axis we display the error rate. In the horizontal axis we display the number of weak-learners fitted for each algorithm.

between pairs of algorithms [15]. Table 3.7 presents the p-values obtained after comparing PIBoost(1) and PIBoost(2) with the others. Again, it is clear that the latter is significantly better than the rest.

Additionally, we have also performed one more experiment with the *Amazon* data base to asses the performance of PIBoost in a problem with a very high dimensional space and with a large number of labels. This data base also belongs to the UCI repository. It has 1500 sample instances with 10000 features grouped in 50 classes. With this data base we follow the same experimental design as with the other data bases, but only use the PIBoost(1) algorithm. In Figure 3.5 we plot the evolution in the performance of each algorithm as the number of weak

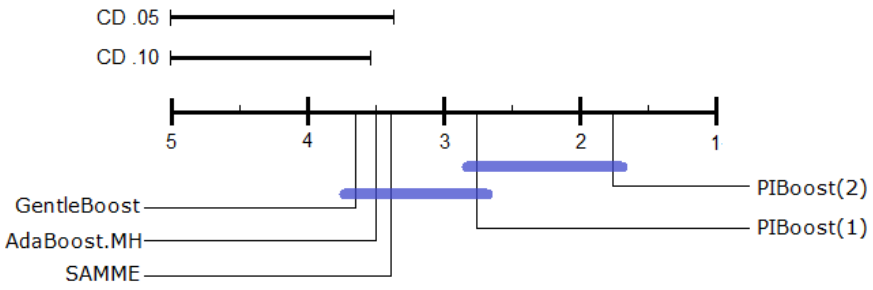


Figure 3.4: Diagram of the Nemenyi test. The average rank for each method is marked on the segment. We show critical differences for both $\alpha = 0.05$ and $\alpha = 0.1$ significance level at the top. We group with thick blue line algorithms with no significantly different performance.

learners increases. At the last iteration, PIBoost(1) has respectively an error rate and a standard deviation of 0.4213 and ($\pm 374 \times 10^{-4}$), whereas Multi-category GentleBoost has 0.5107 and ($\pm 337 \times 10^{-4}$), SAMME 0.6267 and ($\pm 215 \times 10^{-4}$) and, finally, AdaBoost.MH 0.7908 and ($\pm 118 \times 10^{-4}$).

Discussion

The experimental results confirm our initial intuition that by increasing the range of margin values and considering the asymmetries in the class distribution generated by the weak-learners we can significantly improve the performance of Boosting algorithms. This is particularly evident in problems with a large number of classes and few training instances. In the same way, the performance gain is evident in a high dimensional space, see the results for CNAE9, Isolet and, of course, Amazon. Moreover, as can be observed in Table 3.5 and Table 3.6, our second variant of PIBoost produces goods results even when few iterations are computed, see Isolet, OptDigits, SatImage, Segmentation or Vowel.

Data set	**GentleBoost**	**AdaBoost.MH**	**SAMME**	**PIBoost(1)**	**PIBoost(2)**
CarEvaluation	0.0852 (±121)	0.0713 (±168)	0.0487 (±111)	**0.0325** (±74)	0.0377 (±59)
Chess	0.5136 (±61)	**0.4240** (±34)	0.5576 (±63)	0.5260 (±118)	0.5187 (±74)
CNAE9	0.0870 (±239)	0.1028 (±184)	0.1111 (±77)	0.1472 (±193)	**0.0824** (±171)
Isolet	0.1507 (±94)	0.5433 (±179)	0.0812 (±185)	0.1211 (±253)	**0.0559** (±55)
Multifeatures	0.0460 (±128)	0.3670 (±822)	**0.0135** (±44)	0.0340 (±96)	0.0145 (±82)
Nursery	0.1216 (±60)	0.0203 (±32)	0.0482 (±58)	**0.0192** (±29)	0.0313 (±62)
OptDigits	0.0756 (±74)	0.0432 (±59)	0.0365 (±55)	0.0400 (±13)	**0.0240** (±41)
PageBlocks	0.0291 (±52)	**0.0276** (±46)	0.0386 (±87)	0.0364 (±47)	0.0302 (±50)
PenDigits	0.0221 (±11)	**0.0113** (±29)	0.0484 (±62)	0.0358 (±40)	0.0192 (±25)
SatImage	0.1294 (±32)	0.1318 (±51)	0.3691 (±120)	0.1113 (±62)	**0.0949** (±53)
Segmentation	0.0494 (±64)	0.0407 (±88)	0.0238 (±55)	0.0208 (±52)	**0.0177** (±61)
Vehicle	0.2710 (±403)	0.3976 (±297)	**0.2320** (±221)	0.2509 (±305)	0.2355 (±258)
Vowel	0.2818 (±322)	0.3525 (±324)	0.0667 (±114)	0.0646 (±183)	**0.0606** (±160)
Waveform	0.1618 (±75)	0.1810 (±72)	0.1710 (±109)	**0.1532** (±44)	**0.1532** (±44)

Table 3.6: Error rates of GentleBoost, AdaBoost.MH, SAMME, PIBoost(1) and PIBoost(2) algorithms for each data set in table 3.4. Standard deviations appear inside parentheses in 10^{-4} scale. Bold values represent the best result achieved for each data base.

	GentleBoost	**AdaBoost.MH**	**SAMME**	**PIBoost(1)**
PIBoost(2)	0.0012	0.0203	0.0006	0.0081
PIBoost(1)	0.0580	0.1353	0.7148	

Table 3.7: *P*-values corresponding to Wilcoxon matched-pairs signed-ranks test.

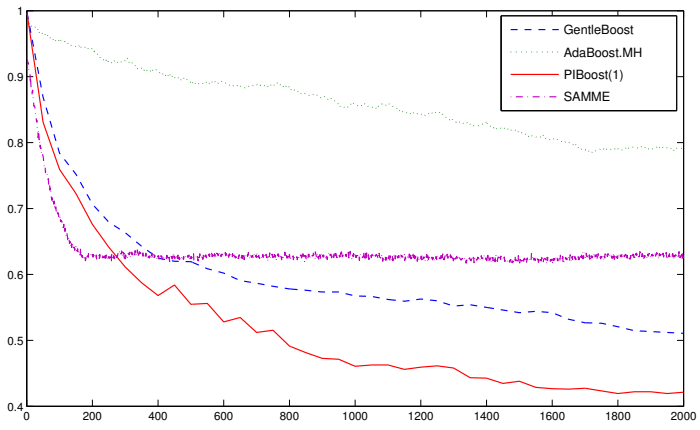


Figure 3.5: Plot comparing the performances of Boosting algorithms for the *Amazon* data base. In the vertical axis we display the error rate. In the horizontal axis we display the number of weak-learners fitted for each algorithm.

Chapter 4

Multi-class Cost-sensitive Boosting

In this chapter we give a second step in our multi-class classification problems by adding a cost matrix. Assigning values for penalizing different types of errors is useful to address some relevant problems. One of them is the need to skew the decision boundaries to reduce the generalization error. It frequently occurs in unbalanced problems, in which the majority classes tend to be favoured by the regular classification rules. The addition of costs is useful to increase the importance of minority classes, hence to correct the decision boundaries. Another important type of problem comes with the ordinal regression. It consists in a classification problem where the labels are obtained as the discretization of a continuous variable and preserving the order is essential. A proper cost matrix may be useful to avoid "distant" failures, which obviously must be more penalized than "closer" ones. Finally, costs can be viewed by themselves as the objective of the classification. This is particularly evident in problems where each decision involves a cost and minimizing it becomes the goal of the algorithm. Such is the case of insurance, banking, or diagnosis applications.

In this chapter we study the addition of a cost matrix to the Boosting framework through the well known exponential loss function. To this aim some algorithms have been proposed but none of them may be considered a canonical extension of AdaBoost to the multi-class cost-sensitive field using multi-class weak learners. We will discuss them in the following section. This topic has been widely studied for binary problems, as was summarized in section 2.4. Now we present an extension to the multi-class field based on a new concept of margin.

The remainder of the chapter is organized as follows. Section 4.1 describes previous theories on multi-class cost-sensitive Boosting. Here we define our new concept of cost-sensitive margin. In section 4.2 we show in detail the structure of our algorithm. This section also presents the main result that supports BAdaCost jointly with some corollaries describing direct generalizations. Finally, in Section 4.3 we show experiments confirming the efficiency of our algorithm.

4.1 Cost-sensitive multi-class Boosting

Let us assume the misclassification costs for our multi-class problem are encoded using a ($K\times K$)-matrix $\mathbf{C}$, where each entry $C(i,j) \geq 0$ measures the cost of misclassifying an instance

with *real* label i when the *prediction* is j,

$$\mathbf{C} = \begin{pmatrix} C(1,1) & C(1,2) & \dots & C(1,K) \\ C(2,1) & C(2,2) & \dots & C(2,K) \\ \vdots & \vdots & \ddots & \vdots \\ C(K,1) & C(K,2) & \dots & C(K,K) \end{pmatrix} .$$

We will expect of this matrix to have costs for correct assignments lower than any wrong classification, i.e. $C(i,i) < C(i,j), \forall i \neq j$. More rigorously, multi-class problems may be affected by costs in an instance-dependent way. In these situations an exclusive row of costs, $\mathbf{C}_n \in \mathbb{R}^K$, is associated to each instance $(\mathbf{x}_n, l_n)$, $n = 1, \dots, N$ [58]. Obviously, using a cost matrix is a special case, since all samples with the same label would share the same penalizing row. We will not address this kind of problems in our study due to the lack of real applications.

Let us introduce some intuitive notations for the remainder of the chapter. Hereafter, $\mathbf{M}(j,-)$ and $\mathbf{M}(-,j)$ will be used for referring to the j-th row and column vector of a matrix $\mathbf{M}$. Again $I(\cdot)$ will denote the indicator function (1 when argument is true, 0 when false).

For cost-sensitive problems the regular Bayes Decision Rule is not suitable, since the label with maximum a posteriori probability may present a high cost. Rather a costs-dependent criterion is applied. If $\mathbf{P}(\mathbf{x}) = (P(1|\mathbf{x}), \dots, P(K|\mathbf{x}))^\top$ is the vector of a posteriori probabilities for a given $\mathbf{x} \in X$, then the Cost-sensitive Bayes Decision rule is

$$F(\mathbf{x}) = arg\min_{j \in L} \mathbf{P}(\mathbf{x})^\top \mathbf{C}(-,j) \,, \tag{4.1}$$

which is but the minimizer of the risk function $\mathcal{R}(\mathbf{P}(\mathbf{x}), \mathbf{C}(-,j)) := \mathbf{P}(\mathbf{x})^\top \mathbf{C}(-,j)$ with regard to $j \in L$.

When dealing with multi-class cost-sensitive problems one has to understand how the addition of a cost matrix influences the decision boundaries. For this requirement, we recommend O'Brien's et al. work [69]. It displays a concise glossary of linear algebra operations on a cost matrix and their respective effects over decision boundaries. Let

$$\sum_{h=1}^{K} C(h,i) P(i|\mathbf{x}) = \sum_{h=1}^{K} C(h,j) P(j|\mathbf{x}) \tag{4.2}$$

be the decision boundary between classes i and j, with $i \neq j$. Here we describe the ones we will use:

1. The decision boundaries are not affected when $\mathbf{C}$ is replaced by $\alpha\mathbf{C}$, for any $\alpha > 0$.
2. Adding a constant to all costs for a particular true class does not affect the final decision. In other words, adding a positive value to a row $C(i,-)$ maintains the result unaffected.
3. As consequence of the previous property, any cost matrix $\mathbf{C}$ can be replaced by an equivalent $\hat{\mathbf{C}}$ with $\hat{C}(i,i) = 0, \forall i$.

Proving each of them is immediate just by plugging each variant in the expression (4.2). Taking into account the last property we will assume without loss of generality that $C(i,i) = 0, \forall i \in L$, i.e. the cost of correct classifications is null. We will denote $0|1$-*matrix* to that with zeros in its diagonal and ones elsewhere (in other words, a matrix representing a pure multi-class problem).

Let us focus our attention on the meaning of a cost matrix depending on its symmetry. We may have:

- *Symmetric matrix*. Since symmetrical values are equal, no additional information is provided when mistaking in one direction or another. This means that the actual information lies in comparing the costs associated to different decision boundaries, which can be ranked according to their importance. Hence this structure is recommended for problems where some boundaries are more important than others. In Graph Theory this kind of matrix would represent an undirected complete graph with different distances between nodes (labels).

- *Asymmetric matrix*. This regular case is the appropriate for situations where some labels are more important than others. It is useful when the problem at hand presents unbalanced data or simply when we are interested in avoiding some types of mistakes (possibly the most usual ones). In Graph Theory this case would represent a directed complete graph with paths of different module even between pairs of nodes (labels).

In the following we will consider essentially this last case.

4.1.1 Previous works

Cost-sensitive classification problems for more than two labels had never been easy to translate to the area of Boosting. Following an initial intuition, one may try to decompose the problem into binary ones and then apply any algorithm described in section 2.4. However this can be a bad choice since the (2×2)-matrix associated to each subproblem may be undefined or even unuseful for the global problem. For instance, think of separating a label from the rest, but there is no justified way to compose the associated binary cost matrix. Furthermore, if the global matrix is symmetric and the idea is separating one label from another (One-vs-One strategy) then every subproblem would be equally important, i.e. every submatrix would be equivalent to a binary cost-free problem.

There are several works in the literature that address the cost-sensitiveness of a problem in a paradigm-independent framework [21, 18, 111, 56, 106]. We will not consider these cases since we are interested in introducing costs in the multi-class Boosting context. The contributions conceived for this purpose are:

- *AdaC2.M1* [86]. The algorithm developed by Y. Sun et al. is probably the first including costs when using multi-class weak learners. The idea behind this is combining the multi-class structure of AdaBoost.M1 [26], with the weighting rule of AdaC2 [89] (see Algorithm 8), hence its name. As can be guessed, no theoretical derivation supports this method. Rather it becomes just an heuristic procedure for merging both extensions of AdaBoost into one. AdaC2.M1's pseudo-code is shown in Algorithm 11. As its multi-class counterpart it fails in computing α-values only available for "not too weak" learners, see step **2- (c)**. Moreover, they comprise the information of the row concerning to a real label l into a single value, $C_l = \sum_{j=1}^{K} C(l, j)$, which misses the structure of the given cost matrix (there are infinite different cost matrices producing the same values C_l, therefore representing the "same problem" for the algorithm).

- *L_p-CSB* [58] This algorithm was originally conceived to solve instance-dependent cost-sensitive problems. See its pseudo-code in Algorithm 12. The authors, A. C. Lozano and N. Abe, provided a new insight for solving the cost-sensitive problem. They resorted

Algorithm 11 :AdaC2.M1

1- Initialize the Weight Vector $\mathbf{W}$ with uniform distribution $w(n) = 1/N\ ,\ n = 1, \ldots, N$.
2- For $m = 1$ to M:
(a) Fit a multi-class classifier $G_m(\mathbf{x})$ to the training data using weights $\mathbf{W}$.
(b) Compute weighted error: $Err_m = \sum_{n=1}^{N} w(n) I\left(G_m(\mathbf{x}_n) \neq l_n\right)$.
(c) Compute $\alpha_m = \frac{1}{2}\log\left((1 - Err_m)/Err_m\right)$.
(d) Update weight vector $w(n) \leftarrow w(n) C_{l_n} \exp\left(-\alpha_m I\left(G_m(\mathbf{x}_n) = l_n\right)\right), n = 1, \ldots, N$.
(e) Re-normalize $\mathbf{W}$.
3- Output Final Classifier: $H(\mathbf{x}) = \arg\max_k \sum_{m=1}^{M} \alpha_m I(G_m(\mathbf{x}) = k)$.

to relational hypothesis, $h : X \times L \rightarrow [0, 1]$, satisfying the stochastic condition, $\forall \mathbf{x} \in X\ , \sum_{l \in L} h(l|\mathbf{x}) = 1$, to solve the minimization

$$\arg\min_h \frac{1}{N} \sum_{n=1}^{N} C(l_n, \arg\max_k h(k|\mathbf{x}_n))\ , \tag{4.3}$$

which obviously is the goal of the problem (minimize the expected cost for an uniform distribution of instances). Since the above function is not convex, they remedied this drawback by translating every term in (4.3) into

$$C(l_n, \arg\max_k h(k|\mathbf{x}_n)) = \sum_{k=1}^{K} \left(\frac{h(k|\mathbf{x}_n)}{\|h(k|\mathbf{x}_n)\|_\infty}\right)^\infty C(l_n, k) \tag{4.4}$$

that, in turn, can be approximated with a p-norm ($p \geq 1$) as follows

$$C(l_n, \arg\max_k h(k|\mathbf{x}_n)) \simeq \sum_{k=1}^{K} \left(\frac{h(k|\mathbf{x}_n)}{\max_j h(j|\mathbf{x}_n)}\right)^p C(l_n, k)\ . \tag{4.5}$$

Since $\max_j h(j|\mathbf{x}_n)$ will be grater or equal than $1/K$, then expression (4.3) can be approximated by the following convexification:

$$\arg\min_h \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} h(k|\mathbf{x}_n)^p C(l_n, k)\ , \tag{4.6}$$

which becomes the aim of the Boosting algorithm. Then an extended-data approach is followed, just like AdaBoost.MH [80], for stochastic hypothesis. Equation (4.7) shows the reweighting scheme applied. This time the voting constants, α, form a convex linear combination of weak learners (see **2- (d)**). We must point out that L_p-CSB generalizes a previous work of N. Abe et al. [1] where a *Data Space Extension* is proposed jointly with an *Iterative Weighting* scheme to derive the GBSE algorithm (Gradient Boosting with Stochastic Ensembles). Specifically, the latter becomes a particular case of L_p-CSB when $p = 1$.

A minor drawback when applying L_p-CSB comes with the selection of the optimal value p for the norm. No clear foundation has been argued for considering values like 3 or 4, as can be seen in the experiments of [58].

Algorithm 12 :L_p-CSB

1- Initialize H_0 with uniform distribution $H_0(k|\mathbf{x}_n) = 1/K$, $\forall n, \forall k$.
2- Set the expanded labeled data $\bar{S} = \{\big((\mathbf{x}_n, k),\ \bar{l}_{n,k}\big) \mid k \in L\ ,\ \bar{l}_{n,k} := I(k = l_n)\}$.
3- For $m = 1$ to M:
(a) Set $w(\mathbf{x}, k) = H_{m-1}(k|\mathbf{x})^{p-1}C(\mathbf{x}, k), \forall (\mathbf{x}, k) \in \bar{S}$.
(b) For all $(\mathbf{x}_n, k) \in \bar{S}$ compute:

$$\bar{w}(\mathbf{x}_n, k) = \begin{cases} w(\mathbf{x}_n, k)/2 & \text{if } \bar{l}_{n,k} = 0 \\ (\sum_{j \neq l} w(\mathbf{x}_n, j))/2 & \text{if } \bar{l}_{n,k} = 1 \end{cases} \tag{4.7}$$

(c) Compute relational hypothesis h_m on $\bar{S}$ with regard to weights $\bar{\mathbf{W}}$.
(d) Choose $\alpha_m \in [0, 1)$, for example $\alpha_m = 1/m$.
(e) Set $H_m(\mathbf{x}) := (1 - \alpha_m)H_{m-1}(\mathbf{x}) + \alpha_m h_m(\mathbf{x})$.
4- Output Final Classifier: $H(\mathbf{x}) = \arg\max_k H_M(k|\mathbf{x})$.

- *MultiBoost* [102]. The most recent derivation and, under our point of view, the closer to AdaBoost's essence. The algorithm proposed by J. Wang resorts to margin vectors, $\mathbf{y}\ ,\ \mathbf{g}(\mathbf{x}) \in Y$; and a special derivative of the exponential loss function

$$\mathcal{L}(l, \mathbf{f}(\mathbf{x})) := \sum_{k=1}^{K} C(l, k) \exp(f_k(\mathbf{x}))\ , \tag{4.8}$$

to carry out a gradient descent search. The structure of the additive model fitted in this fashion is similar to the one obtained with SAMME [112]. MultiBoost's pseudo-code is shown in Algorithm 13. It is clear that the loss defined in (4.8) does not coincide with any other loss function when a $0|1$-cost matrix is used.

The main result supporting MultiBoost's theory states that the optimal pair $(T_m(\mathbf{x}), \beta_m)$ to add to the additive model[1] $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$ is found by solving:

$$T_m(\mathbf{x}) = \arg\min_T \sum_{n=1}^{N} A(n, T(\mathbf{x}_n))\ , \tag{4.9}$$

$$\beta_m = \frac{K-1}{K}\left(\log\left(\frac{1 - Err}{Err}\right) - \log(K-1)\right)\ . \tag{4.10}$$

Where $Err = \sum_{n=1}^{N} A(i, T(\mathbf{x}_n))$ and constants $A(i, k)$ compose a $(N \times K)$-matrix, $\mathbf{A}$, of values derived from costs (see step **2- (d)**). Obviously, each of the above optimal parameter depends on the other. Since there is no direct way to solve (4.9), it is convenient to have of a pool of weak learner from which to select the optimal. It becomes the best strategy to accomplish step **2- (a)**.

Let us go back to the most relevant binary cost-sensitive Boosting algorithms discussed in section 2.4. On the one hand, as far as we know, Cost-Sensitive AdaBoost [63] has not found a direct generalization to the multi-class field. We will show in section 4.2.1 how our new algorithm BAdaCost accomplishes it. On the other hand Cost-Generalized AdaBoost [49]

[1]Remember the equivalence between multi-class weak learners, $T(\mathbf{x})$, defined on L and those defined over margin values, $\mathbf{g}(\mathbf{x})$. See (2.16)

Algorithm 13 :MultiBoost

1- Initialize constants $A(n,j) = C(l_n, j)$, $n = 1, \ldots, N;\ \ k = 1, \ldots, K$ and normalize $\mathbf{A}$.
2- For $m = 1$ to M:
(a) Solve $(T_m(\mathbf{x}), \beta_m)$ in expressions (4.9) and (4.10).
(b) Translate $T_m(\mathbf{x})$ in terms of margin vectors $\mathbf{g}_m(\mathbf{x})$.
(c) Update the additive model: $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$.
(d) Update constants $A(n,j) \leftarrow A(n,j) \exp\left(\beta_m g_m^j(\mathbf{x}_n)\right)$, $n = 1, \ldots, N;\ j = 1, \ldots, K$.
(e) Re-normalize $\mathbf{A}$.
3- Output Final Classifier: $H(\mathbf{x}) = \arg\max_k f_k(\mathbf{x})$; for $\mathbf{f}(\mathbf{x}) = \sum_{m=1}^{M} \beta_m \mathbf{g}_m(\mathbf{x})$.

resorts to the initial reweighting discussed in section 2.4 to induce the cost-sensitive property to the original AdaBoost. Extending this process to multi-class problems seems unclear since it requires assigning a different initial weight to each possible kind of error, which is impossible for just one weight vector.

4.1.2 New margin for cost-sensitive classification

Here we introduce the essence of our second algorithm. We firstly define the concept of multi-class cost-sensitive margin, which serves as link between multi-class margins and the values of the cost matrix, both of them considered as argument of a loss function.

With this in mind, we introduce an essential change in the multi-class exponential loss function. Let $\mathbf{C}^*$ be a $(K \times K)$-matrix defined in the following way

$$C^*(i,j) = \begin{cases} C(i,j) & \text{if } i \neq j \\ -\sum_{h=1}^{K} C(i,h) & \text{if } i = j \end{cases}, \ \forall i,j \in L\,, \tag{4.11}$$

i.e. $\mathbf{C}^*$ is obtained from $\mathbf{C}$ replacing the j-th zero in the diagonal for the sum of the elements in the j-th row with negative sign. For our cost-sensitive classification problem each value $C^*(j,j)$ will represent a "minus cost" associated to a type of correct classification. In other words, elements in the diagonal should be understood as prizes for successes over instances with the same real label.

Notice that, by definition, the j-th row in $\mathbf{C}^*$ is a margin vector that encodes the cost structure of the j-th label. This motivates us to use these rows as the *new set of vectors for encoding true labels*, i.e. we will keep in mind the bijection: $l \leftrightarrow \mathbf{C}(l,-)$. Notice also that our vectors are neither equidistant nor have equal norm. In other words, it serves to reorient the space of labels following the structure of the cost matrix.

Based on the above codification, we define the *Multi-class Cost-sensitive Margin value* for an instance $(\mathbf{x}, l)$ with respect to the multi-class vectorial classifier $\mathbf{f}(\mathbf{x})$ as

$$z_C := \mathbf{C}^*(l,-) \cdot \mathbf{f}(\mathbf{x})\ . \tag{4.12}$$

Analogously to expression (2.18), it is easy to verify that if $\mathbf{f}(\mathbf{x}) = \mathbf{y}_j \in Y$, for a certain $j \in L$, then $\mathbf{C}^*(l,-) \cdot \mathbf{f}(\mathbf{x}) = \frac{K}{K-1}\mathbf{C}^*(l,j)$. Therefore the multi-class cost-sensitive margins obtained from a discrete classifier $\mathbf{f} : X \to Y$ can be calculated directly using the label-valued analogous of $\mathbf{f}$, i.e. $F : X \to L$, through the formula

$$z_C = \mathbf{C}^*(l,-) \cdot \mathbf{f}(\mathbf{x}) = \frac{K}{K-1}\mathbf{C}^*(l, F(\mathbf{x}))\,. \tag{4.13}$$

As a consequence when considering a lineal combination of discrete classifiers $\mathbf{f} = \sum_{m=1}^{M} \alpha_m \mathbf{f}_m$ the following expression can be applied:

$$z_C = \sum_{m=1}^{M} \alpha_m \mathbf{C}^*(l, -) \cdot \mathbf{f}_m(\mathbf{x}) = \frac{K}{K-1} \sum_{m=1}^{M} \alpha_m \mathbf{C}^*(l, F_m(\mathbf{x})) \,. \tag{4.14}$$

Our aim is to use this value as argument for the multi-class exponential loss function in order to obtain the *Cost-sensitive Multi-Class Exponential Loss Function*, that is defined as follows:

$$\mathcal{L}_C(l, \mathbf{f}(\mathbf{x})) := \exp(z_C) = \exp\left(\mathbf{C}^*(l, -) \cdot \mathbf{f}(\mathbf{x})\right) \,. \tag{4.15}$$

It will be the loss function for our problem. See that z_C yields negative values when classifications are good under the cost-sensitive point of view, while positive values come from costly asignments. That is why $\mathcal{L}_C$ does not need a negative sign in the exponent, which was the case of previous exponential loss functions. This is a key point in our proposal.

Let us see now the suitability of our new loss function. Specifically, let us discuss how it extends the loss functions in CS-AdaBoost and SAMME respectively:

$$\mathbf{A)}\ \mathcal{L}_{CS-AdaBoost}(l, f(\mathbf{x})) = I(l=1)\exp\left(-lC_1 f(\mathbf{x})\right) + I(l=-1)\exp\left(-lC_2 f(\mathbf{x})\right)$$

$$\mathbf{B)}\ \mathcal{L}_{SAMME}(\mathbf{y}, \mathbf{f}(\mathbf{x})) = \exp\left(\frac{-1}{K}\mathbf{y}^\top \mathbf{f}(\mathbf{x})\right) \,.$$

Proving how each of them becomes a special case of ours is quite simple. On the one hand $\mathcal{L}_{CS-AdaBoost}$ is exactly the result of (4.15) when applied on a binary cost-sensitive problem, where off-diagonal values have been denoted: $C_1 := C(1,2)$ and $C_2 := C(2,1)$. On the other hand, we can obtain $\mathcal{L}_{SAMME}$ just by fixing a $0|1$-cost matrix re-scaled by a factor $\lambda > 0$, i.e. $\lambda \mathbf{C}_{0|1}$. Specifically, when $\lambda = 1/K(K-1)$ we get exactly the same values provided by SAMME's margin vectors (see 2.19). Hence BAdaCost's loss functions generalizes its multi-class counterpart. In subsection 4.2.1 we will state complete results about this capability of generalization.

4.2 BAdaCost: Boosting Adapted for Cost-matrix

In this section we introduce BAdaCost [25], that stands for *Boosting Adapted for Cost-matrix*. As we have defined the cost-sensitive multi-class exponential loss function and given a training sample $\{(\mathbf{x}_n, l_n)\}$ we minimize the empirical expected loss,$\sum_{n=1}^{N} \mathcal{L}_C(l_n, \mathbf{f}(\mathbf{x}_n))$, to obtain the new Boosting algorithm. Once more, the minimization is carried out by fitting an additive model, $\mathbf{f}(\mathbf{x}) = \sum_{m=1}^{M} \beta_m \mathbf{g}_m(\mathbf{x})$. The weak learner selected at each iteration m will consists of an optimal step of size β_m along the direction $\mathbf{g}_m$ of the largest descent of the expected cost-sensitive multi-class exponential loss function. In Lemma 2 we show how to compute them.

Lemma 2. *Let* $\mathbf{C}$ *be a cost matrix for a multi-class problem. Given the additive model* $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$ *the solution to*

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg\min_{\beta, \mathbf{g}} \sum_{n=1}^{N} \exp\left(\mathbf{C}^*(l_n, -)\left(\mathbf{f}_{m-1}(\mathbf{x}_n) + \beta \mathbf{g}(\mathbf{x}_n)\right)\right) \tag{4.16}$$

is the same as the solution to

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg\min_{\beta,\mathbf{g}} \sum_{j=1}^{K} \left(S_j \exp\left(\beta C^*(j,j)\right) + \sum_{k\neq j} E_{j,k} \exp\left(\beta C^*(j,k)\right) \right) , \quad (4.17)$$

where $S_j = \sum_{\{n:\mathbf{g}(x_n)=l_n=j\}} w(n)$, $E_{j,k} = \sum_{\{n:l_n=j,\mathbf{g}(x_n)=k\}} w(n)$, *and the weight of the* n*-th training instance is given by:*

$$w(n) = \exp\left(\mathbf{C}^*(l_n,-) \sum_{t=1}^{m-1} \beta_m \mathbf{f}_m(\mathbf{x}_n) \right) . \quad (4.18)$$

Given a known direction $\mathbf{g}$*, the optimal step* β *can be obtained as the solution to*

$$\sum_{j=1}^{K} \sum_{k\neq j} E_{j,k} C(j,k) A(j,k)^{\beta} = \sum_{j=1}^{K} \sum_{h=1}^{K} S_j C(j,h) A(j,j)^{\beta} , \quad (4.19)$$

where $A(j,k) = \exp(C^*(j,k))$, $\forall j,k \in L$. *Finally, given a known* β*, the optimal descent direction* $\mathbf{g}$*, equivalently* G*, is given by*

$$\arg\min_{G} \sum_{n=1}^{N} w(n) \left(A(l_n,l_n)^{\beta} I(G(\mathbf{x}_n)=l_n) + \sum_{k\neq l_n} A(l_n,k)^{\beta} I(G(\mathbf{x}_n)=k) \right) . \quad (4.20)$$

The proof of this result is in the Appendix A.3. The BAdaCost pseudo-code is shown in Algorithm 14. Just like other Boosting algorithms we start weights with uniform distribution. At each iteration, we add a new multi-class weak learner $\mathbf{g}_m : X \to Y$ to the additive model weighted by β_m, a measure of the confidence in the prediction of $\mathbf{g}_m$. The optimal weak learner that minimizes (4.20) is a cost-sensitive multi-class classifier trained using the data weights, $w(n)$, and a modified cost matrix, $\mathbf{A}^{\beta} = \exp(\beta \mathbf{C}^*)$. Observe that training a cost-sensitive weak-learner does not imply computational difficulties. For instance, when using classification trees one can proceed in three ways [38]: adjusting decision tresholds, changing the split criteria of impurity, or applying a cost-sensitive pruning. We simply recommend computing the cost-sensitive counterparts of the *Gini index* or the *entropy* as splitting criteria when fitting trees.

Algorithm 14 : BAdaCost

1- Initialize weight vector $\mathbf{W}$, $w(n) = 1/N$; for $n = 1, \ldots, N$.
2- Compute matrices $\mathbf{C}^*$, following equation (4.11), and $\mathbf{A}$ for the given $\mathbf{C}$.
3- For $m = 1$ to M:
  (a) Obtain G_m by minimizing (4.20) for $\beta = 1$. Translate G_m into $\mathbf{g}_m : X \to Y$.
  (b) Compute the constants $E_{j,k}$ and S_j, $\forall j,k$; as described in Lemma 2.
  (c) Compute β_m solving equation (4.19).
  (d) Update weights $w(n) \leftarrow w(n) \exp\left(\beta_m \mathbf{C}^*(l_n,-)\mathbf{g}_m(\mathbf{x}_n)\right)$, for $n = 1, \ldots, N$.
  (e) Re-normalize vector $\mathbf{W}$.
4- Output Classifier: $H(\mathbf{x}) = \arg\min_k \mathbf{C}^*(k,-)\mathbf{f}(\mathbf{x})$, where $\mathbf{f}(\mathbf{x}) = \sum_{m=1}^{M} \beta_m \mathbf{g}_m(\mathbf{x})$.

Unlike other Boosting algorithms [28, 112], here $\mathbf{g}_m$ and β_m, can not be optimized independently. As with PIBoost, we solve this by alternatively fixing the value of one of them and optimizing for the other. It is performed by specifying a seed value of β_m (1 for simplicity) and then computing first $\mathbf{g}_m$, and then β_m, consecutively. Otherwise, a convenient loop should be

included to encompass steps **3- (a)**, **(b)**, and **(c)**; in order to ensure a desired improvement in the cost of classification. Once the improvement condition is satisfied, the optimal pair ($\mathbf{g}_m$,β_m) would be added to the model. As with PIBoost, we have empirically confirmed that there are no significant differences if one performs a single iteration of the loop instead of many. Thus, we proceed as described in Algorithm 14, that is quite more efficient.

Finally, let us justify step **3-** comparing it with multi-class approaches. It is well known that vectorial classifiers, $\mathbf{f}(\mathbf{x})$, provide a *degree of confidence* for classifying sample $\mathbf{x}$ into every class. Hence, the max rule, $\arg\max_k f_k(\mathbf{x})$, can be used for label assignment [112, 24]. It is straightforward to see that this criterion is equivalent to assigning the label that maximizes the multi-class margin, $\arg\max_k \mathbf{y}_k^\top \mathbf{f}(\mathbf{x}) = \arg\min_k -\mathbf{y}_k^\top \mathbf{f}(\mathbf{x})$. Since $-\mathbf{y}_k^\top \mathbf{f}(\mathbf{x})$ is proportional to $\mathbf{C}^*_{0|1}(k,-)^\top \mathbf{f}(\mathbf{x})$, we can extend the decision rule to the cost-sensitive field just by assigning $\arg\min_k \mathbf{C}^*(k,-)\mathbf{f}(\mathbf{x})$.

4.2.1 Direct generalizations

We want to remark that our multi-class algorithm is a canonical generalization of previous Boosting algorithms. Specifically we take into account three canonical generalizations of AdaBoost under different contexts. First, Cost-Sensitive AdaBoost [63] is a canonical extension for cost-sensitive binary problems. In second place, SAMME [112], the best-grounded extension of AdaBoost to the multi-class field using multi-class weak learners. Finally we also generalize PIBoost [24], our previously presented multi-class algorithm.

The following Corollaries of Lemma 2 prove these canonical extension results. Proofs are shown in A.4, A.5 and A.6, respectively.

Corollary 1. *When $C(i,j) = \frac{1}{K(K-1)}$, $\forall i \neq j$, then the above result is equivalent to SAMME. The update for the additive model $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$ is given by*

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg\min_{\beta,\mathbf{g}} \sum_{n=1}^{N} \exp\left(-\mathbf{y}_n^\top \left(\mathbf{f}_{m-1}(\mathbf{x}_n) + \beta \mathbf{g}(\mathbf{x}_n)\right)\right)$$

and both optimal parameters can be computed in the following way

- $\mathbf{g}_m = \arg\min_{\mathbf{g}} \sum_{n=1}^{N} w(n) I\left(\mathbf{g}(\mathbf{x}_n) \neq \mathbf{y}_n\right)$
- $\beta_m = \frac{(K-1)^2}{K}\left(\log\left(\frac{1-E}{E}\right) + \log\left(K-1\right)\right)$,

where E is the sum of all weighted errors.

Corollary 2. *When $K = 2$ the Lemma 2 is equivalent to the Cost-sensitive AdaBoost. If we denote $C(1,2) = C_1$ and $C(2,1) = C_2$, the update (4.17) for the additive model $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m G_m(\mathbf{x})$ becomes:*

$$(\beta_m, G_m(\mathbf{x})) = \arg\min_{\beta,G} \sum_{\{l_n=1\}} w(n) \exp\left(-C_1 \beta G(\mathbf{x}_n)\right) + \sum_{\{l_n=2\}} w(n) \exp\left(C_2 \beta G(\mathbf{x}_n)\right) .$$

For a certain value β the optimal $G_m(\mathbf{x})$ is given by

$$\arg\min_{G} \left(\mathrm{e}^{\beta C_1} - \mathrm{e}^{-\beta C_1}\right) b + \mathrm{e}^{-\beta C_1} T_1 + \left(\mathrm{e}^{\beta C_2} - \mathrm{e}^{-\beta C_2}\right) d + \mathrm{e}^{-\beta C_2} T_2 \,,$$

where[2] *:* $T_1 = \sum_{\{n:l_n=1\}} w(n)$*,* $T_2 = \sum_{\{n:l_n=2\}} w(n)$*,* $b = \sum_{\{n:G(\mathbf{x}_n)\neq l_n=1\}} w(n)$ *and* $d = \sum_{\{n:G(\mathbf{x}_n)\neq l_n=2\}} w(n)$*. Given a known direction,* $G(\mathbf{x})$*, the optimal step* β_m *can be calculated as the solution to*

$$2C_1 b\cosh(\beta C_1) + 2C_2 d\cosh(\beta C_2) = T_1 C_1 \mathrm{e}^{-\beta C_1} + T_2 C_2 \mathrm{e}^{-\beta C_2} \ .$$

Corollary 3. *When using margin vectors to separate a group of* s*-labels,* $S \in \mathcal{P}(L)$*, from the rest, the result of the Lemma 2 is equivalent to PIBoost. The update for each additive model built in this fashion,* $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$*, becomes:*

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg\min_{\beta,\mathbf{g}} \sum_{n=1}^{N} w(n) \exp\left(\frac{-\beta}{K} \mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n)\right) \ .$$

For a certain value β *the optimal step* $\mathbf{g}_m(\mathbf{x})$ *is given by*

$$\arg\min_{\mathbf{g}} \left(\mathrm{e}^{\frac{\beta}{s(K-1)}} - \mathrm{e}^{\frac{-\beta}{s(K-1)}}\right) E1 + \mathrm{e}^{\frac{-\beta}{s(K-1)}} A_1 + \left(\mathrm{e}^{\frac{\beta}{(K-s)(K-1)}} - \mathrm{e}^{\frac{-\beta}{(K-s)(K-1)}}\right) E2 + \mathrm{e}^{\frac{-\beta}{(K-s)(K-1)}} A_2 \ ,$$

where: $A_1 = \sum_{\{n:l_n\in S\}} w(n)$*,* $A_2 = \sum_{\{n:l_n\notin S\}} w(n)$*,* $E1 = \sum_{\{n:G(x_n)\neq l_n\in S\}} w(n)$ *and* $E2 = \sum_{\{n:G(x_n)\neq l_n\notin S\}} w(n)$*.*

Besides, known a direction $\mathbf{g}(\mathbf{x})$*, the optimal step* β_m *can be calculated as* $\beta_m = s(K-s)(K-1)\log R$*, where* R *is the only real positive root of the polynomial*

$$P_m(x) = E1(K-s)x^{2(K-s)} + E2sx^K - s(A_2 - E2)x^{(K-2s)} - (K-s)(A_1 - E1) \ .$$

4.3 Experiments

It is time to evaluate the performance of BAdaCost on two relevant kinds of problems. To this aim, we devote a first subsection to introduce our procedure to compute cost matrices. Subsequently, in 4.3.2, we apply our method to some real data sets belonging to the UCI repository. In this first case, minimizing costs will be the goal of classification. Secondly, we compare BAdaCost on a complex Computer Vision problem 4.3.3, the detection of synapses and mitochondria on medical images. Here the adding of a cost matrix will serve as a tool to deal with unbalanced data sets.

4.3.1 Cost matrix construction

When working with cost-sensitive problems one should have reliable information about the associated penalties for any pair of labels. In Decision Theory this is obtained directly asking an expert in the area. This can be expensive, time-consuming or even impossible to do. That is why having a real cost matrix seldom happens. Unfortunately, as far as we know, no general procedure has been proposed in the literature to solve this problem. For instance, a typical way to evaluate cost-sensitive algorithms is to generate random cost matrices. A process like this could be misleading for the final result since these costs do not follow a reasonable design, thus it can yield meaningless results from the point of view of the decision boundaries between labels.

[2]Here we adopt the notation used in [63].

In the same way, when using a cost-sensitive algorithm for solving unbalanced problem, the selection of a suitable cost matrix, $\mathbf{C}$, becomes essential. In [86] the authors encoded the cost matrix in a K-element vector and use a genetic algorithm to optimize classification performance. The drawbacks of this approach are both the computational complexity and the loss of information incurred by "vectorizing" the cost matrix. The procedure we introduce computes a set of costs that punishes more heavily errors in the minority classes. A straightforward solution would be to set the costs inversely proportional to the class unbalance ratios [33]. This solution is not satisfactory since costs are not only related to the relative number of samples in each class, but also to the complexity of the classification problem, i.e. the amount of class overlapping, within-class unbalance, etc. Rather, we find reasonable to infer this information from the confusion matrix obtained with a standard cost-free classifier.

Bearing both types of problems in mind, we include a pre-process for measuring the hardness of the problem with regard to each pair of labels that is also appropriate for unbalanced data. It consists in running a simple cost-insensitive multi-class algorithm on the training data (for instance, BAdaCost with a $0|1$-cost matrix for few iterations), and then compute the associated confusion[3] matrix, $\mathbf{F}$. This matrix becomes extraordinarily informative to describe the overall difficulty of the problem. More precisely, it serves to measure properly the degree of overlapping between labels, and consequently allows to focus on the most relevant boundaries. Using this information, we proceed as follows.

Let $\mathbf{F}^*$ be the matrix obtained when dividing the i-th row, $\mathbf{F}(i,-)$, of $\mathbf{F}$ by $\sum_j F(i,j)$, i.e. the number of samples in class i. Consequently, $F^*(i,j)$ is the proportion of data in class i classified as j. These values collect the degree of overlapping with regard to each real label i. Hence, high coefficients will be assigned to hard decision boundaries. Secondly, we transform $\mathbf{F}^* = \mathbf{F}^*/max_{i,j}F^*(i,j)$ to obtain a matrix with maximun coefficient equal to 1. Then we set $F^*(i,i) = 0, \forall i$, for the diagonal values. Note that in a complex and unbalanced data set, a $0|1$-loss classifier will tend to over-fit the majority classes. So, off-diagonal elements in rows $\mathbf{F}^*(i,-)$ for majority (alt. minority) classes will have low (high) scores.

If needed, off-diagonal zero values should be replaced by a small $\epsilon > 0$. Doing so we guarantee that only correct classifications have null costs. Finally, to improve numerical conditioning, we rescale the resulting matrix, $\mathbf{C} = \lambda\mathbf{F}^*$, for an appropriate $\lambda > 0$. We can do this transformation based on the fact that any cost matrix can be multiplied by a positive constant without affecting the output labels of the problem [69].

4.3.2 Minimizing costs: UCI repository

To asses the performance of BAdaCost on real problems we resort again to the UCI catalogue of Machine Learning problems. In this first comparison we are interested in measuring the capability to minimize costs. A group of multi-class supervised data sets are selected, just like we did in section 3.4. This time we have to be sure that the selected data represent complex problems for the process to be meaningful. The chosen data bases are: *CarEvaluation*, *Chess*, *CNAE9*, *ContraMethod*, *Isolet*, *Letter*, *Shuttle*, *OptDigits*, *PenDigits*, *SatImage*, *Segmentation*, and *Waveform*. They collect a broad scope of classification problems with regard to number of variables (6 to 856), labels (3 to 26) and instances (1080 to 58000). Table 4.1 shows a description of the selected ones.

We compare BAdaCost with the algorithms introduced in section 4.1.1: AdaC2.M1 [86] ,

[3]This matrix is also known as *contingency* matrix.

Data set	Variables	Labels	Instances
CarEvaluation	6	4	1728
Chess	6	18	28056
CNAE9	856	9	1080
ContraMethod	9	3	1473
Isolet	617	26	7797
Letter	16	26	20000
Shuttle	9	7	58000
OptDigits	64	10	5620
PenDigits	16	10	10992
SatImage	36	7	6435
Segmentation	19	7	2310
Waveform	21	3	5000

Table 4.1: Summary of selected UCI data sets

L_p-CSB [58] and MultiBoost [102]. See their respective pseudo-codes in Algorithm 11, 12 and 13.

For each data set we proceed in the following way. Firstly, if needed, we unify train and test data into a single set. Then we carry out a 5-fold cross validation process taking care of maintaining the original proportion of labels for each fold. When training, we compute a cost matrix following the criteria described in 4.3.1. Then we run each algorithm 100 iterations. We resort to classification trees as base learners. As discussed in 4.1.1, AdaC2.M1 and BAdaCost allow the use of multi-class weak learner. Besides, MultiBoost also uses multi-class weak learners but it requires a pool of them to work properly (minimizing (4.9) is intractable). For this reason we create a pool of 6000 weak learners. To asses different weighting schemes in this group of hypothesis, we sample data from 30%, 45%, and 60% of the training data (2000 weak learners for each ratio). In third place, L_p-CSB translates the multi-class problem into a binary one.

The average misclassification cost, $\frac{1}{N}\sum_{n=1}^{N} C(l_n, H(\mathbf{x}_n))$, is collected at the end of each process. Note that, after rescaling the cost matrix, final costs may sum up to a very small quantity. We show the results in table 4.3.2.

It is clear that PIBoost outperforms the rest of algorithms in most of the data bases. To asses the statistical significance of the performance differences among the four methods we use the Friedman test of average ranks. The statistic supports clearly the alternative hypothesis, i.e. algorithms do not achieve equivalent results. Then a post-hoc analysis complements our arguments. We carry out the Bonferroni-Dunn test for both significance levels $\alpha = 0.05$ and $\alpha = 0.10$. The confidence distances[4] for these tests are $CD_{0.05} = 1.2617$ and $CD_{0.10} = 1.1216$, respectively. Figure 4.1 shows the final result. We can conclude that BAdaCost is significantly better than the AdaC2.M1 and L_p-CSB algorithms for the above levels of significance. In the case of MultiBoost we can state the same conclusion for $\alpha = 0.10$, but not for $\alpha = 0.05$ (the difference between ranks is 1.25).

[4]This values depends on the number of classifiers being compared jointly with the number of data sets over which comparison is carried out. See [15].

Data set	AdaC2.M1	MultiBoost	Lp-CSB	BAdaCost
CarEval	0.0026 ($\pm$9)	0.0232 ($\pm$36)	0.0038 ($\pm$15)	**0.0024** ($\pm$15)
Chess	0.0029 ($\pm$5)	0.0262 ($\pm$34)	**0.0004** ($\pm$3)	0.0160 ($\pm$9)
Isolet	0.0289 ($\pm$48)	0.0140 ($\pm$15)	0.0149 ($\pm$18)	**0.0066** ($\pm$14)
SatImage	0.0478 ($\pm$62)	0.0187 ($\pm$23)	0.0170 ($\pm$26)	**0.0132** ($\pm$11)
Letter	0.0491 ($\pm$78)	0.0319 ($\pm$53)	0.0161 ($\pm$23)	**0.0066** ($\pm$7)
Shuttle	**2.1**e^{-5} ($\pm$0.07)	8.9e^{-5} ($\pm$0.08)	3.5e^{-5} ($\pm$0.13)	3.9e^{-5} ($\pm$0.3)
ContraMeth.	0.0980 ($\pm$129)	0.1058 ($\pm$214)	0.0938 ($\pm$359)	**0.0928** ($\pm$253)
CNAE9	0.0397 ($\pm$103)	**0.0171** ($\pm$57)	0.0241 ($\pm$108)	0.0191 ($\pm$51)
OptDigits	0.0366 ($\pm$120)	0.0134 ($\pm$27)	0.0170 ($\pm$34)	**0.0030** ($\pm$8)
PenDigits	0.0326 ($\pm$29)	0.0193 ($\pm$34)	0.0162 ($\pm$72)	**0.0018** ($\pm$5)
Segmenta.	0.0242 ($\pm$123)	0.0154 ($\pm$48)	0.0094 ($\pm$18)	**0.0050** ($\pm$30)
Waveform	0.0905 ($\pm$113)	0.0515 ($\pm$128)	0.0632 ($\pm$201)	**0.0367** ($\pm$96)

Table 4.2: Classification cost rates of Ada.C2M1, MultiBoost, L_p-CSB, and BAdaCost algorithms for each data set after 100 iterations. Standard deviations appear inside parentheses in 10^{-4} scale. Bold values represent the best result achieved for each data base.

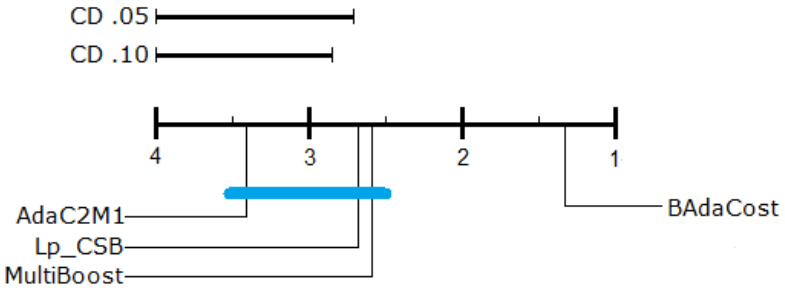
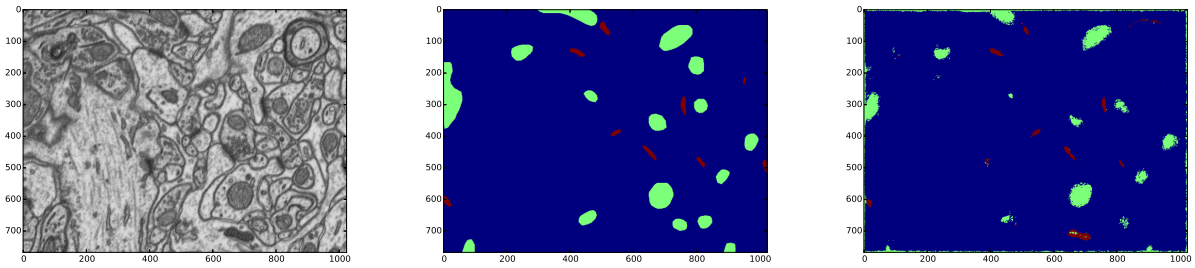


Figure 4.1: Comparison of ranks through the Bonferroni-Dunn test. BAdaCost's average rank is taken as reference. Algorithms significantly worse than our method for a significance level of 0.10 are unified with a blue line.

4.3.3 Unbalanced Data: Synapse and Mitochondria segmentation

To complete the section, we show the performance of our algorithm in the domain of unbalanced classification problems. These problems are characterized for having large differences in the number of samples in each class. Such a situation frequently occurs in complex data sets, like those involving class overlapping, small sample size, or within-class unbalance [38]. When working with unbalanced data, standard classifiers usually perform poorly since they minimize the number of misclassified training samples disregarding minority classes. It is worth mentioning how this has become an important research area in Pattern Recognition [33, 38, 88] insomuch as unbalanced classification problems frequently occur in relevant practical problems, such as Bio-medical Image Analysis [6], object detection in Computer Vision [100] or medical decision-making [65].

Solutions to the class unbalance problem may be coarsely organized into data-based, that



a) **b)** **c)**

Figure 4.2: Example of a segmented image. In **b)**, green pixels belong to mitochondria while red ones belong to synapses. Figure **c)** indicates an estimation.

re-sample the data space to balance the classes, and algorithm-based approaches, that introduce new algorithms that bias the learning towards the minority class [38]. The symbiotic relation between data- and ensemble-based algorithms in the context of two-class unbalanced classification is highlighted in a recent survey [33]. Boosting algorithms have also been extensively used to address this kind of problems [33, 38, 86, 91].

However, with the exception of AdaC2.M1 [86], no previous work has addressed the problem of multi-class Boosting in presence of unbalanced data by using a cost matrix[5]. We compare BAdaCost with the multi-class cost-sensitive algorithms considered in section 4.3.2: AdaC2.M1, L_p-CSB and MultiBoost. We also add the SAMME algorithm to our experiment, considering it a good example of cost-free multi-class algorithm with multi-class weak learners. Let us briefly describe our problem.

In the last years we have seen advances in the automated acquisition of large series of images of brain tissue. The complexity of these images and the high number of neurons in a small section of the brain, makes the automated analysis of these images the only practical solution. Mitochondria and synapses are two cell structures of neurological interest that are suitable for automated processing. However, the classification process to segment these structures is highly unbalanced, since most pixels in these images belong to the background, few of them belong to mitochondria and a small minority belong to synapses. Figure 4.2 shows an example.

For this experiment we collect a training set composed of 10000 background, 4000 mitochondria and 1000 synapse data and a testing set with 20000 data per class. We apply to each image in the stack a set of linear Gaussian filters at different scales to compute zero, first and second order derivatives. For each pixel we get a vector of responses $S = (s_{00}, s_{10}, s_{01}, s_{02}, s_{11}, s_{20})$ that are respectively obtained applying the filters $G_{\sigma *}, \sigma \cdot G_\sigma * \frac{\partial}{\partial x}, \sigma \cdot G_\sigma * \frac{\partial}{\partial y}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial x^2}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial xy}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial y^2}$ where G_σ is a zero mean Gaussian with σ standard deviation. For a value of σ the pixel feature vector is given by $f(\sigma) = (s_{00}, \sqrt{s_{10}^2 + s_{01}^2}, \lambda_1, \lambda_2)$ where λ_1 and λ_2 are the eigenvalues of the Hessian matrix of the pixel, that depend on s_{20}, s_{02} and s_{11}. The final 16-dimensional feature vector for each pixel is given by the concatenation of the $f(\sigma)$ vector at 4 scales (values of σ).

Having described the features, our goal in this subsection is to use the BAdaCost algorithm to label pixels in these images as *mitochondria*, *synapse* and *background*. We compare the five

[5] In [103] the authors propose AdaBoost.NC, a multi-class Boosting algorithm specialized in unbalanced data. They do not resort to cost matrices, rather they use a different insight based on instance- and iteration-dependent penalties that are applied in the reweighting scheme.

algorithms using a cost matrix obtained as described in 4.3.1. It is done obtaining the confusion matrix after applying 15 iterations of the BAdaCost algorithm (with $0|1$-cost matrix) to the training set. Once more, we use classification trees as base learners for each algorithm. In all the algorithms we use a re-sampling factor of $r = 0.7$. In case of SAMME, AdaC2.M1 and BAdaCost we use a shrinkage factor of $s = 0.1$ (this factor is not needed for the rest). For the L_p-CSB algorithm we select $p = 4$ and for MultiBoost we create a pool of 10000 learners following the mentioned re-sampling factor. We run the five algorithms for 150 iterations.

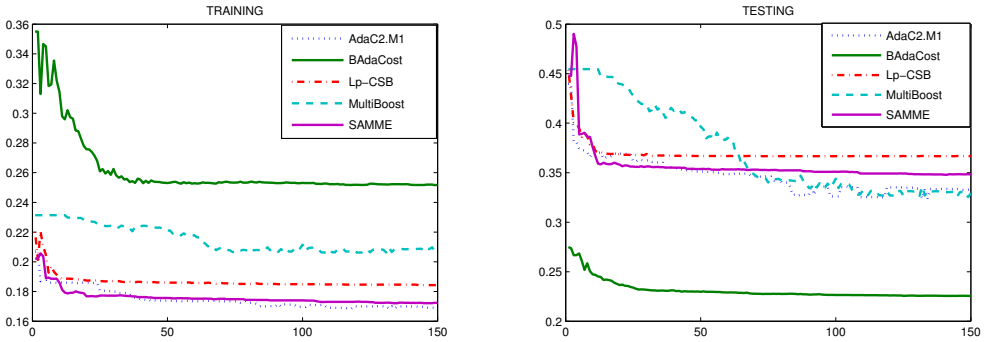


Figure 4.3: Brain images experiment with a heavily unbalanced data set. Training and testing error rates, along the iterations, for each algorithm are shown.

In Fig. 4.3 we show the training and testing classification results for this experiment. Table 4.3 shows the error rates for each algorithm after the last iteration. The unbalance in the training data reflects the essence of this segmentation problem in which synapses cover a very small area in the image, mitochondria a slightly larger and, finally, the background class the largest area. A classifier unaware of this unbalance would overfit the largest class to achieve the lowest error rate on the unbalanced training data set. We can see this effect in plots in Fig. 4.3. The SAMME algorithm achieves the lowest error rate on the training data set, but the poorest in the balanced testing set, clearly showing that it has overfitted the background class. Note here that, although the classes are unbalanced, the error rate is a meaningful classification measure in the testing data set because it is balanced. On the other hand, BAdaCost, since it is a cost-sensitive classifier, gets a much better testing error rate. The training error rate in this case is much higher than SAMME. This is an expected result, since the cost matrix has effectively moved the class boundary towards the majority class. The MultiBoost and AdaC2.M1 algorithms perform better than SAMME in the test set, but clearly worse than BAdaCost. Although L_p-CSB, MultiBoost and AdaC2.M1 are cost-sensitive algorithms, their poor performance in this experiment proves that their results are far from optimal.

	SAMME	**AdaC2.M1**	**L_p-CSB**	**MultiBoost**	**BAdaCost**
Train	0.1723	0.1691	0.1843	0.2095	0.2515
Test	0.3327	0.3327	0.3667	0.3293	0.2255

Table 4.3: Error rates of the five algorithms after the last iteration.

Discussion

In this section we carry out two types of experiments.

The first is conceived to minimize decision costs on real problems when a cost structure is given. To this aim, we have consider a set of UCI benchmark problems for which we compute cost matrices following the process described in 4.3.1. Doing so, the cost-sensitive classification focusses on the difficult decision boundaries. We carry out a comparison with other multi-class cost-sensitive Boosting algorithms. At the light of the results, we conclude that BAdaCost significantly finds the most appropriate assignation of labels for this purpose. This reaffirms the use of the cost-sensitive multi-class exponential loss function for developing algorithms in the area.

In our second experiment we solve a complex problem in presence of unbalanced data. Specifically, we address the segmentation of mitochondria and synapses in images of small sections of the brain. In this case, we endow the multi-class problem with a cost matrix to move the decision boundaries towards the majority class. In this way the classifier does not neglect the minority classes. We compared BAdaCost with other multi-class cost-sensitive algorithms. We used the process explained in section 4.3.1 to estimate a cost matrix for this problem. The results confirm that BAdaCost outperforms significantly the rest of algorithms since the testing error decreases the most. This is essentially due to our unbalance-aware learning. In other words, BAdaCost really takes advantage of a cost matrix to correct the unbalance in the data.

Chapter 5

Conclusions

In the present dissertation we have exploited two definitions of margin to develop new algorithms for multi-class Boosting.

On the one hand, we have proposed a new multi-class Boosting algorithm called PIBoost. The main contribution in it is the use of binary classifiers whose response is encoded in a multi-class vector and evaluated under an exponential loss function. Data labels and classifier responses are encoded in different vector domains in such a way that they produce a set of asymmetric margin values that depend on the distribution of classes separated by the weak learner. In this way PIBoost properly addresses possible class unbalances appearing in the problem binarization. The range of rewards and penalties provided by this multi-class loss function is also related to the amount of information yielded by each weak-learner.

The most informative weak learners are those that classify samples in the smallest class set and, consequently, their sample weight rewards and penalties are the largest. Here the codification produces a fair distribution of the vote or evidence among the classes in the group. We match it with a pattern of common sense, namely: the fact of learning to guess something by discarding possibilities in the proper way. The resulting algorithm maintains the essence of AdaBoost, that, in fact, is a special case of PIBoost when the number of classes is two. Furthermore, the way it translates partial information about the problem into multi-class knowledge let us think of our method as a canonical extension of AdaBoost using binary information.

The experiments performed confirm that PIBoost significantly improves the performance of other well known multi-class classification algorithms. We do not claim that PIBoost is the best multi-class Boosting algorithm in the literature. Rather, we emphasize that the multi-class margin expansion introduced improves existing binary multi-class classification approaches and open new research venues for margin-based classification.

Following the insight behind PIBoost we extended the original AdaBoost for multi-class cost-sensitive problems using multi-class weak learners. By extending the notion of multi-class margin to the cost-sensitive case we introduced a new method and also developed theoretical connections proving that it canonically generalizes SAMME [112], Cost-sensitive AdaBoost [63] and PIBoost [24]. The resulting algorithm, BAdaCost, stands for Boosting Adapted for Cost-matrix. We have shown experimentally that BAdaCost outperforms other relevant Boosting algorithms in the area. We perform this comparison in two types of tasks: minimizing costs on standard data sets, and also improving the test accuracy on a unbalanced data problem.

When making an algorithm cost-sensitive, the costs are a new set of parameters. Estimating these parameters is a difficult problem per se. Most of the cost-sensitive algorithms published

either do not provide a procedure to compute them [58, 102], they resort to a computationally demanding search procedure [86], or, in the unbalance case, they set the costs inversely proportional to the class unbalance ratio [33]. In our experiments, we used a simple procedure to estimate the costs from the confusion matrix obtained with a simple cost-free multi-class rule. This procedure has shown to work well in practice. However, we are aware that this matrix is far from optimal.

Finally we must say that the horizon of applicabilities for both algorithms is vast. Frameworks based on variations of the multi-class margin yield flexible derivatives to new paradigms or new types of problems. The following section is devoted to comment these new insights of future work.

5.1 Future work

Here we discuss some interesting topics for future venues of research. In a first paragraph we review theoretical aspects that are considered as immediate goals. Then we comment different areas of applicability in the field of supervised classification, especially for Computer Vision.

5.1.1 New theoretical scopes

Many Boosting algorithms asses their convergence through error bounds [28, 26, 80]. We derived our algorithms based on a statistical view of Boosting. In a future research we will derive bounds on the training and test errors of PIBoost and BAdaCost. In the case of PIBoost, its weak learners are grouped in separators but their responses, encoded as margin vectors, are sum up jointly. Therefore, the demonstration of the error bounding should take into account each partial classification and the result when they are merged into a final decision. Such an analysis is not immediate and would require a thorough research.

Likewise, as we stated in section 2.5 there is a plethora of works discussing different perspectives of Boosting. It would be quite interesting to link some of them to multi-class margin-based theory. Three examples are the reduction of entropy as analyzed in [46, 84], the game-theory point of view described in [67] and the use of different margin-based loss functions [61, 10].

With regard to PIBoost's separators, in our experiments we did not take into account sets of labels larger than a pair. If the number of labels allows it, it is also possible to add new "clues" to the additive model fitted coming from trios or quartets of labels. The main disadvantage, obviously, is the computational load associated. As we mentioned in section 3.4 the number of separators could become prohibitive. For this reason we are interested in developing a well grounded scheme for selecting the best subsets of labels for our separators.

Establishing the proper cost matrix is a difficult key step in many cost-sensitive problems. Aware of this fact, we have introduced a procedure for measuring costs on hard-to-fit decision boundaries and also for tackling asymmetries in data. In future research we would like to devise a more efficient and optimal algorithm to estimate cost matrices in certain problems, such as for example unbalanced ones. Similarly, we also want to derive symmetric matrices suitable for problems where some decision boundaries between pairs of labels are more important than others, regardless the importance of the labels in a pair.

5.1.2 Other scopes of supervised learning

Firstly, we must point out that the supervised classification addressed in the dissertation is concerned with a discrete set of values (in fact, a finite set of labels). Many real problems require predictions on continuous magnitudes, which invites us to extend both PIBoost and BAdaCost to regression problems.

Labeling data is a time consuming and costly process, hence the interest in developing semi-supervised algorithms. We would like to study the problem of semi-supervised multiclass boosting with costs.

Some problems present such a high dimensionality that a feature selection process is firstly needed. AdaBoost has been successfully used for selecting discriminative features while building a strong cascade classifier, e.g. [101, 100]. Sharing this perspective we are also motivated to apply PIBoost to retrieve the best features for multi-class problems.

In Computer Vision, detection problems usually suffer from unbalanced data since positive classes have much less samples than the background one. The usual technique to deal with such unbalance is sub-sampling the background class, what entails an information loss. With BAdaCost the majority class sub-sampling is not needed and the full training set can be used. Moreover, it is well known that training a single classifier for each object (a pure binary problem) is worse than building a multi-class object detector [95] because visual features can be shared among the weak learners. We are studying the application of PIBoost to multi-pose detection of objects as practical and intuitive application.

Last but not least, we would like to develop a cost-sensitive derivation of PIBoost. In other words, we plan to derive a canonical generalization of AdaBoost to tackle classifications in presence of a multi-class cost matrix by using binary weak learners.

Appendix A

Proofs

A.1 Proof of expression (3.3)

Assumee, without loss of generality, that $\mathbf{x}$ belongs to the first class and we try to separate the set S, with the first s-labels, from the rest using $\mathbf{f}^S$. Assume also that there is success when classifying with it. In that case the value of the margin will be

$$\begin{aligned}\mathbf{y}^\top \mathbf{f}^S(\mathbf{x}) &= \left(1, \frac{-1}{K-1}, \ldots, \frac{-1}{K-1}\right)\left(\frac{1}{s}, \ldots, \frac{1}{s}, \frac{-1}{K-s}, \ldots, \frac{-1}{K-s}\right)^\top \\ &= \frac{1}{s} - \frac{s-1}{s(K-1)} + \frac{(K-s)}{(K-1)(K-s)} = \frac{K}{s(K-1)}.\end{aligned}$$

If the separator fails then $\mathbf{f}^S(x)$ would have opposite sign and therefore the result.

Besides, assume now that the real label of the instance is the same but now we separate the last s-labels from the rest. Assume also that this time $\mathbf{f}^S$ erroneously classifies the instance as belonging to those last labels. The value of the margin will be

$$\begin{aligned}\mathbf{y}^\top \mathbf{f}^S(\mathbf{x}) &= \left(1, \frac{-1}{K-1}, \ldots, \frac{-1}{K-1}\right)\left(\frac{-1}{K-s}, \ldots, \frac{-1}{K-s}, \frac{1}{s}, \ldots, \frac{1}{s}\right)^\top \\ &= \frac{-1}{K-s} + \frac{(K-s-1)}{(K-1)(K-s)} - \frac{s}{(K-1)s} = \frac{-K}{(K-1)(K-s)}.\end{aligned}$$

Again, the sign of the result would be opposite if $\mathbf{f}^S$ excludes $\mathbf{x}$ from the first labels group.

A.2 Proof of Lemma 1

Let us fix a subset of s labels, $s = |S|$, and assume that we have fitted a separator $\mathbf{f}_m(\mathbf{x})$ (whose S-index we omit) as an additive model $\mathbf{f}_{m+1}(\mathbf{x}) = \mathbf{f}_m(\mathbf{x}) + \beta\mathbf{g}(\mathbf{x})$, in the m-th step. We fix a $\beta > 0$ and rewriting the expression to look for the best $\mathbf{g}(\mathbf{x})$:

$$\sum_{n=1}^{N} \exp\left(\frac{-1}{K}\mathbf{y}_n^\top\left(\mathbf{f}_m(\mathbf{x}_n) + \beta\mathbf{g}(\mathbf{x}_n)\right)\right) = \sum_{n=1}^{N} w(n)\exp\left(\frac{-1}{K}\beta\mathbf{y}_n^\top\mathbf{g}(\mathbf{x}_n)\right) = \tag{A.1}$$

$$= \sum_{l_n \in S} w(n) \exp\left(\frac{\mp\beta}{s(K-1)}\right) + \sum_{l_n \notin S} w(n) \exp\left(\frac{\mp\beta}{(K-s)(K-1)}\right) = \tag{A.2}$$

$$= \left(\sum_{l_n \in S} w(n)\right) \exp\left(\frac{-\beta}{s(K-1)}\right) + \left(\sum_{l_n \notin S} w(n)\right) \exp\left(\frac{-\beta}{(K-s)(K-1)}\right) +$$

$$+ \left[\exp\left(\frac{\beta}{s(K-1)}\right) - \exp\left(\frac{-\beta}{s(K-1)}\right)\right] \sum_{l_n \in S} w(n) I(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0) + \tag{A.3}$$

$$+ \left[\exp\left(\frac{\beta}{(K-s)(K-1)}\right) - \exp\left(\frac{-\beta}{(K-s)(K-1)}\right)\right] \sum_{l_n \notin S} w(n) I(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0).$$

The last expression is a sum of four terms. As can be seen, the first and second are constants while the third and fourth are the ones depending on $\mathbf{g}(\mathbf{x})$. The values in brackets are positive constants. Let us use denote them B_1 and B_2, respectively. So minimizing the above expression is equivalent to minimizing

$$B_1 \sum_{l_n \in S} w(n) I(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0) + B_2 \sum_{l_n \notin S} w(n) I(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0)\ . \tag{A.4}$$

Hence the first point of the Lemma follows.

Now assume known $\mathbf{g}(\mathbf{x})$ and its error E on training data. The error can be decomposed into two parts:

$$E = \sum_{l_n \in S} w(n) I(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0)\ + \sum_{l_n \notin S} w(n) I(\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n) < 0) = E1 + E2\ . \tag{A.5}$$

The expression (A.3) can be written now as

$$A_1 \exp\left(\frac{-\beta}{s(K-1)}\right) + \left[\exp\left(\frac{\beta}{s(K-1)}\right) - \exp\left(\frac{-\beta}{s(K-1)}\right)\right] E1 +$$
$$+ A_2 \exp\left(\frac{-\beta}{(K-s)(K-1)}\right) + \left[\exp\left(\frac{\beta}{(K-s)(K-1)}\right) - \exp\left(\frac{-\beta}{(K-s)(K-1)}\right)\right] E2\ , \tag{A.6}$$

where $A_1 = \sum_{l_n \in S} w(n)$ and $A_2 = \sum_{l_n \notin S} w(n)$. It can be easily verified that the above expression is convex with respect to β. So differentiating w.r.t. β, equating to zero and simplifying terms we get:

$$\begin{aligned} &\frac{E1}{s} \exp\left(\frac{\beta}{s(K-1)}\right) + \frac{E2}{K-s} \exp\left(\frac{\beta}{(K-s)(K-1)}\right) = \\ &= \frac{(A_1 - E1)}{s} \exp\left(\frac{-\beta}{s(K-1)}\right) + \frac{(A_2 - E2)}{K-s} \exp\left(\frac{-\beta}{(K-s)(K-1)}\right). \end{aligned} \tag{A.7}$$

There is no direct procedure to solve β here. We propose the change of variable $\beta = s(K-s)(K-1)\log(x)$ with $x > 0$. This change transform the last equation into the polynomial

$$(K-s)E1x^{(K-s)} + sE2x^s - s(A_2 - E2)x^{-s} - (K-s)(A_1 - E1)x^{-(K-s)} = 0, \tag{A.8}$$

or, equivalently, multiplying by $x^{(K-s)}$

$$(K-s)E1x^{2(K-s)} + sE2x^K - s(A_2 - E2)x^{(K-2s)} - (K-s)(A_1 - E1) = 0. \tag{A.9}$$

According to Descartes' *Theorem of the Signs* the last polynomial has a single real positive root, which proves the second point of the Lemma.

A.3 Proof of Lemma 2

Assume that in the *m*-th iteration we have fitted a classifier $\mathbf{f}_m(\mathbf{x})$ as an additive model and we are searching for the parameters $(\beta, \mathbf{g})$ to add in the next step, $\mathbf{f}_{m+1}(\mathbf{x}) = \mathbf{f}_m(\mathbf{x}) + \beta\mathbf{g}(\mathbf{x})$.

$$\sum_{n=1}^{N} \exp\left(\mathbf{C}^*(l_n, -)\left(\mathbf{f}_m(\mathbf{x}_n) + \beta\mathbf{g}(\mathbf{x}_n)\right)\right) \tag{A.10}$$

$$= \sum_{n=1}^{N} w(n) \exp\left(\beta\mathbf{C}^*(l_n, -)\mathbf{g}(\mathbf{x}_n)\right) \tag{A.11}$$

$$= \sum_{j=1}^{K} \sum_{\{n:l_n=j\}} w(n) \exp\left(\beta\mathbf{C}^*(j, -)\mathbf{g}(\mathbf{x}_n)\right) \tag{A.12}$$

$$= \sum_{j=1}^{K} \sum_{\{n:l_n=j\}} w(n) \exp\left(\frac{\beta K}{K-1} C^*(j, G(\mathbf{x}_n))\right) . \tag{A.13}$$

In the last step we take into account the equivalence between vectorial-valued functions, $\mathbf{g}$, and label-valued functions, G. If we denote $S(j) = \{n : l_n = G(\mathbf{x}_n) = j\}$ the set of index of well classified instances with $l_n = j$ and $F(j,k) = \{n : l_n = j, G(\mathbf{x}_n) = k\}$ denotes the index where G classifies as k when the real label is j then the above expression can be rewritten

$$\sum_{j=1}^{K} \left(\sum_{n\in S(j)} w(n) \exp\left(\frac{\beta K C^*(j,j)}{K-1}\right) + \sum_{n\in F(j,k)} w(n) \exp\left(\frac{\beta K C^*(j,k)}{K-1}\right) \right) \tag{A.14}$$

$$= \sum_{j=1}^{K} \left(S_j \exp\left(\frac{\beta K C^*(j,j)}{K-1}\right) + \sum_{k\neq j} E_{j,k} \exp\left(\frac{\beta K C^*(j,k)}{K-1}\right) \right) . \tag{A.15}$$

Where $S_j = \sum_{n\in S(j)} w(n)$ and $E_{j,k} = \sum_{n\in F(j,k)} w(n)$. Taking into account that these constants are positive values and also $\exp(\beta C^*(i,j)) > 0$ ($\forall i, j \in L$), we can omit the term $K/(K-1)$ appearing in the exponents in order to address the minimization. Subsequently, the objective function can be written:

$$\sum_{j=1}^{K} \left(S_j \exp\left(\beta C^*(j,j)\right) + \sum_{k\neq j} E_{j,k} \exp\left(\beta C^*(j,k)\right) \right) . \tag{A.16}$$

Now, fixed a value $\beta > 0$, the optimal step, $\mathbf{g}$, can be found minimizing

$$\sum_{j=1}^{K} \left(S_j \underbrace{\exp\left(\beta C^*(j,j)\right)}_{A(j,j)^\beta} + \sum_{k\neq j} E_{j,k} \underbrace{\exp\left(\beta C^*(j,k)\right)}_{A(j,k)^\beta} \right) \tag{A.17}$$

$$= \sum_{j=1}^{K} \left(\left(\sum_{n\in S(j)} w(n) \right) A(j,j)^\beta + \sum_{k\neq j} \left(\sum_{n\in F(j,k)} w(n) \right) A(j,k)^\beta \right) \tag{A.18}$$

$$= \sum_{n=1}^{N} w(n) \left(A(l_n, l_n)^\beta I\left(G(\mathbf{x}_n) = l_n\right) + \sum_{k\neq l_n} A(l_n, k)^\beta I\left(G(\mathbf{x}_n) = k\right) \right) . \tag{A.19}$$

Finally if we assume known a direction, $\mathbf{g}$, then its weighted errors, $E_{j,k}$, and successes, S_j, will be computable. So differentiating (A.16) with respect to β (note that is a convex function) and equating to zero we get

$$\sum_{j=1}^{K}\left(\sum_{k\neq j} E_{j,k}C^*(j,k)\exp\left(\beta C^*(j,k)\right) + S_j C^*(j,j)\exp\left(\beta C^*(j,j)\right)\right) = 0 \tag{A.20}$$

$$\sum_{j=1}^{K}\sum_{k\neq j} E_{j,k}C^*(j,k)\exp\left(\beta C^*(j,k)\right) = -\sum_{j=1}^{K} S_j C^*(j,j)\exp\left(\beta C^*(j,j)\right) \tag{A.21}$$

$$\sum_{j=1}^{K}\sum_{k\neq j} E_{j,k}C^*(j,k)A(j,k)^{\beta} = -\sum_{j=1}^{K} S_j C^*(j,j)A(j,j)^{\beta} \tag{A.22}$$

$$\sum_{j=1}^{K}\sum_{k\neq j} E_{j,k}C(j,k)A(j,k)^{\beta} = \sum_{j=1}^{K}\sum_{h=1}^{K} S_j C(j,h)A(j,j)^{\beta}\ . \tag{A.23}$$

There is no direct procedure to solve this equation so we resort to iterative methods.

A.4 Proof of Corollary 1

As was commented in section 4.1.2 it is easy to see that when $\mathbf{C}$ is defined in the following way:

$$C(i,j) := \begin{cases} 0 & \text{for } i = j \\ \frac{1}{K(K-1)} & \text{for } i \neq j \end{cases} \quad \forall i,j \in L \tag{A.24}$$

, then a discrete vectorial weak learner, $\mathbf{f}$, yields $\mathbf{C}^*(l,-)\mathbf{f}(\mathbf{x}) = -1/(K-1)$ for right classifications and $\mathbf{C}^*(l,-)\mathbf{f}(\mathbf{x}) = 1/(K-1)^2$ for mistakes. Both quantities are in fact the values of $\frac{-1}{K}\mathbf{y}^\top\mathbf{f}(\mathbf{x})$ appearing in the exponent of the loss function in SAMME. Thus expression (A.16) can be written

$$\sum_{j=1}^{K}\left(S_j\exp\left(\frac{-\beta}{K-1}\right) + \sum_{k\neq j} E_{j,k}\exp\left(\frac{\beta}{(K-1)^2}\right)\right) \tag{A.25}$$

$$= \underbrace{\left(\sum_{j=1}^{K} S_j\right)}_{S}\exp\left(\frac{-\beta}{K-1}\right) + \underbrace{\left(\sum_{j=1}^{K}\sum_{k\neq j} E_{j,k}\right)}_{E}\exp\left(\frac{\beta}{(K-1)^2}\right) \tag{A.26}$$

$$= S\exp\left(\frac{-\beta}{K-1}\right) + E\exp\left(\frac{\beta}{(K-1)^2}\right) \tag{A.27}$$

$$= (1-E)\exp\left(\frac{-\beta}{K-1}\right) + E\exp\left(\frac{\beta}{(K-1)^2}\right) \tag{A.28}$$

$$= \exp\left(\frac{-\beta}{K-1}\right) + E\left(\exp\left(\frac{\beta}{(K-1)^2}\right) - \exp\left(\frac{-\beta}{K-1}\right)\right). \tag{A.29}$$

So, despite the value of β, the above expression is minimized when $E = \sum_{n=1}^{N} w(n)I\left(G(\mathbf{x}_n) \neq l_n\right)$ is minimum.

For the second point of the corollary we just need to consider the above expression as a function of β. Differentiating and equating to zero we get

$$-\exp\left(\frac{-\beta}{K-1}\right)+E\left(\frac{1}{K-1}\exp\left(\frac{\beta}{(K-1)^2}\right)+\exp\left(\frac{-\beta}{K-1}\right)\right)=0 \tag{A.30}$$

$$\frac{E}{K-1}\exp\left(\frac{\beta}{(K-1)^2}\right)=(1-E)\exp\left(\frac{-\beta}{K-1}\right) \tag{A.31}$$

$$\exp\left(\frac{K\beta}{(K-1)^2}\right)=\frac{(K-1)(1-E)}{E} \tag{A.32}$$

and taking logarithms

$$\frac{K\beta}{(K-1)^2}=\log\left(\frac{1-E}{E}\right)+\log\left(K-1\right) \tag{A.33}$$

$$\beta=\frac{(K-1)^2}{K}\left(\log\left(\frac{1-E}{E}\right)+\log\left(K-1\right)\right). \tag{A.34}$$

Hence the second point of the corollary follows.

A.5 Proof of Corollary 2

Given the (2×2)-cost-matrix, let $C_1=C(1,2)$ and $C_2=C(2,1)$ denote the non diagonal values. The expression (A.16) becomes:

$$\sum_{j=1}^{2}\left(S_j\exp\left(-\beta C_j\right)+\underbrace{E_{j,k}}_{k\neq j}\exp\left(\beta C_j\right)\right)= \tag{A.35}$$

$$S_1\exp\left(-\beta C_1\right)+E_{1,2}\exp\left(\beta C_1\right)+S_2\exp\left(-\beta C_2\right)+E_{2,1}\exp\left(\beta C_2\right). \tag{A.36}$$

Let us assume now that $\beta>0$ is known. Using the Lemma the optimal discrete weak learner minimizing the expected loss is

$$\arg\min_{\mathbf{g}}\left[\mathrm{e}^{\beta C_1}E_{1,2}+\mathrm{e}^{-\beta C_1}S_1+\mathrm{e}^{\beta C_2}E_{2,1}+\mathrm{e}^{-\beta C_2}S_2\right] \tag{A.37}$$

, changing the notation $T_1=\sum_{\{n:l_n=1\}}w(n)$, $T_2=\sum_{\{n:l_n=2\}}w(n)$, $E_{1,2}=b=T_1-S_1$ and $E_{2,1}=d=T_2-S_2$,

$$\arg\min_{\mathbf{g}}\left[\mathrm{e}^{\beta C_1}b+\mathrm{e}^{-\beta C_1}(T_1-b)+\mathrm{e}^{\beta C_2}d+\mathrm{e}^{-\beta C_2}(T_2-d)\right]= \tag{A.38}$$

$$\arg\min_{\mathbf{g}}\left[\left(\mathrm{e}^{\beta C_1}-\mathrm{e}^{-\beta C_1}\right)b+\mathrm{e}^{-\beta C_1}T_1+\left(\mathrm{e}^{\beta C_2}-\mathrm{e}^{-\beta C_2}\right)d+\mathrm{e}^{-\beta C_2}T_2\right]. \tag{A.39}$$

Besides, if we assume known the optimal weak learner, $\mathbf{g}$, then its weighted success/error rates will be computable. We can find the best value β using the Lemma. In this binary case the following expression must be solved

$$E_{1,2}C_1\mathrm{e}^{\beta C_1}+E_{2,1}C_2\mathrm{e}^{\beta C_2}=S_1C_1\mathrm{e}^{-\beta C_1}+S_2C_2\mathrm{e}^{-\beta C_2}. \tag{A.40}$$

Again, using the notation T_1, T_2, b and d we get

$$bC_1\mathrm{e}^{\beta C_1}+dC_2\mathrm{e}^{\beta C_2}=(T_1-b)C_1\mathrm{e}^{-\beta C_1}+(T_2-d)C_2\mathrm{e}^{-\beta C_2} \tag{A.41}$$

$$bC_1\left(\mathrm{e}^{\beta C_1}+\mathrm{e}^{-\beta C_1}\right)+dC_2\left(\mathrm{e}^{\beta C_2}+\mathrm{e}^{\beta C_2}\right)=T_1C_1\mathrm{e}^{-\beta C_1}+T_2C_2\mathrm{e}^{-\beta C_2} \tag{A.42}$$

$$2C_1b\cosh\left(\beta C_1\right)+2C_2d\cosh\left(\beta C_2\right)=T_1C_1\mathrm{e}^{-\beta C_1}+T_2C_2\mathrm{e}^{-\beta C_2}\,, \tag{A.43}$$

that proves the equivalence between both algorithms for binary problems.

A.6 Proof of Corollary 3

Let S denote a subset of s-labels of the problem. We can simplify the notation using 1 or 2 for denoting the presence or absence of labels of S in data. The margin values are $\mathbf{y}^\top \mathbf{f}(\mathbf{x}) = \frac{\pm K}{s(K-1)}$, when $\mathbf{y} \in S$, and $\mathbf{y}^\top \mathbf{f}(\mathbf{x}) = \frac{\pm K}{(K-s)(K-1)}$, when $\mathbf{y} \notin S$. In both cases there is positive/negative sign in case of right/wrong classification. In turn the exponential loss function, $\exp(-\mathbf{y}^\top \mathbf{f}(\mathbf{x})/K)$, yields $\exp(\frac{\mp 1}{s(K-1)})$ and $\exp(\frac{\mp 1}{(K-s)(K-1)})$ respectively.

Let $\mathbf{C}$ be (2×2)-cost-matrix with non diagonal values $C(1,2) = \frac{1}{sK}$ and $C(2,1) = \frac{1}{(K-s)K}$. This matrix produces cost-sensitive multi-class margins with the same values on the loss function. Thus we can apply our result to this binary cost-sensitive sub-problem. In particular we can apply the Corollary 2 directly.

Let $\beta > 0$ be known. Substituting in expresion (A.39) we get the optimal weak learner, $\mathbf{g}$, solving

$$\begin{aligned}\arg\min_{\mathbf{g}} &\left(\exp\left(\frac{\beta}{s(K-1)}\right) - \exp\left(\frac{-\beta}{s(K-1)}\right)\right) E1 + A_1 \exp\left(\frac{-\beta}{s(K-1)}\right) + \\ &+ \left(\exp\left(\frac{\beta}{(K-s)(K-1)}\right) - \exp\left(\frac{-\beta}{(K-s)(K-1)}\right)\right) E2 + A_2 \exp\left(\frac{-\beta}{(K-s)(K-1)}\right) .\end{aligned} \tag{A.44}$$

Where $A_1 = \sum_{\{n:l_n=1\}} w(n)$, $A_2 = \sum_{\{n:l_n=2\}} w(n)$, $E1 = \sum_{\{n:g(x_n)\neq l_n=1\}} w(n)$ and $E2 = \sum_{\{n:g(x_n)\neq l_n=2\}} w(n)$.

If we assume known the optimal direction of classification $\mathbf{g}$, then its weighted errors and successes, we can compute the optimal step β using (A.43) as the solution to

$$\begin{aligned}&\frac{2E1}{s}\cosh\left(\frac{\beta}{s(K-1)}\right) + \frac{2E2}{(K-s)}\cosh\left(\frac{\beta}{(K-s)(K-1)}\right) = \\ &\frac{A_1}{s}\exp\left(\frac{-\beta}{s(K-1)}\right) + \frac{A_2}{(K-s)}\exp\left(\frac{-\beta}{(K-s)(K-1)}\right) .\end{aligned} \tag{A.45}$$

Making the change of variable $\beta = s(K-s)(K-1)\log x$ we get

$$\frac{E1}{s}\left(x^{(K-s)} - x^{-(K-s)}\right) + \frac{E2}{(K-s)}\left(x^s - x^{-s}\right) = \frac{A_1}{s}x^{-(K-s)} + \frac{A_2}{(K-s)}x^{-s} . \tag{A.46}$$

Which is equivalent to find the only real solution (Descartes Theorem of signs) of the following polynomial:

$$P(x) = E1(K-s)x^{2(K-s)} + E2sx^K - s(A_2 - E2)x^{(K-2s)} - (K-s)(A_1 - E1) . \tag{A.47}$$

Hence the Corollary follows.

Bibliography

[1] Naoki Abe, Bianca Zadrozny, and John Langford. An iterative method for multi-class cost-sensitive learning. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 3–11, 2004.

[2] Alekh Agarwal. Selective sampling algorithms for cost-sensitive multiclass prediction. In *Proc. International Conference on Machine Learning (ICML)*, volume 28, pages 1220–1228, 2013.

[3] Erin L. Allwein, Robert E. Schapire, Yoram Singer, and Pack Kaelbling. Reducing multi-class to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[4] Yonatan Amit, Dekel Ofer, and Yoram Singer. A boosting algorithm for label covering in multilabel problems. *Journal of Machine Learning Research*, 2:27–34, 2007.

[5] Shumeet Baluja and Henry A. Rowley. Boosting sex identification performance. *International Journal of Computer Vision*, 71(1):111–119, 2007.

[6] C. Becker, K. Ali, G. Knott, and P. Fua. Learning context cues for synapse segmentation. *IEEE Transactions on Medical Imaging*, 32(10):1864–1877, 2013.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag New York, Inc., 2006.

[8] Leo Breiman. Bagging predictors. *Machine Learning*, pages 123–140, 1996.

[9] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[10] Peter Bühlmann and Bin Yu. Boosting with the l_2 loss: Regression and classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

[11] Wen-Chung Chang and Chih-Wei Cho. Multi-class boosting with color-based haar-like features. In *Signal-Image Technologies and Internet-Based System (SITIS)*, pages 719–726, 2007.

[12] Junli Chen, Xuezhong Zhou, and Zhaohui Wu. A multi-label chinese text categorization system based on boosting algorithm. In *Computer and Information Technology (CIT)*, pages 1153–1158, 2004.

[13] Ke Chen and Shihai Wang. Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions. *Transactions on Pattern Analysis and Machine Intelligence*, 33(1):129–143, 2011.

[14] Corinna Cortes, Mehryar Mohri, and Umar Syed. Deep boosting. In *Proc. International Conference on Machine Learning (ICML)*, 2014.

[15] Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[16] Hongbo Deng, Jianke Zhu, Michael R. Lyu, and Irwin King. Two-stage multi-class adaboost for facial expression recognition. In *International Joint Conference on Neural Networks (IJCNN)*, pages 3005–3010, 2007.

[17] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, pages 263–286, 1995.

[18] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proc. International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 155–164, 1999.

[19] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2 edition, 2000.

[20] Gunther Eibl and Karl-Peter Pfeiffer. Multiclass boosting for weak classifiers. *Journal of Machine Learning Research*, 6:189–210, 2005.

[21] Charles Elkan. The foundations of cost-sensitive learning. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 973–978, 2001.

[22] Andrea Esuli, Tiziano Fagni, and Fabrizio Sebastiani. Treeboost.mh: A boosting algorithm for multi-label hierarchical text categorization. *String Processing and Information Retrieval*, pages 13–24, 2006.

[23] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. Adacost: Misclassification cost-sensitive boosting. In *Proc. International Conference on Machine Learning (ICML)*, pages 97–105, 1999.

[24] Antonio Fernández-Baldera and Luis Baumela. Multi-class boosting with asymmetric weak-learners. *Pattern Recognition*, 47(5):2080–2090, 2014.

[25] Antonio Fernández-Baldera, José M. Buenaposada, and Luis Baumela. Multi-class boosting for imbalanced data. In *Proc. Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, pages 1–8, 2015.

[26] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. International Conference on Machine Learning (ICML)*, pages 148–156, 1996.

[27] Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *Conference on Computational Learning Theory*, pages 325–332, 1996.

[28] Yoav Freund and Robert E. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:199–139, 1997.

[29] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[30] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining*. Springer series in statistics. Springer, 2009.

[31] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[32] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

[33] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. 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: Applications and Reviews*, 42(4):463–484, July 2012.

[34] Tianshi Gao and Daphne Koller. Multiclass boosting with hinge loss based on output coding. In *Proc. International Conference on Machine Learning (ICML)*, 2011.

[35] Venkatesan Guruswami and Amit Sahai. Multiclass learning, boosting, and error-correcting codes. In *Proc. Annual Conference on Computational Learning Theory (COLT)*, pages 145–155, 1999.

[36] Zhihui Hao, Chunhua Shen, Nick Barnes, and Bo Wang. Totally-corrective multi-class boosting. In *Asian Conference on Computer Vision (ACCV)*, volume 6495, pages 269–280, 2010.

[37] Trevor Hastie and Robert Tibshirani. *Generalized Additive Models*. Monographs on Statistics and Applied Probability. Chapman and Hall, 1990.

[38] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[39] Jian Huang, Seyda Ertekin, Yang Song, Hongyuan Zha, and C. Lee Giles. Efficient multiclass boosting classification with active learning. In *SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2007.

[40] R. L. Iman and J. M. Davenport. Approximation of the critical region of the friedman statistic. *Communications in Statistics*, pages 571–595, 1980.

[41] Alan Julian Izenman. *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer Publishing Company, Inc., 1 edition, 2008.

[42] Wei Jiang, Shih-Fu Chang, and Alexander C Loui. Kernel sharing with joint boosting for multi-class concept detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[43] Matt Johnson and Roberto Cipolla. Improved image annotation and labelling through multi-label boosting. In *British Machine Vision Conference (BMVC)*, 2005.

[44] Michael Kearns and Leslie Valiant. Learning boolean formulae or finite automata is hard as margin. Technical report, Harvard University, August 1988.

[45] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

[46] Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. In *Proc. Annual Conference on Computational Learning Theory (COLT)*, pages 134–144, 1999.

[47] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[48] Tae kyun Kim and Roberto Cipolla. Mcboost: Multiple classifier boosting for perceptual co-clustering of images and visual features. In *Advances in Neural Information Processing Systems (NIPS)*, pages 841–848, 2009.

[49] Iago Landesa-Vázquez and José Luis Alba-Castro. Shedding light on the asymmetric learning capability of adaboost. *Pattern Recognition Letters*, 33(3):247–255, 2012.

[50] Iago Landesa-Vázquez and José Luis Alba-Castro. Double-base asymmetric adaboost. *Neurocomputing*, 118:101–114, 2013.

[51] Yoonkyung Lee, Yi Lin, and Grace 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.

[52] Leonidas Lefakis and Francois Fleuret. Joint cascade optimization using a product of boosted classifiers. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1315–1323, 2010.

[53] Christian Leistner, Helmut Grabner, and Horst Bischof. Semi-supervised boosting using visual similarity learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[54] Yen-Yu Lin and Tyng-Luh Liu. Robust face detection with multi-class boosting. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 680–687, 2005.

[55] Li Liu, Ling Shao, and Peter Rockett. Boosted key-frame selection and correlated pyramidal motion-feature representation for human action recognition. *Pattern Recognition*, 46(7):1810–1818, 2013.

[56] Xu-Ying Liu and Zhi-Hua Zhou. Towards cost-sensitive learning for real-world applications. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) Workshops*, volume 7104, pages 494–505, 2012.

[57] Hung-Yi Lo, Ju-Chiang Wang, Hsin-Min Wang, and Shou-De Lin. Cost-sensitive multi-label learning for audio tag annotation and retrieval. *IEEE Transactions on Multimedia*, 13(3):518–529, 2011.

[58] Aurelie C. Lozano and Naoki Abe. Multi-class cost-sensitive boosting with p-norm loss functions. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 506–514, 2008.

[59] Pavan Kumar Mallapragada, Rong Jin, Anil K. Jain, and Yi Liu. Semiboost: Boosting for semi-supervised learning. *Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2000–2014, 2009.

[60] H. Masnadi-Shirazi and N. Vasconcelos. Asymmetric boosting. In *Proc. International Conference on Machine Learning (ICML)*, pages 609–619, 2007.

[61] Hamed Masnadi-Shirazi, Vijay Mahadevan, and Nuno Vasconcelos. On the design of robust classifiers for computer vision. In *Computer Vision and Pattern Recognition (CVPR)*, pages 779–786, 2010.

[62] Hamed Masnadi-Shirazi and Nuno Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1049–1056, 2008.

[63] Hamed Masnadi-Shirazi and Nuno Vasconcelos. Cost-sensitive boosting. *Transactions on Pattern Analysis and Machine Intelligence*, 33:294–309, 2011.

[64] Hamed Masnadi-Shirazi, Nuno Vasconcelos, and Vijay Mahadevan. On the design of robust classifiers for computer vision. In *Computer Vision and Pattern Recognition (CVPR)*, pages 779–786, 2010.

[65] Maciej A Mazurowski, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker, and Georgia D Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21(2):427–436, 2008.

[66] David Mease and Abraham Wyner. Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, 9:131–156, 2008.

[67] I. Mukherjee and R. E. Schapire. A theory of multiclass boosting. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1714–1722, 2010.

[68] P. B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.

[69] Deirdre B. O'Brien, Maya R. Gupta, and Robert M. Gray. Cost-sensitive multi-class classification from probability estimates. In *Proc. International Conference on Machine Learning (ICML)*, pages 712–719, 2008.

[70] A. Opelt, A. Pinz, and A. Zisserman. Learning an alphabet of shape and appearance for multi-class object detection. *International Journal of Computer Vision*, 80(1):16–44, 2008.

[71] Mark D. Reid, Robert C. Williamson, and Peng Sun. The convexity and design of composite multiclass losses. In *Proc. International Conference on Machine Learning (ICML)*, pages 687–694, 2012.

[72] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[73] Saharon Rosset. Robust boosting and its relation to bagging. In *Proc. International Conference on Knowledge Discovery in Data Mining (KDD)*, pages 249–255, 2005.

[74] Mohammad J. Saberian and Nuno Vasconcelos. Multiclass boosting: Theory and algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[75] Amir Saffari, Christian Leistner, and Horst Bischof. Regularized multi-class semi-supervised boosting. In *Computer Vision and Pattern Recognition (CVPR)*, pages 967–974, 2009.

[76] Raúl Santos-Rodríguez, Alicia Guerrero-Curieses, Rocío Alaiz-Rodríguez, and Jesús Cid-Sueiro. Cost-sensitive learning based on bregman divergences. *Machine Learning*, 76(2-3):271–285, 2009.

[77] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[78] Robert E. Schapire. Using output codes to boost multiclass learning problems. In *Proc. International Conference on Machine Learning (ICML)*, pages 313–321, 1997.

[79] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the efectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[80] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

[81] Robert E Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168, 2000.

[82] Chunhua Shen, Junae Kim, Lei Wang, and Anton Hengel. Positive semidefinite metric learning with boosting. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1651–1659, 2009.

[83] Chunhua Shen, Junae Kim, Lei Wang, and Anton Van Den Hengel. Positive semidefinite metric learning using boosting-like algorithms. *Journal of Machine Learning Research*, 13:1007–1036, 2012.

[84] Chunhua Shen and Hanxi Li. On the dual formulation of boosting algorithms. *Transactions on Pattern Analysis and Machine Intelligence*, 32:2216–2231, 2010.

[85] Yifan Shi, Aaron F. Bobick, and Irfan A. Essa. Learning temporal sequence model from partially labeled data. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1631–1638, 2006.

[86] Yanmin Sun, Mohamed S. Kamel, and Yang Wang. Boosting for learning multiple classes with imbalanced class distribution. In *Proc. International Conference on Data Mining (ICDM)*, ICDM '06, pages 592–602, 2006.

[87] Yanmin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[88] Yanmin Sun, Andrew K. C. Wong, and Mohamed S. Kamel. Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009.

[89] Yanmin Sun, Andrew K. C. Wong, and Yang Wang. Parameter inference of cost-sensitive boosting algorithms. In *Proc. International Conference on Machine Learning and Data Mining (MLDM)*, pages 21–30, 2005.

[90] Yijun Sun, Zhipeng Liu, Sinisa Todorovic, and Jian Li. Adaptive boosting for sar automatic target recognition. *IEEE Transactions on Aerospace and Electronic Systems*, 43(1):112–125, 2007.

[91] Yijun Sun, Sinisa Todorovic, and Jian Li. Unifiyng multi-class adaboost algorithms with binary base learners under the margin framework. *Pattern Recognition Letters*, 28:631–643, 2007.

[92] Matus J. Telgarsky. The fast convergence of boosting. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1593–1601, 2011.

[93] Kai Ming Ting. A comparative study of cost-sensitive boosting algorithms. In *Proc. International Conference on Machine Learning (ICML)*, pages 983–990, 2000.

[94] Kai Ming Ting and Zijian Zheng. Boosting cost-sensitive trees. In *Proc. International Conference on Discovery Science (DS)*, pages 244–255, 1998.

[95] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 762–769, 2004.

[96] Tomasz Trzcinski, Mario Christoudias, Vincent Lepetit, and Pascal Fua. Learning image descriptors with the boosting-trick. In *Advances in Neural Information Processing Systems 25*, pages 269–277, 2012.

[97] L. G. Valiant. A theory of the learnable. *Journal of Computer and System Sciences*, 27(11):1134–1142, 1984.

[98] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[99] Nuno Vasconcelos and Mohammad J. Saberian. Boosting classifier cascades. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2047–2055, 2010.

[100] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[101] Paul A. Viola and Michael J. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1311–1318, 2001.

[102] Junhui Wang. Boosting the generalized margin in cost-sensitive multiclass classification. *Journal of Computational and Graphical Statistics*, 22(1):178–192, 2013.

[103] Shuo Wang and Xin Yao. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(4):1119–1130, 2012.

[104] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Elsevier, 2005.

[105] Fei Wu, Yahong Han, Qi Tian, and Yueting Zhuang. Multi-label boosting for image annotation by structural grouping sparsity. In *ACM International Conference on Multimedia (ACM-MM)*, pages 15–24, 2010.

[106] Fen Xia, Yan-wu Yang, Liang Zhou, Fuxin Li, Min Cai, and Daniel D. Zeng. A closed-form reduction of multi-class cost-sensitive learning to weighted multi-class learning. *Pattern Recognition*, 42(7):1572–1581, 2009.

[107] Xun Xu and Thomas S. Huang. Soda-boosting and its application to gender recognition. In *Analysis and Modeling of Faces and Gestures (AMFG)*, pages 193–204, 2007.

[108] Rong Yan, Jelena Tesic, and John R. Smith. Model-shared subspace boosting for multi-label classification. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 834–843, 2007.

[109] Jieping Ye. Least squares linear discriminant analysis. In *Proc. International Conference on Machine Learning (ICML)*, 2007.

[110] Yin Zhang and Zhi-Hua Zhou. Cost-sensitive face recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1758–1769, 2010.

[111] Zhi-Hua Zhou and Xu-Ying Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010.

[112] Ji Zhu, Hui Zou, Saharon Rosset, and Trevor Hastie. Multi-class adaboost. *Statistics and Its Interface*, 2:349–360, 2009.

[113] Hui Zou, Ji Zhu, and Trevor Hastie. The margin vector, admissible loss and multi-class margin-based classifiers. Technical report, University of Minnesota, 2007.

[114] Hui Zou, Ji Zhu, and Trevor Hastie. New multicategory boosting algorithms based on multicategory fisher-consistent losses. *Annals of Applied Statistics*, 2:1290–1306, 2008.