# Handling imbalanced datasets by reconstruction rules in decomposition schemes

Roberto d'Ambrosio

Università Campus Bio-Medico di Roma
School of Engineering
PhD Course in Biomedical Engineering
(XXVI - 2011/2013)

Université de Nice - Sophia Antipolis
École doctorale STIC
Sciences et Technologies de l'Information et de la Communication

A dissertation presented by

Roberto D'Ambrosio

in partial fulfillment
of the requirements for the degree of

*Doctor of Philosophy in Biomedical Engineering*
at Università Campus Bio-Medico di Roma

*Doctor of Science and Technologies of information and Communication*
at Université de Nice - Sophia Antipolis

# Handling Imbalanced Datasets by Reconstruction Rules in Decomposition Schemes

Defended on the 7th March 2014

# Contents

# List of Figures

# List of Tables

# Abstract

Disproportion among class priors is encountered in a large number of domains making conventional learning algorithms less effective in predicting samples belonging to the minority classes. Most of the proposals reported in the literature deal with classification imbalanced problems composed of two classes usually named as dichotomies, whereas small efforts have been directed towards multiclass recognition tasks, which are referred to as polychotomies. In this latter case existing approaches can be roughly divided in two groups. In the first one there are the methods addressing directly the multiclass tasks, whereas in the second group there are the methods using *decomposition schemes*. A decomposition scheme divides the polychotomy into several binary classification tasks, and then a *reconstruction rule* combines the predictions of binary learners to estimate the final decisions. In the case of a skewed classification task addressed by a decomposition approach, existing methods propose specialized classification algorithms that work at level of single binary learners, while little efforts have been directed to develop a reconstruction rule specifically tailored for class imbalance learning. On this motivation, we aim at developing a reconstruction rule suited to multiclass skewed data. In performing this task we look with interest to the classification reliability i.e. a measure of the goodness of classification acts. This quantity takes into account phenomena like noise, borderline samples, etc., and conveys useful information on the classification process. Hence, we decide to use these information in reconstruction rule tailored for imbalanced domains. In the framework of One-per-Class decomposition scheme we design a novel reconstruction rule, which is referred to as Reconstruction Rule by Selection. This rule uses classifiers reliabilities, crisp labels and a-priori samples distribution to compute the final decision. Experimental tests carried out with four classifiers on several public and artificial datasets show that system performance improves using this rule rather than using well-established reconstruction rules. Furthermore, the use of this rule improves classification performance in terms of accuracy, geometrical mean of accuracies per class and $F\,measure$, proving to be suited for skewed classification task. To further explore the effects of reconstruction rule in handling imbalanced domains we investigate a statistical reconstruction rule in the Error Correcting Output Code (ECOC) decomposition framework. Inspired by a statistical reconstruction rule designed for the One-per-Class and Pair-Wise Coupling decomposition approaches, we have developed a rule for ECOC scheme that applies softmax regression in order to estimate the final classification. To exploit the information provided by the reliability, we introduce this quantity in the reconstruction rule. Experimental results show that this choice improves the performances with respect to the existing statistical rule extended to ECOC framework, as well as to other well-established reconstruction rules. On the topic of reliability estimation we notice that several methods exist to estimate reliability and, in certain cases, posterior probability. Nevertheless small attention has been given to efficient posteriors estimation in the boosting framework. On this reason we develop an efficient posteriors estimator by boosting Nearest Neighbors. Using Universal Nearest Neighbours classifier we prove that a sub-class of surrogate losses exists, whose minimization brings simple and statistically efficient estimators for Bayes posteriors. Furthermore, we perform tests to evaluate the contribution of posterior estimation to set the final decision of the Universal Nearest Neighbors classifier. Results show also that the posterior reliability used at the reconstruction stage leads to an improvement of the system performance.

# 1. Introduction

Machine learning is the field that concerns the study of the algorithms that can learn from data [14]. These algorithms find application in a wide range of fields: speech and handwriting recognition, computer vision and object recognition, medical diagnosis, brain-machine interfaces, information retrieval and affective computing, to name a few. The large diffusion of these systems is due to the heterogeneity of the data that they can process: images, video sequences, signals, measures, etc. These raw data are typically preprocessed to transform them into some new space of variables where, it is hoped, the pattern recognition problem will be easier to solve. This pre-processing stage is also called *feature extraction* and maps the raw data into a vector of values referred to as *features vector*. The categories of the data, referred also as *classes*, are known in advance, typically by inspecting them individually and hand-labelling them with a *label*. Where samples belong to two ore more classes are named as binary or multiclass classification tasks. Furthermore, they are also referred to as *dichotmies* and *policotmies*, respectively.

Techniques existing in machine learning field can be roughly divided in three main branches. The first one deals with pattern recognition problems where the training data consists of a set of input vectors without any corresponding target values. The goal in such *unsupervised learning* problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization [10, 66, 93, 95].

The second one is the technique of *reinforcement learning* that concerns with the problem of finding suitable actions to take in a given situation in order to maximize a reward [77, 134]. Here the learning algorithm discovers the optimal outputs by a process of trial and error. Typically there is a sequence of states and actions in which the learning algorithm is interacting with its environment.

The last one concerns applications where the training data comprises examples of the input vectors along with their corresponding target vectors are known. This problem are referred to as *supervised learning*. Cases in which the aim is to assign each input vector to one of a finite number of discrete categories, are called classification problems. If the desired output consists of one or more continuous variables, then the task is called regression.

We focus in this work on supervised learning and in particular on classification problems. In these problems, a collection of samples is used to tune the parameters of an adaptive model. This provides to the classifier the knowledge of the problem at hand. This phase is called *training phase*, also known as *learning phase*, and the set of samples used is referred to as *training set*. During this training stage, the model's parameters of the learner are computed minimizing a loss function which reduces the error rate on the training set. After the training stage, new samples are presented to the learner, which assigns a label accordingly with its model. This step is called *testing phase*. Since in practical applications, the input vectors can comprise only a tiny fraction of all possible input vectors, classifier knowledge is limited. This issue limits the learner's generalization ability, i.e. the ability to infer information on unknown data. The number of misclassifications can depend also on several factors such as overlapping of class distributions, borderline samples, dataset noise, to name a few.

Our work deals with particular attention to the *Class imbalance learning* that refers to classification problems where datasets have a disproportion between class priors. The skewed distribution makes many conventional learning algorithms less effective, especially in predicting samples belonging to the minority classes. This happens because they are designed to minimize errors over training samples, and also assume or expect balanced class distributions. Therefore, when skewed datasets are presented to most standard learning algorithms, this cause an improper representation of data distributive characteristics, producing a bias towards the majority classes and providing unsatisfactory accuracies across the classes composed of few instances. When this phenomenon occurs in real-world domains, skewed data represents a recurring problem of high importance with wide-ranging applications such as text classification, currency validation, medical diagnosis and protein fold classification, to name a few [20, 37, 52, 104, 116, 132, 130, 135, 151, 152]. The relevance of this issue and its potential impact on the development of learning algorithms suited for real-world domains have motivated recent research on class imbalance learning. In this respect, most of the existing literature concerns binary classification problems while smaller efforts have been directed towards multiclass recognition tasks.

In case of binary problems, existing solutions work at pre-classification stage, at algorithmic level and at post-classification stage. At pre-classification level they provide different forms of resampling, such as undersampling and oversampling [5, 13, 44, 47, 48, 52, 56, 57, 64, 70, 73, 85, 91, 142, 92]. At algorithmic level they introduce a bias to compensate the skewness of the classes, e.g. using ensemble techniques and adjusting the costs of classes [9, 25, 41, 43, 68, 110, 74, 94, 137, 144, 145]. At post-classification stage they adjust decision thresholds or combine several learners in an ensemble system [17, 20, 71, 84, 92, 104, 108, 116, 121, 132, 140, 151] .

The large number of domains where samples belong to more than two classes pose new challenges that have not been observed in two classes problems [139, 156]. When classes have different misclassification costs, Zhou *et al.* [156] showed that it is harder to cope with a polychotomy than a dichotomy. They reported that most of learning techniques originally designed only for two-class scenarios are less effective, or even cause a negative effect when applied to multiclass tasks. Recent works tackling with imbalanced multiclass distributions can be roughly divided into two groups. In the first one there are the approaches directly addressing the polychotomy [1, 96, 139, 147, 156]. In the second group there are the approaches handling multiclass imbalance problems using decomposition schemes, which reduce the multiclass problem in less complex binary subtasks, each one addressed by a dichotomizer [3]. In this framework the three most popular decomposition schemes are One-per-Class (OpC), Pairwise Coupling (PC) and Error Correcting Output Code (ECOC) [3, 40, 46, 50, 55, 72, 111, 123, 128, 148, 150]. To provide the final classification, dichotomizers' outputs are combined according to a *reconstruction rule*. It is worth noting that results of experiments carried out by Alejo *et al.* [1] show that a decomposition approach achieves larger recognition performances than directly addressing the polychotomy. This last result agrees with [111], where the authors prove that OpC is preferable to a more complex error-correcting coding scheme or a single-machine scheme. Focusing on using decomposition approaches in multiclass imbalance learning, we observe that most of the aforementioned proposals work at level of single dichotomizer [24, 46, 50, 90, 126, 135, 155], while little efforts have been directed to develop a reconstruction rule specifically tailored for class imbalance learning.

On these motivations, we aim at developing new reconstruction rules suited for imbalanced domains. In order to achieve this goal, we look with interest to the classification reliability, which is a measure of the classifier's "confidence" on its predictions. A large value of the reliability suggests that the recognition system is likely to provide a correct classification [27, 82]. Conversely a low value of reliability suggests that the decision on the test sample is not safe. This quantity takes into account several issues influencing the achievement of a correct classification such as border line samples, dataset noise, outliers, etc.

Considering these interesting characteristics of the reliability, we decide to use this quantity in the reconstruction rule in order to deal with imbalanced domains. Hence, we develop an heuristic reconstruction rule in the OpC decomposition framework suited to classify skewed data [30, 34]. The key idea of this approach is that learner reliabilities can be used to detect classification acts where the presence of an imbalanced distribution among the classes is leading to a misclassification. To this aim, our rule therefore incorporates the reliabilities at reconstruction stage in order to correct possible misclassifications. We carried out tests on several imbalanced domains, both real and synthetic, using four different classification algorithms. Tests results point out two main contributions. First, this rule provides larger overall performance compared with well-established reconstruction rules. Second, our rule is suited for classify imbalanced domains since geometric means of accuracies and $F\,measure$ show that this rule improves performance with respect to minority classes.

In a decomposition scheme each binary learner outputs its prediction on the input sample. The collection of these predictions build a vector that maps the sample into a new space and thus it can be considered as a new feature vector. Hence, after binary classification, test samples are described in a new set of second order features that, together with the original labels, define a new classification task. Considering the problem from this point of view, we further investigate the use of the reconstruction rule in order to handle imbalanced datasets. We propose a statistical reconstruction rule extending an existing method, suited for One-per-Class and Pairwise Coupling, in the case of Error Correcting Output Code (ECOC) [31, 32]. This rule applies the softmax regression on the feature vectors generated by binary learners. In order to achieve improvements with respect to the minority classes we integrate classifier reliabilities in the reconstruction stage. Results show that the rule provides satisfactory performance when compared with well established rules in the ECOC framework and when compared with the softmax regression without the use of reliability.

Exploring the reliability issue, we became aware that several methods to compute reliability measure from classifier outputs exist. In some cases, it is even possible to compute classification reliability in terms of posterior probability [18, 60, 107, 154]. Among all the proposals, to the best of our knowledge, little efforts have been directed toward efficient posteriors estimation in boosting approach. Boosting algorithms are remarkably simple and efficient from the classification standpoint, and are being used in a rapidly increasing number of domains and problems [18]. Nevertheless it is widely believed that boosting and conditional class probability estimation are in conflict with each other, as boosting iteratively improves classification at the price of progressively over-fitting posteriors [19, 54]. Existing experimental results display that this estimation is possible, but it necessitates a very fine tuning of the algorithms [18].

For this reason, we propose a novel efficient posterior estimator by boosting Nearest Neighbors. We use the Universal Nearest Neighbours demonstrating that a sub-class of surrogate losses exists, whose minimization brings simple and statistically efficient estimators for Bayes posteriors [33]. The point of our work is that boosting topological approaches, like nearest neighbors, is possible to estimate class conditional probabilities, without tedious tunings, and without overfitting. Furthermore, experimental results show that the use of the estimated posterior probabilities to set the final decisions leads to an improvement of system performances.

The thesis is organized as follows. The next chapter presents an overview of the literature related to the issue of classify imbalanced datasets and the rationale behind our research activities. Chapter 3 presents the datasets, the learners and the performance metrics used to validate our proposals. Chapters 4,5 and 6 present our contributions and, finally, Chapter 7 summarizes results obtained with the proposed approaches.

# 2. Background

In an imbalanced dataset the most relevant source of misclassification is the skewed data distribution between classes. Many authors pointed out that the problem of imbalance distribution occurs often together with other phenomena that influence the performance of learning algorithms in detecting the minority samples. As an example in [73] authors show that dataset complexity and the presence of small disjunctions can cause a degradation in standard classifiers' performance. The difficulty in separating the small class from the prevalent one is the key issue in this task and if patterns of each class are overlapping at different levels in some feature space, discriminative rules are hard to induce.

Furthermore one of the critical factors in learning from imbalanced datasets is the sample size. When the sample size is limited, uncovering regularities inherent in small class is unreliable. In [70] authors report that as the size of the training set increases, the error rate caused by the imbalanced class distribution decreases. When more data can be used, relatively more information about the minority class benefits the classification modeling, which becomes able to distinguish rare samples. It is obvious that in real life problems it is not always possible to increase the size of the sample.

Therefore the imbalance distribution issue is rather complex and in general it is not easily solvable [49, 63, 65, 83, 131]. It is not limited to binary classification tasks (dichotomies) and it holds also in multiclass problems (polichotomies). In the latter case an imbalanced dataset has one or more classes with fewer samples than others.

A number of proposed solutions can be tracked back in the literature to solve imbalanced datasets issue. These solutions have been focused mainly in case of binary problems whereas contribution for multiclass tasks is still limited.

In the following we firstly present the solutions proposed to solve binary problems distinguishing between methods addressing the imbalanced issue at pre-classification level, in-algorithms approaches and post-classification techniques. Secondly we present solutions that aim at solving multiclass tasks.Thirdly, we describe the decomposition techniques. Finally, we introduce the classification reliability.

## 2.1. Binary Methods

Many solutions have been proposed to handle imbalanced dataset issue in case of binary classifications. As reported in figure 2.1 these methods are divided in three main areas: *Pre-classification*, *In-Algorithms* and *Post-classification* techniques. The objective of the formers is to re-balance the class distributions by resampling the data space. At the algorithm level, solutions try to adapt existing classifiers to strengthen learning with regards to the small class. Post-classification techniques combine classifier outputs or tune prediction thresholds in order

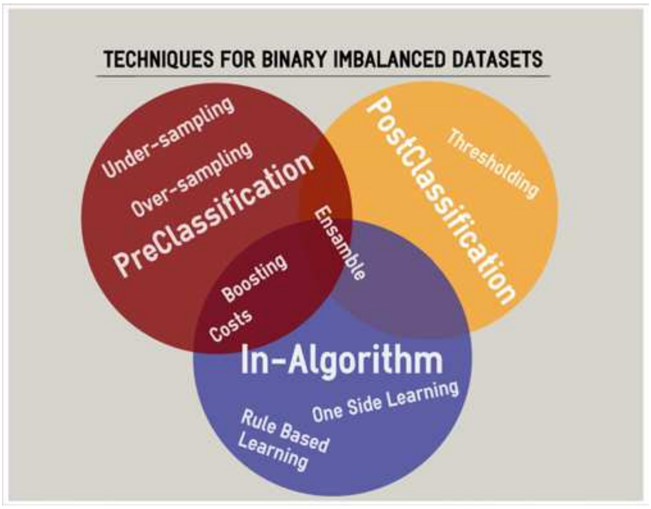

Figure 2.1.: Techniques tassonomy for binary inbalanced dataset tasks

to reduce the misclassification in the minority class. As reported in the figure, often the division in this taxonomy is not strict since in many cases proposals use techniques belonging to different areas.

## 2.1.1. Pre-classification techniques

At the data level, different forms of re-sampling methods have been proposed aiming at generate balanced data distributions. Indeed in the specialized literature, several papers study the effect of changing class distributions empirically proving that a preprocessing step is usually a positive solution [13, 44, 47, 48, 91].

In [70] the effect of imbalance in a dataset is discussed and two re-sampling strategies are considered. *Random re-sampling* consists of re-sampling the smaller class at random until it consists of as many samples as the majority class, whereas *focused re-sampling* consists of re-sampling only those minority instances that occur on the boundary between the minority and majority classes. Experiments in [70] show that both the two sampling approaches are effective, and the author proves that using more sophisticated sampling techniques do not give any clear advantage in the domain considered.

In addition to these classical re-sampling methods, many others have been presented in the literature, such as heuristic re-sampling methods, combination of over-sampling and under-sampling methods, embedding re-sampling methods into data mining algorithms, and so on.

Examples of proposals regarding improved under-sampling methods are as follows. In [85] authors presented the *one-side selection* under-sampling method, which heuristically balances the dataset through eliminating the noise and redundant examples of the majority class. The majority class instances are classified as ”safe”, ”borderline” and ”noise” instances. Borderline and noisy cases are detected using Tomek links, and are removed from the dataset. Only safe majority class instances and all minority class instances are used for training the learning system. In [56, 57] authors propose an under-sampling procedure where genetic algorithms are applied for the correct identification of the most significant instances. A features and instances selection method [52] has been tested on imbalanced domains in [130]. Authors' method computes the discriminative power of each feature and then selects those samples that show the highest score. In [5] author proposes a Condensed Nearest Neighbour Rule that performs under-sampling bases on the notion of a consistent subset of a sample set, which is a subset who can correctly classifies all of the remaining examples in the training set when used as a stored reference set for the NN rule. One of the advantages of this algorithm is the fast learning speed. In [142] the Edited Nearest Neighbour Rule is proposed. This algorithm removes any example whose class label differs from the class of at least two of its three nearest neighbours. In [92] two under-sampling algorithms are presented: EasyEnsemble and BalanceCascade. The first one samples several subsets from the majority class, trains a learner using each of them, and combines the outputs of those learners. The second one trains the learners sequentially, where in each step, the majority class examples that are correctly classified by the current trained learners are removed from further consideration.

Among proposals of over-sampling techniques there is SMOTE (Synthetic Minority Oversampling Technique) method [22], which generates new synthetic instances along the line between the minority examples and their selected nearest neighbours. Authors show that best performances are achieved combining SMOTE and under-sampling. The advantage of SMOTE is that it makes the decision regions larger and less specific. In [64] authors propose two methods based on SMOTE aiming at oversampling only the minority examples near the borderline: borderline-SMOTE1 and borderline-SMOTE2. The key idea of this approach is that borderline examples of the minority class are more easily misclassified than those ones far from the borderline. In [73] authors put forward a cluster-based over-sampling method which deals with between-class imbalance and within-class imbalance simultaneously. The idea behind this method is that classification performances drop when class imbalanced problem is related also with the problem of small disjunctions.

Other methods propose a combination between over-sampling and under-sampling to resolve the imbalance distribution problem. In [13] authors show results that contradict the literature. Testing ten different methods they show that over-sampling methods are more accurate than under-sampling methods. In [44] there is an attempt to investigate three aspects: i) which one is the most performing technique between under-sampling and over-sampling; ii) which one is the ideal re-sampling rate; iii) if it is possible to combine re-sampling methods to improve classification performance. Authors finally propose a method that performs multiple re-sampling, both oversampling and under-sampling, selecting the most appropriate re-sampling rate adaptively. Authors in [43] report that when using C4.5s default settings, over-sampling is surprisingly ineffective, often producing little or no change in performance in response to modifications of misclassification costs and class distributions. Moreover, they noted that over-sampling prunes less the trees and therefore generalizes less than under-sampling.

The level of imbalance is reduced in both under-sampling and over-sampling methods, with the hope that a more balanced training set can give better results. Both sampling methods are easy to implement and have been shown to be helpful in imbalanced problems. Both methods have also drawbacks. Under-sampling requires shorter training time, at the cost of ignoring potentially useful data. Oversampling increases the training set size and thus requires longer training time. Furthermore, it tends to lead to over-fitting since it repeats minority class examples. In this way, a symbolic classifier, for instance, might construct rules that are apparently accurate, but actually, cover one replicated instance.

### 2.1.2. In Algorithms

In this section we report all the methods that operate on the algorithm rather than at dataset level.

One of the most used algorithms in the field of machine learning is the Support Vector Machine (SVM). In its traditional form, when deals with imbalanced data sets, it increases the misclassified rate of the minority class. For this reason several attempts have been done to modify internally this classifier to hanlde imbalanced distributions [74, 137, 144, 145]. In [74] authors present an SVM that can directly optimize a large class of performance measures (e.g. $F\,measure$, Precisoon/Recall at Breakeven point) formulating the problem as a multivariate prediction of all the examples. In [137] two methods to control the sensitivity and specificity of the SVM are proposed. With this aim the authors introduce different loss functions for positive and negative samples. In [144, 145] authors modify the kernel matrix according to the imbalanced data distribution. This is done in order to compensate for the skew associated with imbalanced datasets which pushes the hyper-plane closer to the positive class.

The SVM is not the only classifier that has been modified to solve the problem of imbalanced data. In [9] authors try to compensate for the imbalance in the training sample without altering the class distributions. They use a weighted distance in the classification phase of kNN. Thus, weights are assigned, unlike in the usual weighted k-NN rule, to the respective classes and not to the individual prototypes. In this way, since the weighting factor is greater for the majority class than for the minority one, the distance to positive minority class prototypes becomes much lower than the distance to prototypes of the majority class. This produces a tendency for the new patterns to find their nearest neighbours among the prototypes of the minority class. C4.5 algorithms performances in [43] are evaluated when re-sampling techniques are used together with algorithm parameters tuning. It is shown that over-sampling is ineffective if C4.5s parameter to increase the influence of pruning and other over-fitting avoidance factors are not well set. In [68] authors propose the Biased Minimax Probability Machine to resolve the imbalance distribution problem. Given the reliable mean and covariance matrices of the majority and minority classes, this algorithm can derive the decision hyper-plane by adjusting the lower bound of the real accuracy of the testing set.

Furthermore, there are other effective methods such as one-class learning [25, 94] and cost-based learning.

The first strategy, i.e. One-class learning, creates the learning model using only examples from the positive class. Differently from a discriminative based approach that distinguishes between positive and negative samples, the One-class learning recognizes only the samples of the minority class. Hence, it belongs to recognition-based approaches. In [25] authors show that one-class learning from positive class examples can be very robust classification technique when dealing with imbalanced data. They argue that the one-class approach is related to aggressive feature selection methods, but is more practical since feature selection can often be too expensive to apply. Algorithms such as SHRINK that looks for the *best positive region* and BRUTE [110] that performs an exhaustive search for accurate predictive rule belong also to this category.

The second approach integrates costs during the decision making process leading to an improvement in performance in case of imbalanced domain. The cost matrix is usually expressed in terms of average misclassification costs for the problem. The goal in cost sensitive classification is to minimize the cost of misclassification, which can be realized by choosing the class with the minimum conditional risk. In [41] authors propose a general framework that makes an arbitrary classifier costs sensitive. This procedure is called MetaCost and it estimates class probabilities using bagging and then relabels the training examples with their minimum expected cost classes, and finally relearns a model using the modified training set. This approach can be used with any number of classes. Standard boosting algorithms , e.g. Adaboost, increase the weights of misclassified examples and decrease the weights of those correctly classified. The weights updating rule is uniform for all the samples and does not consider the imbalance of the data sets. For this reason these algorithms do not perform well on the minority class. In [45] authors propose a cost sensitive version of Adaboost referred to as Adacost. In this algorithm to the examples belonging to rare class that are misclassified are assigned higher weights than those belonging to common class. It is empirically shown that the proposed system produces lower cumulative misclassification costs than AdaBoost. In [76] an improved boosting algorithm is proposed, which updates weights of positive predictions differently from weights of negative predictions. It scales false-positive examples in proportion to how well they are distinguished from true-positive examples and scales false-positive examples in proportion to how well they are distinguished from true-negative examples, allowing the algorithm to focus on both Recall and Precision equally. The new algorithm can achieve better prediction for the minority class. In SMOTEBoost [23] authors recognize that boosting may suffer from the same problems as over-sampling (e.g., over-fitting), since will tend to weight examples belonging to the rare classes more than those belonging to the common classes. For this reason SMOTEBoost alters the distribution by adding new minority-class examples using the SMOTE algorithm. The synthetic samples for the rare class are added into the training set to train a weak classifier and discarded after the classifier is built. The SMOTE procedure in each iteration makes every classifier learn more from the rare class, and thus broadens the decision regions for the the rare class. All these Boosting variation can be applied to binary problems as well as to multiclass tasks. An analysis of the cost-sensitive boosting algorithms is reported in [133].

Other variations of traditional classification algorithms have been proposed in the area of rule based algorithms. Indeed these traditional methods often show poor performances when learned from imbalanced datasets. We have already introduced BRUTE algorithm [110] where brute-force induction is applied in Boeing manufacturing domain. In [2] authors use Emerging Patterns (EPs) [42] to handle imbalanced problems. The algorithm works in three stage: generating new undiscovered rare class EPs, pruning low-support EPs and increasing the supports of rare class EPs. In [75] authors propose a two-phase rule induction method in the context of learning complete and precise signatures of minority classes. The first phase aims for high recall by inducing rule with high support and reasonable level of accuracy. The second phase tries to improve the precision by learning rules to remove false positive in the collection of the records covered by the first phase.

### 2.1.3. Post-classification techniques and ensemble

One of the strategy adopted in the post classification stage is the tuning of decision thresholds [116, 121] . Provost [108] highlights that adjusting decision thresholds is a critical factor in classification of imbalanced data. This strategy is adopted in [132] where authors show that SVM threshold relaxation can be used in hierarchical text classification to avoid blocking documents at high-level categories in the hierarchy. Some classifiers, such as the Naive Bayes classifier or Neural Networks, supply a score that represents the degree to which an example belongs of a class. Such ranking can be used to tune the final decision by varying the threshold of an example pertaining to a class [140] . In [17] authors, in addition of experiments with cost-based adjustment of the dividing hyperplane, show that the learner achieves improved performance mostly altering the score threshold directly.

An Ensemble learner [39] contains a number of learners which are usually called base learners that are combined by combination schemes. Since ensemble learning has established its superiority in machine learning, in recent years many attempts have been done in using ensemble systems to handle imbalanced domains. In ensemble systems results of several classifiers are combined to provide the final prediction. The diversity among base classifiers guarantees an improvement in final system performance. The diversity can be achieved also by using various class distributions. Boosting algorithms like Adacost [45], Rare-Boost [76] and SMOTEBoost [23] are enclosed also in this category.

One of the most adopted strategy in ensemble systems is to generate many subsets starting from the original distribution. Usually this datasets are generated through re-sampling techniques. In Section 2.1.1 we have already introduced this method when we described the two systems in [92]: EasyEnsemble and BalanceCascade. In [20] and in [151] authors, starting from the original dataset, generate many subsets each one containing all the minority class examples and an equal number of samples drawn from the majority one. In the first work it is presented wrapper method where each learning algorithm is trained using a subset and the final decision is taken according to a stacking strategy. It can be used with any learning method internally. In the second work authors use an ensemble system of SVMs. They show that this method is more stable than the traditional re-sampling techniques.

In [104] authors present a method that, in the domain of fraud detection, uses a single meta-classifier to choose the best base classifiers, and then combine their predictions to improve costs-saving. The data subsets are generated trough oversampling of minority class. They show that their stacking-bagging procedure achieves the highest costs saving which is almost the twice of the conventional back-propagation procedure.

In [84] authors use techniques of agent-based knowledge discovery to handle the problem of imbalanced datasets. They use three agents (the first learns using Naive Bayes, the second using C4.5 and the third using 5NN) on a filtered version of training data and combine their predictions according to a voting scheme. The intuition of authors is that the models generated using different learning are more likely to make errors in different way and thus increase the diversity of system.

In [71] the author combines classification techniques from both supervised and unsupervised learning. He uses an unsupervised method of re-labeling already labelled data. A classifier is then run on several version of the same dataset and their results are combined using a

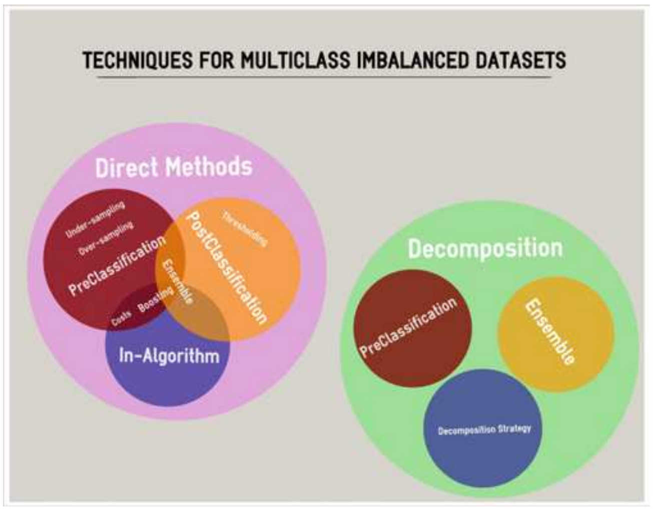

Figure 2.2.: Techniques tassonomy for multiclass inbalanced dataset tasks

voting techniques.

## 2.2. Multiclass

It is known that it is harder to cope with a polychotomy than a dichotomy when classes have different misclassification costs [156]. In [156] authors show that most of learning techniques originally designed only for two-class scenarios are less effective or even cause a negative effect when applied to multiclass tasks. The large number of domains where samples belong to more than two classes defines new challenges that have not been observed in two classes problems [139, 156].

Recent works tackling with imbalanced multiclass distributions can be roughly divided into two groups (figure 2.2). In the first one there are approaches directly addressing the polychotomy [1, 96, 139, 156]. In the second group there are approaches handling multiclass imbalance problems using class decomposition schemes [24, 46, 90, 126, 135, 155]. To provide the final classification, dichotomizers' outputs are combined according to a reconstruction rule. It is worth noting that, results of experiments carried out by Alejo et al. [1] show that a decomposition approach achieves larger recognition performances than directly addressing the polychotomy.

### 2.2.1. Direct Methods

As in the case of binary methods we can divide the proposal to handle multiclass imbalanced classes at the level of Pre-classification, In-Algorithm and Post-classification.

In the first category there is the method proposed in [146] where on the majority classes is applied a local clustering, whereas on the minority ones, is adopted oversampling. The algorithm adjusts the over-sampling parameter to match with the clustering result so that the rare class size is approximate to the average size of the partitioned majority classes.

In the second category there are works such as: [1, 26, 96]. RIPPER [26] algorithm is a rule induction system that utilizes a separate-and-conquer approach to iteratively build rules to cover previously uncovered training examples. Each rule is grown by adding conditions until no negative examples are covered. It normally generates rules for each class from the most rare class to the most common class. Given this architecture, it is quite straightforward to learn rules only for the minority class, a capability that Ripper provides. In [1] authors introduce several cost functions in the learning algorithm in order to improve the generalization ability of the networks and speed up the convergence process. In [96] a two stage evolutionary algorithm is presented with two sequential fitness functions, the entropy for the first step and the area for the second one. This algorithm is based on the accuracy and minimum sensitivity given by the lowest percentage of examples correctly predicted to belong to each class. The two-stage approach obtains high classification rate level in the global dataset with an acceptable level of accuracy for each class.

In the last category we report, as in the case of binary, thresholding techniques [152] and ensemble learning systems[37]. The use of the threshold is studied in [152] where three thresholding strategies in text classification are studied on the performance of a kNN classifier. The author shows that proportional thresholding performs well in classifying minority class samples for multicategory classification tasks. A system that automatically discovers classification patterns by applying several empirical learning methods to different representation of datasets is presented in [37] in the field of document categorization. Different representations of the datasets are obtained performing feature selection based on genetic algorithm. The final document category is obtained by the genetic combination of the decision made by all the learners.

Among the methods that address directly the problem of multicass imbalanced datasets we include SMOTEBoost [23] , MetaCost [41] and AdaCost [45] that can be applied both to binary and to multiclass tasks.

In [156] the effect of sampling and threshold-move is empirically studied in a training cost-sensitive neural networks. Both over-sampling and under-sampling are considered. The threshold is moved toward inexpensive classes such that examples with higher costs become harder to be misclassified. Furthermore the effect of hard and soft voting is also used to build the ensemble decision. This paper can be categorized in all the three sections. Indeed, re-sampling techniques, a cost-sensitive classifier, a threshold moving strategy and finally an ensemble decision are used in the approach proposed by the authors.

### 2.2.2. In Decomposition Framework

Given a polychotomy with $K > 2$ classes, decomposition methods can be traced back to the following three categories [3, 24, 40, 46, 72, 90, 126, 128, 135, 155]: One-per-Class (OpC), Pairwise Coupling (PC), and distributed output code. There exist other proposals that do not perfectly fit this categorization, e.g. the hierarchical dichotomies generation [86], but this does not introduce any limitations in the rest of the work. We provide in Section 2.3 a complete description of decomposition methods, whereas we report the description of approaches that handle imbalanced multiclass datasets based on these schemes [24, 46, 90, 135] in the following. In [24] authors use the min-max modular network to decompose a multi-label problem into a series of small two-class subproblems. They present several decomposition strategies to improve the performance of min-max modular networks showing that the proposed method has better generalization than SVM. In [46] authors use pairwise coupling framework where each sub-problem is processed with SMOTE algorithm in order to balance data distribution. As a base classifier is used a linguistic Fuzzy Rule Based Classication system. The experimental results support the goodness of their methodology as it generally outperforms the basic and pairwise learning multi-classifier approach. In [90] 22 data preprocessing methods are tested to perform classification of weld aws with imbalanced classes in an OpC decomposition scheme. Their results show that some data preprocessing methods do not improve any criterion and they vary from one classifier to another. In [135] authors propose a novel ensemble machine learning method that improves the coverage of the classifiers under the multi-class imbalanced sample sets by integrating knowledge induced from different base classifiers, and they illustrate that the approach performs at least as well as the traditional technique over a single joined data source. Finally, in [50] authors experimentally study the contribution of re-sampling techniques in the OpC ans PC decomposition schemes.

## 2.3. Decomposition Methods

The techniques reducing a multiclass recognition problem to several binary subtasks are usually named as *decomposition methods*. Several proposals exist in the literature, and the most used ones are the One-per Class (OpC), the Pairwise Coupling (PC), and the distributed output code.

The first decomposition method, OpC, is also known as One-against-All. It is based on a pool of $K$ binary learning functions, each one separating a single class from all the others. Thus in the OpC framework the $j$th dichotomizer is specialized in the $j$th class when it aims at recognizing if the input sample belongs either to the $j$th class or, alternatively, to any other class. This decomposition scheme, even if it is often used to derive multiclass classifier by binary learning algorithms, has not received the same attention in literature as other rules. Some authors state that other schemes are preferable to OpC [50], nevertheless it has been proven that OpC performs as well as more complex error-correcting coding schemes or dicothomizer are well tuned [111].

The second approach, PC, it is also cited as $n^2$ classifier, One-against-One or even Round Robin classification [55]. In this case the recognition system is composed of $K * (K - 1)/2$ base dichotomizers, each one specialized in discriminating between pair of classes. Predictions of the base classifiers are then aggregated to a final decision using a voting criterion. For example, in [72, 128] the authors propose a voting scheme adjusted by the credibilities of the base classifiers, which are calculated during the learning phase of the classification. Indeed, in this case the typical approach consists in using the confusion matrix.

The third approach, distributed output code, assigns a unique codeword, i.e. a binary string, to each class. Assuming that the string has $L$ bits, the recognition system is composed by $L$ binary classification functions. Given an unknown sample, the classifiers provide an $L$-bits string that is compared with the codewords to set the final decision. For example, the input sample can be assigned to the class with the closest codeword according to a distance measure such as the Hamming distance. In this framework, in [40] the authors propose an approach, known as *Error-Correcting Output Codes* (ECOC), where the use of error correcting codes as distributed output representation yield a recognition system less sensitive to noise. This result can be achieved via the implementation of an error-recovering capability derived from the coding theory. Recently, other researchers investigated the ECOC approach proposing diversity measures between codewords and the output of dichotomizers that differ from the Hamming distance. For example, Kuncheva in [87] presents a measure accounting for the overall diversity in the ensemble of binary classifiers, whereas Windeatt [143] describes two techniques for correlation reduction between different codes. As regard to classification reliability in ECOC decompositions, in [40] the authors propose a reliability estimator based on two Euclidean distances: the first between the outputs of the base classifiers and the nearest codeword; the second between these outputs and the second-nearest codeword. The confidence is then estimated as the difference between these two distances. The limit of this approach is that the confidence does not explicitly depend upon the position of the pattern in the feature space.

More formally given a polychotomy with $K > 2$ classes represented by the label set $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$, the decomposition through the application of decomposition schemes generates a pool of $L$ dichotomizers each one denoted as $M_j$. When feed with a test sample $\boldsymbol{x} \in \Re^n$, each dichotomizer outputs the quantity $M_j(\boldsymbol{x})$, which is collected in the vector $\mathbf{M}(\boldsymbol{x}) = [M_1(\boldsymbol{x}), M_2(\boldsymbol{x}), \ldots, M_L(\boldsymbol{x})]$, with the value of $L$ depending upon the decomposition approach adopted. Decomposition schemes can be unified in a common framework representing the outputs of the dichotomizers by a binary code matrix, named as decomposition matrix $\boldsymbol{D} \in \Re^K$ x $\Re^L$. Its elements are defined as:

$$\boldsymbol{D}(\omega_c, j) = \begin{cases} 1 & \text{if class } c \text{ is in the subgroup associated to label 1 of } M_j \\ -1 & \text{if class } c \text{ is in the subgroup associated to label -1 of } M_j \\ 0 & \text{if class } c \text{ is in neither groups associated to label -1 or 1 of } M_j \end{cases} \tag{2.1}$$

with $c = \{1, 2, \ldots, K\}$. We also denote as $\mathbf{D}(\omega_c)$ the $c$th row of $\mathbf{D}$ which is the binary codeword associated to the class $c$. Hence, the labels are coded as $\{1, +1\}$ according to their class membership, whereas zero entries indicate that a particular class is not significative for a given dichotomy. Obviously, this latter situation occurs only in the PC approach.

When the decomposition system is fed by a test sample $\boldsymbol{x} \in \Re$, the $L$ binary classifiers outputs crisp or soft labels, which are collected into the *test codeword*. To set the final label, a reconstruction rule compares, according to a certain criterion, this test codeword with the base codewords associated to each class and defined in the matrix $\boldsymbol{D}$.

**Hard reconstruction rule.** In this case crisp decisions are made on the outputs of the binary learners. Using the previously introduced notation, the crisp output vector $\mathbf{M}(\boldsymbol{x}) = [M_1(\boldsymbol{x}), M_2(\boldsymbol{x}), \ldots, M_L(\boldsymbol{x})]$ (with $M_j(\boldsymbol{x}) = \{-1, 1\}$) contains the binary decisions provided by the dichotomizers for each sample $\boldsymbol{x} \in \Re^n$. A well-known reconstruction rule, usually referred to as *Hamming decoding* (HMD) [3], sets the index $s$ of the final class $\omega_s \in \Omega$ as:

$$s = argmin_c\, d_{\text{HMD}}(\mathbf{C}(\omega_c), \boldsymbol{M}(\boldsymbol{x})) \tag{2.2}$$

where

$$d_{\text{HMD}}(\mathbf{D}(\omega_c), \mathbf{M}(\boldsymbol{x})) = \sum_{j=1}^{L} \left( \frac{1 - sign(\mathbf{D}(\omega_c, j) M_j(\boldsymbol{x}))}{2} \right) \tag{2.3}$$

with $c = \{1, \ldots, K\}$. This reconstruction rule can be used whatever the type of the classifier, i.e. abstract, rank or measurement[1], since it requires the crisp labels, only.

**Soft reconstruction rule.** A disadvantage of the hard decoding technique is that it completely ignores the magnitude of the soft outputs, which represent an indicator of the reliability of the decision taken by the dichotomizer. Therefore, a common strategy is to consider the real-values $f_j(\boldsymbol{x})$ provided by the $j$th dichotomizer, which are collected in $\mathbf{f}(\boldsymbol{x})$. In many approaches this quantity is combined with the crisp label, thus computing the margin. The margin of a training sample is a number that is positive if and only if the sample is correctly classified by a given classifier and whose magnitude is a measure of confidence in the prediction. In case of a test sample, the margin of binary learners can be collected in the vector $\mathbf{m}(\boldsymbol{x})$, whose elements

$$m_j(\boldsymbol{x}) = M_j(\boldsymbol{x}) f_j(\boldsymbol{x}) \tag{2.4}$$

This vector is exploited looking for the binary learner returning the largest positive output [40]. Hence, the index $s$ of the final class $\omega_s$ is given by:

$$s = argmax_c\, \boldsymbol{m}(\boldsymbol{x}) \tag{2.5}$$

An extension of the original maximum rule was provided by Allwein *et al.* [3], which introduced the loss-based decoding (LBD). This rule is based on a loss function $\Gamma$ evaluated on the margin. Hence, the final label is given by equation 2.2 where $d_{\text{HMD}}$ is replaced by $d_{\text{LBD}}$ that is computed as follows:

$$d_{\text{LBD}}(\mathbf{D}(\omega_c), \mathbf{M}(\boldsymbol{x})) = \sum_{j=1}^{L} \Gamma(\mathbf{D}(\omega_c, j), f_j(\boldsymbol{x})) \tag{2.6}$$

It is worth observing that such an approach can be used also when the loss function of the dichotomizers is not known since can be substituted by either $L_1$ or $L_2$ norm distance [123].

[1]The various classification algorithms can be divided into three categories [150]: *type I* (*abstract*), that supplies only the label of the presumed class, *type II* (*rank*) that ranks all classes in a queue where the class at the top is the first choice, *type III* (*measurement*) that attributes each class a value that measures the degree that the input sample belongs to that class.

## 2.4. Classification Reliability

Classification performances are often deteriorated by several factors, e.g. the noise affecting the samples, borderline samples and the differences between the objects to be recognized and those used to train the classifier.

The classification reliability is a measure that takes into account such several issues influencing the achievement of a correct classification. It permits to estimate the "confidence" of a classifier in its classification act, providing useful information on classifier decision [27, 82, 113]. A large value of the reliability suggests that the recognition system is likely to provide a correct classification [27, 82]. Conversely a low value of reliability suggests that the decision on the test sample is not safe.

The reliability measure is very useful in a wide range of tasks. For instance, the reliability is used in ensemble learning to derive the "Weighted Voting" methods [67, 89] which works as follows: first it collects the crisp outputs of all the experts, second it computes their reliability, third it weights the outputs with the corresponding reliability and, fourth it assigns the sample to the class that shows the highest sum of votes.

Several approaches exist to compute classifier reliability. In general, the most common choice for evaluating the classification reliability is to use the confusion matrix or other measures that depend on the recognition performance achieved during the learning phase. For example, if an expert assigns the input sample to a certain class, a reliability proportional to the recognition rate achieved on the training set on that class is attributed to such a decision [150]. The drawback of this approach is that all the patterns attributed to the same class have equal reliability, regardless of the quality of the sample. Indeed, the average performance on the learning set, although significant, does not necessarily reflect the actual reliability of each classification act. However, several works have demonstrated that more effective solutions could be achieved by introducing parameters that estimate the accuracy of each single classification act of the system [27, 113, 153].

A reliability parameter should permit to distinguish between the two reasons causing unreliable classifications : (a) either the sample is significantly different from those presented in the reference set, i.e. in the feature space the sample point is far from those associated with any class, (b) the sample point lies in the region where two or more classes overlap. In [27] authors propose a reliability computation method, suited for Nearest Neighbours (NN) and Multi Layer Perceptron (MLP), that considers these situations. For each one of these two cases, it is defined a reliability parameter, named $\psi_a$ and $\psi_b$, respectively. Based on these definitions, the parameter providing an inclusive measure of the classification reliability can be defined as follows:

$$\psi = \min\left(\psi_a, \psi_b\right) \tag{2.7}$$

This form is conservative since it considers a classification unreliable as soon as one of the two alternatives causing unreliable classifications happens. The definition of both the parameters $\psi_a$ and $\psi_b$ relies on the particular classifier architecture adopted.

In the case of NN classifiers, following [27], the samples belonging to the training set are divided into two sets: the reference set and the training test set. The former is used to perform the classification of the unknown pattern $x$, i.e. it plays the role of training set for the NN classifier, whereas the latter provides further information needed to evaluate the $\psi_a$ parameter. More specifically, the two reliability estimators are defined as:

$$\psi_a = \max(1 - \frac{O_{min}}{O_{max}}, 0) \tag{2.8}$$

$$\psi_b = 1 - \frac{O_{min}}{O_{min2}} \tag{2.9}$$

where: $O_{min}$ is the distance between $\boldsymbol{x}$ and the nearest sample of the reference set, i.e. the sample determining the class $\omega_j(\boldsymbol{x})$, $O_{max}$ is the highest among the values of $O_{min}$ obtained from all samples of class $\omega_j(\boldsymbol{x})$ belonging to the training test set, and $O_{min2}$ is the distance between x and the nearest sample in the reference set belonging to a class other than $\omega_j(\boldsymbol{x})$. In the case of MLP classifiers the reliability can be estimated as:

$$\psi = \min(O_{win}, O_{win} - O_{2win}) = O_{win} - O_{2win} = \psi_b \tag{2.10}$$

where: $O_{win}$ is the output of the winner neuron, $O_{2win}$ is the output of the neuron with the highest value after the winner. The interested reader may find further details in [27]. Note that such estimators have been useful also in other applications, e.g. in [36, 124].

In general, the use of classification reliability does not limit the choice of a classifier architecture since it is always possible to obtain a soft label output for each classification act of any kind of classifier [69]. In some specific cases, it is even possible to compute classification reliability as posterior probability. One of the most known approach that maps classifier continuous output to posterior probability is the Platt sigmoid function [107]. This method transforms SVM continuous output, i.e. the distance from hyperplane ($h(\boldsymbol{x})$), in posterior probability ($p(\boldsymbol{x})$) through:

$$p(\boldsymbol{x}) \doteq \frac{1}{1 + \exp(ah(\boldsymbol{x}) + b)}, \tag{2.11}$$

where $a$ and $b$ are estimated by maximizing the log-likelihood of the training sample with a five-fold cross validation.

Considering all the characteristics of the reliability, we believe that using this measure can lead to an improvement of the performance with regard to the minority classes in a multiclass imbalanced problem. On this motivation we present in the following two reconstruction rule based on classification reliability.

The first one is a reconstruction rule in the One-Per-Class decomposition scheme. This rule, referred to as Reconstruction Rule by Selection (RRS), uses the useful information contained in the classification reliability to distinguish between safe and dangerous dichotomizer classifications, and then it applies different rules for each of these two cases.

The second rule that we propose is a statistical rules. Shirahishy et al. [122] proposed an effective statistical rule designed for OpC and PwC decomposition schemes which use the raw outputs of the binary classifiers. Inspired by their work, we extend their method to the ECOC decomposition scheme and, furthermore, we improve their proposal incorporating the use of reliability in the decision stage.

In addition on the development of the two reconstruction rule we propose also an efficient posterior estimation in case of boosting algorithms. Indeed we notice that several methods exist to compute the reliability or posterior probability, but small effort has been done in the case of boosting algorithm. This is due mainly to the fact that is widely believed that boosting and conditional class probability estimation are in conflict with each other. Indeed boosting iteratively improves classification at the price of progressively overfitting posteriors [19, 54]. We use the Universal Nearest Neighbours (UNN) [105], which is an algorithm that leverages nearest neighbors while minimizing a convex loss function. We demonstrate that a sub-class of surrogate losses exists, whose minimization brings simple and statistically efficient estimators for Bayes posteriors. We show also that the use of posteriors in the final decision improves system performance.

# 3. Materials and Methods

In this chapter we present information regarding the general experimental set-up used to validate our proposals. We briefly describe datasets used to test the two methods, the classification algorithms used as base classifiers in the decomposition framework and the metrics chosen to evaluate classification performances. We depute to specific sections in the next chapters the task to provide further details on the specific set-up used.

The chapter is organized as follow. In the first section we describe the datasets used, in the second section we give a short description of the classification algorithms employed and we finally describe the performance metrics adopted to evaluate classification outputs.

## 3.1. Datasets

We use 15 datasets in which 9 of them belong to medical domains. In order to validate the proposed methods for each one we chose a subset of these datasets providing an heterogeneous test bench. We report a short description of each one hereunder, and a summary in terms of number of samples, classes, features and class distributions can be found in table 3.1.

- *Breast Tissue (*BRTISS*)*: This dataset collects electrical impedance spectroscopy measurements performed on breast tissue samples. Each one of these samples belong to one out of 6 possible classes i.e. carcinoma, fibro-adenoma, mastopathy, glandular, con connective, adipose. Samples distribution among classes ranges from 20.8% to 13.2%.

- *Cells* (BIOCELLS)[106]: The images are acquired by means of a fully fluorescence microscope. In biological experiments different NIS proteins mutated are expressed for putative sites of phosphorylation. The effect on the protein localization of each mutation is studied after immunostaining using anti-NIS antibodies. Immunocytolocalization analysis on 489 cells revealed 2 cell types with different subcellular distributions of NIS.

- *Dermatology (*DERM*)*: This dataset is composed of 366 samples described by 33 features. The classification task is to classify each sample in 6 classes aiming at predict a differential diagnosis of erythemato-squamous diseases. Samples distribution range from 30.6% to 5.5%.

- *Ecoli (*ECOLI*)*: This dataset is composed by 336 samples. Each sample, described by 8 features, represents a localization site. Samples are distributed in 6 classes. As common practice we remove the classes with less than 10 samples. Distribution among classes ranges from 43.7% to 7.5%.

| Dataset | Number of samples | Number of classes | Number of features | Class distribution (%) Majority class | Minority class |
|---|---|---|---|---|---|
| BIOCELLS | 489 | 2 | 64 | 79.6% | 20.5% |
| BRTISS | 106 | 6 | 9 | 20.8% | 13.2% |
| DERM | 366 | 6 | 33 | 30.6% | 5.5% |
| ECOLI | 327 | 5 | 7 | 43.7% | 6.1% |
| FER | 876 | 6 | 50 | 28.1% | 7.5% |
| GLASS | 205 | 5 | 9 | 37.0% | 6.3% |
| ICPR$_{BOF}$ | 721 | 6 | 1024 | 28.9% | 8.0% |
| ICPR$_{BIF}$ | 721 | 6 | 1024 | 28.9% | 8.0% |
| IIFI | 600 | 3 | 57 | 36.0 % | 31.5% |
| ORHD | 5620 | 10 | 60 | 10.2% | 9.8% |
| SAT | 6425 | 6 | 36 | 23.9% | 9.7% |
| SEEDS | 210 | 7 | 3 | 33.0% | 33.0% |
| SUN10 | 1677 | 10 | 2048 | 14.4% | 6.9% |
| WFRN | 5456 | 4 | 24 | 40.4% | 6.0% |
| WINE | 178 | 3 | 13 | 39.9% | 27.0% |
| YEAST | 1479 | 9 | 8 | 31.3% | 1.6% |

Table 3.1.: Summary of datasets characteristics. For each dataset are shown the number of samples, the numbero of classes, the number of features, the number of majority class samples (%) and the number fo minority class samples (%)

- *Facial Expression Recognition (*FER*)*: This dataset is derived from Cohn-Kanade AU-Coded Facial Expression Database [79]. It is composed of videos showing an actor that performs 6 prototypical facial expressions i.e. anger, happiness, disgust, fear, sadness, surprise. These expressions correspond to the classes of the dataset. This dataset is composed by 876 instances, described by 50 features accordingly to [35]. A priori probabilities of classes range from 7.5% to 28.1%.

- *Glass (*GLASS*)*: This dataset is composed of 205 samples, described by 10 attributes [6]. The dataset is developed for glass type classification motivated by criminological investigation. As common practice we remove the two classes having less than ten samples. Remaining classes distribution ranges between 6.3% to 37.0%.

- *International Conference on Pattern Recognition HEp2 Cells* (ICPR): HEp2 images are acquired by means of a fluorescence microscope coupled with a 50W mercury vapor lamp. This dataset has 791 instances distributed over 6 classes. We generated two version of this dataset, ICPR$_{BOF}$ and ICPR$_{BIF}$ using two kind of descriptors: Bag of Features and BIF respectively.

- *Indirect Immunofluorescence intensity (*IIFI*)*: Connective tissue diseases are autoimmune disorders characterized by a chronic inflammatory process involving connective tissues [112]. Test based on HEp-2 substrate is usually performed, since it is the recommended method [127]. The dataset consists of 14 features extracted from 600 images of patient sera thorough the Indirect Immunofluorescence method [127]. The samples are distributed over 3 classes, namely positive (36.0%), negative (31.5%) and intermediate (32.5%) .

- *Optical Recognition of Handwritten Digits (*ORHD*)*: This dataset is composed by 5620 samples representing handwritten digits through 64 attributes [6]. Samples are divided in 10 classes where the a priori distributions range between 9.9% and 10.1%.
- *Statlog (Landsat Satellite) (SAT)*: The dataset consists of the multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood [6]. There are 6425 samples described by 36 features. The sample are distributed in 6 classes: red soil (23.9%), cotton crop (10.9%), grey soil (21.1%), damp grey soil (9.7%), soil with vegetation stubble (11.0%), very damp grey soil (23.4%).
- *Sun (*SUN10*)*: This dataset is a collection of annotated images covering a large variety of environmental scenes, places and the objects within [149]. We have extracted 1677 samples divided in 10 classes. Each samples is described by 2048 attributes generated applying the bag-of-features approach to SIFT descriptors. Class prior ranges between 14.4% and 6.9%.
- *Wall-Following Robot Navigation (*WFRN*)*: This is a dataset with 5456 samples represented by 24 features. The data were collected as the SCITOS G5 navigates through the room following the wall in a clockwise direction, for 4 rounds [6]. To navigate, the robot uses 24 ultrasound sensors arranged circularly around its ”waist”. The sample are distributed in 4 classes: move-forward (40.4%), slight-right-turn (15.2%) , sharp-right-turn (38.4%), slight-left-turn (6.0%).
- *Wine (*WINE*)*: This dataset is the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars [6]. The analysis determined the quantities of 13 constituents found in each of the three types of wines. There are 178 samples describe by 13 features. Samples are distributed in three classes, whose priors are 39.9%, 33.1% and 27.0%.
- *Yeast* (YEAST)[6]: This database contains information about 10 localization sites of Yeast cells. It is composed of 1484 instances represented by 8 features. We remove the endoplasmic reticulum lumen class that makes impossible perform ten-fold cross validation since it has only 5 samples.

## 3.2. Classifiers

As classification algorithms we used Support Vector Machine (SVM) as kernel machine, k-Nearest Neighbours as non-parametric algorithm, Multi Layer Perceptron as Neural Networks and Adaboost as Boosting approach. Classifiers' hyper-parameter values and optimization methods are reported in a specific paragraph in the description of each method. A brief description of these algorithms is reported in the following:

- *SVM* algorithm performs classification building hyperplane or set of hyperplanes in a high-dimensional space. In addition to performing linear classification, SVM can efficiently perform a non-linear classification using what is called the kernel trick i.e. implicitly mapping its inputs into high-dimensional feature spaces. A good separation is achieved by the hyperplane that has the largest distance to the nearest training datapoints of any class, since in general the larger the margin the lower the generalization error of the classifier. An important property of SVM is that the determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum.

- *k-nearest neighbor* (k-NN) algorithm is amongst the simplest of all machine learning algorithms and should be one of the first choices for a classification task when there is little or no prior knowledge about the distribution of the data. K-nearest neighbour classification was developed from the need to perform discriminant analysis when reliable parametric estimates of probability densities are unknown or difficult to determine. The k-nearest neighbours algorithm is a non-parametric method for classification and regression, that predicts objects' "values" or class memberships based on the k closest training examples in the feature space. An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common amongst its k nearest neighbours (k is a positive integer, typically small). If $k = 1$ then the object is simply assigned to the class of that single nearest neighbour. Usually Euclidean distance is used as the distance metric.

- *multilayer perceptron* (MLP) is a modification of the standard linear perceptron and can distinguish data that are not linearly separable. It consists of multiple layers of nodes in a directed graph, it is a feed-forward neural network whose processing nodes (neurons) compute the weighted average of its inputs and then transform the average by an activation function such as the hyperbolic tangent and logistic function. What makes a multilayer perceptron different from perceptron is that each neuron uses a nonlinear activation function which was developed to model the frequency of action potentials, or firing, of biological neurons in the brain. This function is modelled in several ways, but must always be normalizable and differentiable.

- *AdaBoost* (Adaptive Boosting) extends boosting to multi-class and regression problems. AdaBoost has many variations, such as AdaBoost.M1 for classification problems where each classifier can attain a weighted error of no more than $1/2$ , AdaBoost.M2 for those weak classifiers that cannot achieve this error maximum (particularly for problems with large number of classes, where achieving an error of less than 1/2 becomes increasingly difficult), among many others. We adopt the most popular of AdaBoost's variations, AdaBoost.M1 for multi-class problems. In AdaBoost.M1, bootstrap training data samples are drawn from a distribution D that is iteratively updated such that subsequent classifiers focus on increasingly difficult instances. This is done by adjusting D such

| | | **Hypotesized Class** | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\zeta_1$ | $\zeta_2$ | $\zeta$ | $\zeta_c$ | $\dots$ | $\zeta_K$ |
| **True Class** | $\omega_1$ | $n_{11}$ | $n_{12}$ | $\dots$ | $n_{1j}$ | $\dots$ | $n_{1K}$ |
| | $\omega_2$ | $n_{21}$ | $n_{22}$ | $\dots$ | $n_{2j}$ | $\dots$ | $n_{2K}$ |
| | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ |
| | $\omega_c$ | $n_{j1}$ | $n_{j2}$ | $\dots$ | $n_{jj}$ | $\dots$ | $n_{jK}$ |
| | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ |
| | $\omega_K$ | $n_{K1}$ | $n_{K2}$ | $\dots$ | $n_{Kj}$ | $\dots$ | $n_{KK}$ |

Table 3.2.: Confusion matrix of a $K$-classes classifier

that previously misclassified instances are more likely to appear in the next bootstrap sample. The classifiers are then combined through weighted majority voting.

## 3.3. Performance metrics

Traditionally, the most frequently used performance metrics are the accuracy ($acc$) and its counterpart, the error rate. Given $N$ samples distributed over $K$ classes, let $\{\zeta_1, \zeta_2, \dots, \zeta_c, \dots, \zeta_K\}$ be the predicted class labels. A representation of classification performance can be formulated by the confusion matrix, as illustrated in Table 3.2. The recognition accuracy is defined as

$$acc = \frac{\sum_{j=1}^{K} n_{jj}}{N} \tag{3.1}$$

where $n_{jj}$ is the number of elements of class $j$ correctly labelled.

In certain situations, measuring the performances using only accuracy can be deceiving since it fails to reflect the extent of minority class misclassifications. For example, consider the a-priori distribution of GLASS dataset, where the $6.3\%$ of samples are in the minority class, the $37.0\%$ of samples belong to the majority one, and the remaining $56.7\%$ of samples are in the other classes. One should develop a classification system that perfectly classifies every sample on classes except for the minority one, achieving an accuracy of $93.7\%$, that should appears satisfactory. That is to say, the accuracy in this case does not provide adequate information on a classifier's functionality with respect to the type of classification required. Indeed, the accuracy is a performace measure based on values from both rows of confusion matrix, whose values depend on class distribution. Any performance measure based on values from rows will be inherently sensitive to class skew, as accuracy is.

Hence, it would be more interesting to use a performance measure dissociating the hits (or the errors) that occur in each class. From Table 3.2, we compute the accuracy per class, which is defined as $acc_j = n_{jj}/N_j$, with $j = 1, \dots, K$. Since each $acc_j$ is estimated considering only one row of the confusion matrix, it is independent of prior probabilities. Furthermore, the combination of the accuracies per class provides an estimator of summarizing the performances of the classifier. It is the geometric mean of accuracies (GACC) given by:

$$g = \left( \prod_{j=1}^{K} acc_j \right)^{\frac{1}{K}} \tag{3.2}$$

It is worth nothing that $g$ is a non-linear measure. Indeed, a change in one of its arguments has a different effect depending on its magnitude; for instance, if a classifier misses the labels of all samples in the $j$th class, it results in $acc_j = 0$, and $g = 0$. Another measure used to evaluate classifiers performance is the F-measure. It is defined as F-$measure = 2((Recall)^{-1} \times (Precision)^{-1})^{-1}$. Where *Recall* is the fraction of samples labelled as belonging to the considered class that are correctly classified, whereas *Precision* is the fraction of samples in the considered class that are correctly classified. F-$measure$ shares with the GACC the property of not suffering from the same issues that affect the Accuracy. Indeed it is computed from independent rows of the confusion matrix.

Hence, these three metrics, Accuracy, geometric mean of accuracy per class and $F\ measure$ provide an overall analysis of the classification performance tacking in account also the performance with respect of each class.

# 4. Reconstruction Rule by Selection

Literature analysis (Chapter 2) reveals that existing proposals, addressing multiclass skewness in decomposition framework, work at level of single dichotomizer [3, 24, 40, 46, 72, 90, 126, 128, 135, 155], whereas, to the best of our knowledge, no attempt have been done to solve this issue at reconstruction rule level. On this motivation, we propose here a reconstruction rule for OpC decomposition approach which copes with skewness between classes. First, it distinguishes between safe and dangerous binary classifications using the classification reliabilities assigned by binary classifiers to the input sample, and then it sets the final multiclass label applying different reconstruction rules for each of these two cases. Hereinafter, the proposed rule is referred to as *Reconstruction Rule by Selection* (RRS). The decision to develop our proposal in the OpC framework arises from the fact that this decomposition scheme, even if it is often used to derive multiclass classifier by binary learning algorithms, has not received the same attention in literature as other rules. Some authors state that other schemes are preferable to OpC [50], nevertheless it has been proven that OpC performs as well as more complex error-correcting coding schemes when dicothomizers are well tuned [111]. Furthermore it is well know that OpC scheme produces imbalanced binary tasks and then, among all the decomposition schemes, it is the one that could benefit more of a rule suited for imbalanced domains.

We extensively compare this rule with other two well-established reconstruction criteria on a set of eight public and four artificial datasets, testing four classification architectures. The results show that the proposed reconstruction rule provides larger performances than those returned by the other criteria, reducing the effects of class skewness. The large number of experiments we carry out shows also that the employment of reliability in the reconstruction rule permits to achieve larger values of accuracy and geometric mean of accuracies than using only the crisp labels.

This chapter is organized as follows. We firstly describe the proposed method, secondly we provide details on the experimental set-up, finally we present and discuss results.

## 4.1. Method description

In order to present RRS method we introduce the following notation:

- $\Omega = \{\omega_1, \omega_2, \dots, \omega_K\}$ is the set of class labels;
- $N$ is the total number of samples;
- $N_j$ is the number of samples belonging to the class $j$;
- $x \in \Re^n$ is a sample;

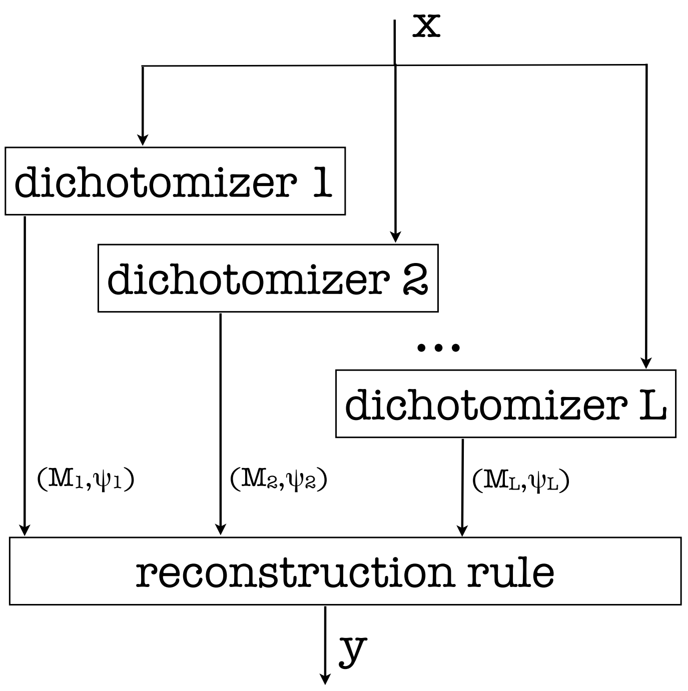

Figure 4.1.: System architecture of traditional One-per-class decomposition.

- the binary profile $\mathbf{M}(x)$ is the $K$-bit vector which collects dichotomizers' outputs on $x$ defined as: $\mathbf{M}(x) = [M_1(x), M_2(x), \ldots, M_K(x)]$. $M_j(x)$ is 1 if $x \in \omega_j$, 0 otherwise[1];
- the reliability profile $\mathbf{\Psi}(x)$ is a $K$ elements vector collecting classification reliability i.e confidence of a classifier on its output [51]. Formally, $\mathbf{\Psi}(x) = [\psi_1(x), \psi_2(x), \ldots, \psi_K(x)]$. Each entry $\psi_j(x)$ lies in $[0, 1]$ and represents the degree that $x$ belong or not to class predicted by the $j$th dichotomizer.
- the reverse a-priori probability profile $\mathbf{R}$ contains the knowledge on the a-priori classes distribution. It is a $K$ elements vector defined as $\mathbf{R} = [r_1, r_2, \ldots, r_K]$, where $r_j = 1 - N_j/N$;

Moreover, in the following for brevity a binary dichotomizer classification is referred to as *positive* if the sample is assigned to the dichotomizer own class, i.e. $M_j(x) = 1$, *negative* otherwise.

The basic idea of our approach is depicted in Figure 4.2, where the block named as profile analysis distinguishes between *safe* and *dangerous* classifications on the basis of measures derived from dichotomizers' soft labels. Intuitively, safe classifications are those where an analysis of binary $\mathbf{M}(x)$ and reliability $\mathbf{\Psi}(x)$ profiles suggest that all dichotomizers are strongly confident about their output. Conversely, dangerous classifications are classifications where the same profiles suggest that dichotomizers' output might be negatively affected by

[1]Hereinafter, instead of using {-1;1} labels for negative and positive outputs, we adopt the {0;1} notation to simplify the following formulas.

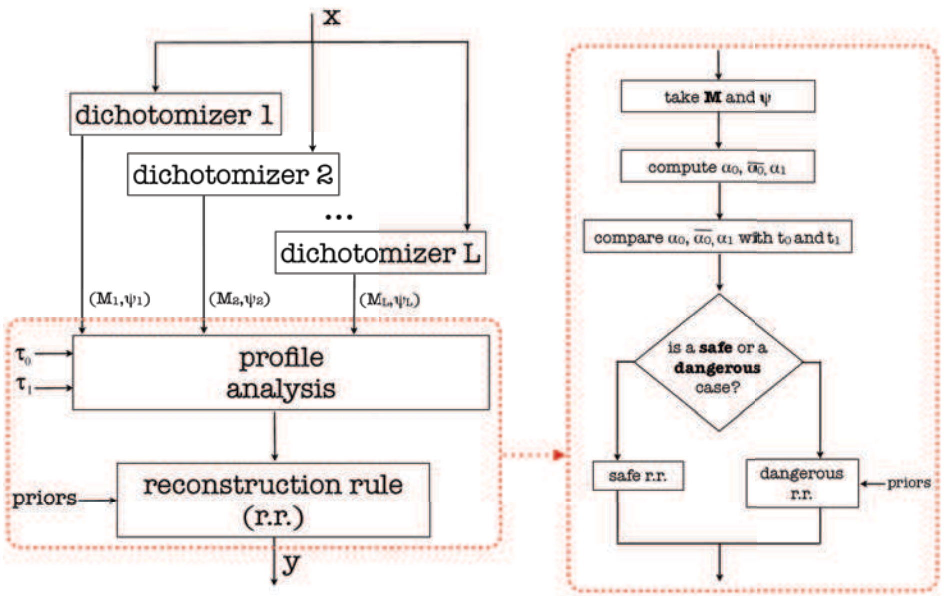

Figure 4.2.: RRS system architecture. Left side: decomposition of the polychotomy into several dichotomies, introducing also the profile analysis. Right side: it shows how RRS works, distinguishing between safe and dangerous classifications. The abbreviation r.r. stands for reconstruction rule.

the skewed nature of the dataset. The rule used to compose dichotomizers' outputs is different in the two cases and it uses the reverse a-priori probability profile **R** to set the final decision in case dichotomizers' classifications turn out to be dangerous. The introduction of this block is therefore the main difference with respect to the traditional OpC reconstruction approach represented in Figure 4.1.

Formally, let $\alpha_0(x) = min_j(\psi_j(x)|M_j(x) = 0)$ be the lowest reliability value among those provided by dichotomizers whose output is 0, i.e. the dichotomizers providing a negative classification, and let $\alpha_1(x) = max_j(\psi_j(x)|M_j(x) = 1)$ be the largest value of reliability among those provided by dichotomizers whose output is 1, i.e. the dichotomizers providing positive classifications. Let also $\overline{\alpha_0}(x) = \frac{\sum_i^K(\psi_j(x)|M_j(x)=0)}{\sum_i^K[M_j(x)=0]}$, with $[\cdot]$ indicator function, be the average value of reliabilities associated with dichotomizers whose output is 0. Furthermore, the minimum and the maximum conventionally evaluate to 0 if sets $(\psi_j(x)|M_j(x) = 0)$ and $(\psi_j(x)|M_j(x) = 0)$ are empty, respectively.

With these positions, the classifications provided by dichotomizers are considered dangerous if:

$$(\alpha_1(x) < \tau_1 \wedge \alpha_0(x) < \tau_0) \vee (\alpha_1(x) \geq \tau_1 \wedge \overline{\alpha_0}(x) < \tau_0) \qquad (4.1)$$

where $\tau_0$ and $\tau_1$ are thresholds in $[0, 1]$ estimated on a validation set maximizing the average accuracy per class. We will discuss the contributes of this two parameters at the and of this section. Condition 4.1 states that classifications are dangerous when: (i) both the highest reliability of classifiers providing positive classifications and the lowest reliability of classifiers providing negative classifications are below given thresholds or, alternatively, (ii) the highest reliability of classifiers providing positive classifications is higher than the corresponding threshold, but the average value of reliabilities of classifiers providing negative classifications is below a given threshold. Case (i) corresponds to when all positive classifications are scarcely reliable and there is at least one negative classification that is scarcely reliable too (meaning this classification might have been positive instead), whereas case (ii) corresponds to when many negative classifications are scarcely reliable although there is at least one positive classification sufficiently reliable.

To make more clear the rationale of condition 4.1, consider its negate, that can be rewritten as:

$$(\alpha_1(x) < \tau_1 \wedge \alpha_0(x) \geq \tau_0) \vee (\alpha_1(x) \geq \tau_1 \wedge \overline{\alpha_0}(x) \geq \tau_0) \tag{4.2}$$

Condition 4.2 states that classifications can be retained safe when either all classifiers providing negative classifications are sufficiently confident although positive classifications have low reliabilities, or there is at least one positive classification that is sufficiently reliable and many negative classification are sufficiently reliable too.

In the following we refer to quantities $\alpha_0(x)$, $\alpha_1(x)$, and $\overline{\alpha_0(x)}$ omitting the dependence on $x$ if this does not introduce ambiguity.

To set the final classification, RRS applies different criteria for safe and dangerous classifications.

In the former case, let be $\overline{M_j(x)}$ the negate of the $j$th dichotomizer output and $m = \sum_{j=1}^{K} M_j(x)$ the number of dichotomizers providing a positive classification, i.e. recognizing $x$ as belonging to their own class. The index $s$ of the dichotomizer setting the final class $\omega_s \in \Omega$ is given by:

$$s = \begin{cases} argmax_j(M_j(x) \cdot \psi_j(x)) & \text{if} \quad m \in [1, K] \\ \\ argmim_j(\overline{M_j(x)} \cdot \psi_j(x)) & \text{if} \quad m = 0 \end{cases} \tag{4.3}$$

Since we are in a safe case, now the final decision depends on both $\mathbf{M}(x)$ and $\mathbf{\Psi}(x)$, without considering data related to the degree of imbalance presented in the dataset.

When a dangerous classification occurs, it should be interpreted as an error due to class skew. In this case, we cannot rely any more on dichotomizers decision only, but we should also take into account somehow the a-priori class distribution. Indeed, we have to decide if the sample should be assigned either to the class *recognized* with the highest reliability (all other positive classifications, if any, are less reliable) or to the class *not recognized* with the lowest reliability (all other negative classifications are more reliable). In this respect there are two alternatives: either (i) relying on purely bayesian classification, or (ii) deciding in favor of the minority class. Note, however, that the latter makes sense only if the reliability of the positive classification is high, since this could indicate that class unbalance may have led to a wrong decision (remember we are considering the case when there are chances the classification is wrong). We therefore consider the quantities $\alpha_0$ and $\alpha_1$ to discriminate between these two possibilities.

When $\alpha_0 > \alpha_1$, i.e. when the reliability of the most reliable positive classification is low, we choose to rely on purely bayesian classification and the sample is assigned to the class with the lower reverse probability, i.e. the class most populated in the training set. Indeed, in this case reliabilities of both classifiers are very low and it is unlikely that a classification error has been caused by class imbalance. Conversely, when $\alpha_0 \leq \alpha_1$, i.e. when the reliability of the most reliable positive classification is high, we assign the sample to the class with highest reverse a-priori class distribution. Note that this means that the most reliable positive classification is confirmed if and only if the corresponding a-priori class distribution is lesser that the one associated to the least reliable negative classification.

More formally, the rule to be applied in case of dangerous classification is the following. Let $j_0$ and $j_1$ be the indices of the most reliable positive classification and of the least reliable negative classification, respectively. The index $s$ of the dichotomizer setting the final class $\omega_s \in \Omega$ is given by:

$$s = \begin{cases} argmin_{j \in \{j_0, j_1\}} r_j & \text{if} \quad \alpha_0 > \alpha_1 \\ \\ argmax_{j \in \{j_0, j_1\}} r_j & \text{if} \quad \alpha_0 \leq \alpha_1 \end{cases} \tag{4.4}$$

### 4.1.1. About $\tau_0$ and $\tau_1$

We try here to give a deeper insight of $\tau_0$ and $\tau_1$ parameters role in the proposed rule. $\{\tau_0, \tau_1\} \in \Re$ values ranges in [0,1]. They are chosen maximizing average accuracies per class on a validation sets. The search of the optimal value has been exhaustively performed using stepwise construction of a grid with step equal to 0.05. Graphical samples of grid search results are reported in figure 4.3 where average accuracies per class values are represented as function of $\tau_0$ and $\tau_1$ on the considered validation set. The corner $[0, 0]$ corresponds to not applying any distinction between safe and dangerous classification. Observing the shapes in the figure, we notice the importance of the tuning of the two parameters. Indeed variation of these values improve or drop significantly the classification performance. We chose to optimize this value on average accuracy per class since this lead to a better generalization ability with respect of the minority classes.

## 4.2. Experimental set-up

In this section we present our experimental set-up, providing the list of used datasets and details on classification paradigms employed as well as the list of performance metrics.

### 4.2.1. Datasets

We used twelve datasets: eight are a collection of real public datasets and four are artificial datasets. The real datasets that we have chosen in order to provide an heterogeneous test-bench to validate our proposal are: FER, GLASS, IIFI, ORHD, SAT, SUN10, WFRN and WINE. Description of real datasets is reported in section 3.1 and datasets details can be found in table 3.1 whereas artificial datasets description is reported in the following.

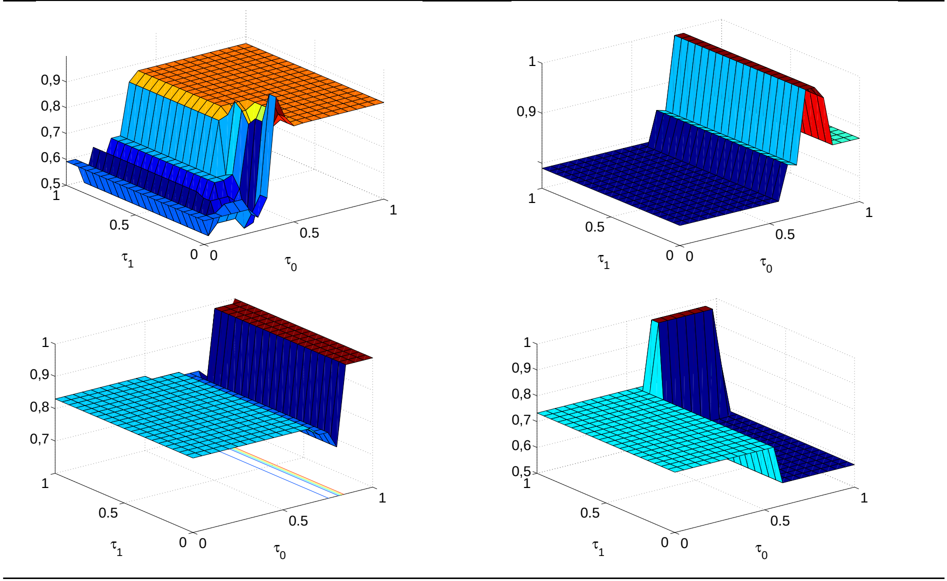

Figure 4.3.: Examples of performance on validation set, in term of average accuracies per class, obtained varying $\tau_0$ and $\tau_1$ parameters. From left to right and from top to bottom: kNN, SVM, MLP and ADA.

Inspired by [88], in Figure 4.4 we graphically represent the degree of imbalance for each dataset. The chart represents the prior probability of the minority class as a function of prior probability of the majority class. The feasible space is below the diagonal line of the plot, which also corresponds to equiprobable classes. This line can be therefore thought of as the edge of balanced problems. The balance disappears towards the bottom right corner. Point (0.5,0.5) corresponds to two equiprobable classes. This graphical representation helps us to observe how much the datasets are heterogeneous with respect to the imbalance ratio. For instance, in the figure we notice that ORHD is a quite balanced dataset, whereas GLASS and WFRN have a strong degree of imbalance.

## Artificial datasets

We generate simulated examples involving fairly complex decision boundaries. To this aim, synthetic samples are represented by a feature vector composed of 15 elements randomly drawn from a 15-dimensional normal distribution $x \sim N(\mu_c, \sigma * I)$. Mean value of each normal distribution $\mu_c \in \{\mu_1, \mu_2, \ldots, \mu_K\}$ is randomly taken in the range $[0, 1]$, while $\sigma$ is equal to 1.5 for all the distributions. We generate four artificial sets with different number of classes, i.e. $K = \{5, 10, 15, 20\}$, which are referred to as SIM1, SIM2, SIM3 and SIM4 respectively. In each dataset the smallest class has ten samples, whereas the largest class has one thousand samples, providing a ratio between the two classes always equal to $\frac{1}{100}$. The number of samples belonging to other classes is computed as follows:

$$N_c = \frac{2 \cdot 1000}{K - (c - 2)} \qquad j \in \{2, 3, \ldots, K - 1\}. \tag{4.5}$$

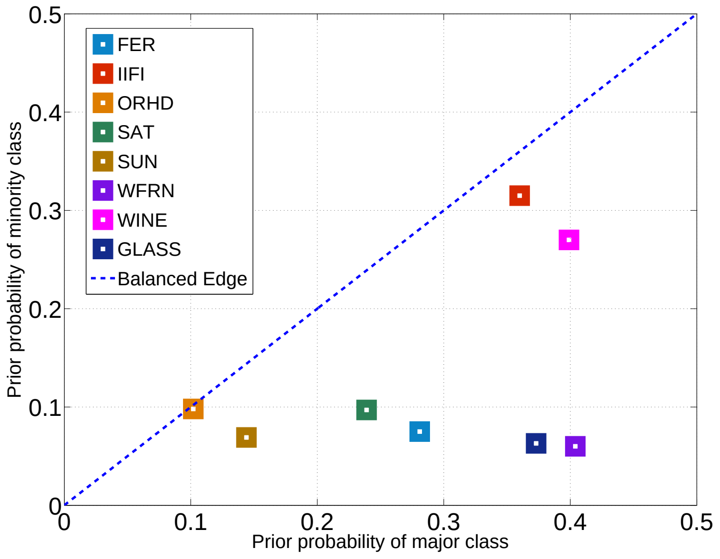

Figure 4.4.: Dataset distribution as function of prior distribution of majority (x-axis) and minority (y-axis) class.

For instance consider $K = 5$. Each class has a number of samples equal to 1000, 666, 500, 400, 333, 10; providing ratios between the smallest class and the others which are equal to 0.010, 0.015 ,0.020, 0.025, 0.030 and 1.

### 4.2.2. Classifiers

We employ a k-Nearest Neighbour (kNN) as a statistical machine, an Support Vector Machine (SVM) as a kernel machine, an Adaboost (ADA) as a weak learning algorithms, and a Multi-Layer Perceptron (MLP) as a neural network. Brief descriptions of these algorithms are reported in 3.2. In this subsection we describe how we tune the free parameters of the classifiers and how we estimate the classification reliablities.

**kNN**. The kNN require no specific set-up. We test values of $k$ equal to $\{1, 3, 5, 7\}$ and choose the value providing the best performances on a validation set according to a five-fold cross validation. We estimate the reliability of each classification act on the basis of information directly derived from the output of the expert and analysing also the reasons in the feature space giving rise to unreliable classification. For further details the interested reader may refer to [27].

**SVM**. We test a SVM with a gaussian radial basis kernel. Values of regularization parameter $C$ and scaling factor $\sigma$ are selected within $[1, 10^4]$ and $[10^{-4}, 10]$, adopting a $\log$ scale to sample the two intervals. The value of each parameter is tuned using a five fold cross-validation on a validation set. The reliability of a SVM classification is estimated as proposed in [107], where the decision value of the classifier is transformed in a posterior probability.

**MLP**. We use a MLP with a number of hidden layers equal to half of the sum of features number plus class number. The number of neurons in the input layer is given by the number

| Datasets | Metrics | kNN | SVM | MLP | ADA |
|---|---|---|---|---|---|
| FER | HMD | 73.78 ± 1.92 | 94.10 ± 1.19 | 86.22 ± 1.00 | 39.57 ± 1.34 |
| | LBD | 79.69 ± 1.41 | 94.33 ± 1.33 | 88.27 ± 0.75 | 40.40 ± 1.07 |
| | RRS | 81.13 ± 1.38 | 96.83 ± 0.78 | 79.58 ± 6.56 | 46.97 ± 3.10 |
| GLASS | HMD | 68.36 ± 2.68 | 65.29 ± 2.58 | 68.28 ± 2.15 | 62.47 ± 2.96 |
| | LBD | 70.55 ± 2.70 | 54.46 ± 2.00 | 57.99 ± 2.12 | 56.50 ± 2.81 |
| | RRS | 70.55 ± 2.70 | 66.97 ± 3.49 | 63.27 ± 2.88 | 69.49 ± 2.40 |
| IIFI | HMD | 65.49 ± 2.19 | 66.18 ± 1.40 | 66.51 ± 1.78 | 63.85 ± 2.45 |
| | LBD | 63.75 ± 2.38 | 64.59 ± 1.93 | 65.45 ± 1.24 | 64.68 ± 2.68 |
| | RRS | 68.93 ± 2.05 | 72.17 ± 1.43 | 68.16 ± 1.24 | 58.62 ± 4.13 |
| ORHD | HMD | 97.43 ± 0.21 | 97.53 ± 0.16 | 96.94 ± 0.14 | 76.76 ± 0.89 |
| | LBD | 97.87 ± 0.22 | 98.65 ± 0.16 | 98.45 ± 0.17 | 87.97 ± 0.37 |
| | RRS | 97.87 ± 0.22 | 98.65 ± 0.16 | 98.45 ± 0.17 | 87.97 ± 0.37 |
| SAT | HMD | 86.81 ± 0.54 | 91.40 ± 0.33 | 86.79 ± 0.38 | 87.78 ± 0.20 |
| | LBD | 86.61 ± 0.51 | 90.31 ± 0.43 | 86.66 ± 0.59 | 93.93 ± 0.24 |
| | RRS | 90.64 ± 0.42 | 91.92 ± 0.35 | 90.68 ± 0.25 | 72.20 ± 4.17 |
| SUN10 | HMD | 51.92 ± 0.94 | 65.78 ± 1.05 | 64.28 ± 0.95 | 47.46 ± 1.00 |
| | LBD | 57.25 ± 0.77 | 74.96 ± 1.43 | 72.21 ± 1.05 | 58.78 ± 1.30 |
| | RRS | 57.43 ± 0.87 | 74.76 ± 1.39 | 72.21 ± 1.05 | 58.72 ± 1.32 |
| WFRN | HMD | 89.56 ± 0.45 | 89.93 ± 0.38 | 87.88 ± 0.32 | 71.33 ± 0.48 |
| | LBD | 90.64 ± 0.42 | 90.77 ± 0.36 | 88.52 ± 0.38 | 76.97 ± 0.57 |
| | RRS | 86.99 ± 0.50 | 91.84 ± 0.35 | 88.60 ± 0.27 | 95.38 ± 0.19 |
| WINE | HMD | 95.96 ± 1.65 | 96.51 ± 1.77 | 96.59 ± 1.73 | 94.95 ± 1.51 |
| | LBD | 95.11 ± 1.66 | 96.65 ± 1.22 | 97.15 ± 0.92 | 94.92 ± 2.07 |
| | RRS | 95.96 ± 1.65 | 97.72 ± 1.51 | 97.74 ± 0.90 | 84.10 ± 8.14 |

Table 4.1.: Average values of the global accuracy (ACC) on real datasets when kNN, SVM, MLP and ADA are used as base classifier.

of the features whereas the number of neurons in the output layer is two. The reliability is a function of the values provided by neurons in the output layer [27].

**ADA**: We use the "Adaboost M1" algorithm proposed in [53], where weak learners are decision stumps. The number of iteration is equal to 100. The reliabilities of ADA classifications are estimated using the magnitude of the final hypothesis [115].

### 4.2.3. Performance metrics

Performance of the propose method and competitors are evaluated in term of accuracy ($acc$) , the geometric mean of accuracies (GACC) and $F\,measure$. For further details on these metrics see Section 3.3.

| | | Classifiers | | | |
|---|---|---|---|---|---|
| Datasets | Metrics | kNN | SVM | MLP | ADA |
| FER | HMD | 61.37 ± 7.03 | 91.71 ± 1.93 | 81.38 ± 1.60 | 0.00 ± 0.00 |
| | LBD | 72.22 ± 5.71 | 92.84 ± 1.87 | 85.09 ± 1.27 | 4.93 ± 3.21 |
| | RRS | 76.95 ± 2.14 | 95.94 ± 1.08 | 74.30 ± 8.24 | 2.49 ± 2.42 |
| GLASS | HMD | 6.67 ± 6.49 | 0.00 ± 0.00 | 0.00 ± 0.00 | 0.00 ± 0.00 |
| | LBD | 11.16 ± 8.75 | 6.07 ± 5.90 | 0.00 ± 0.00 | 0.00 ± 0.00 |
| | RRS | 11.16 ± 8.75 | 0.00 ± 0.00 | 6.96 ± 6.76 | 8.18 ± 7.95 |
| IIFI | HMD | 62.53 ± 2.45 | 60.29 ± 1.97 | 62.40 ± 2.09 | 59.13 ± 2.91 |
| | LBD | 62.26 ± 2.52 | 61.46 ± 2.13 | 63.18 ± 1.70 | 61.37 ± 2.91 |
| | RRS | 67.94 ± 2.17 | 70.01 ± 1.70 | 65.77 ± 1.49 | 57.70 ± 4.21 |
| ORHD | HMD | 97.39 ± 0.21 | 97.49 ± 0.16 | 96.90 ± 0.14 | 46.52 ± 12.31 |
| | LBD | 97.85 ± 0.23 | 98.64 ± 0.16 | 98.44 ± 0.17 | 87.62 ± 0.42 |
| | RRS | 97.85 ± 0.23 | 98.64 ± 0.16 | 98.44 ± 0.77 | 87.62 ± 0.42 |
| SAT | HMD | 86.48 ± 0.57 | 85.97 ± 0.68 | 82.65 ± 0.42 | 0.00 ± 0.00 |
| | LBD | 88.06 ± 0.47 | 87.66 ± 0.64 | 85.01 ± 0.54 | 12.46 ± 8.07 |
| | RRS | 88.06 ± 0.47 | 89.13 ± 0.56 | 87.71 ± 0.34 | 59.02 ± 3.60 |
| SUN10 | HMD | 26.43 ± 5.7 | 56.21 ± 5.34 | 58.74 ± 1.50 | 17.94 ± 5.83 |
| | LBD | 43.06 ± 4.81 | 72.59 ± 1.64 | 70.70 ± 1.22 | 54.46 ± 1.93 |
| | RRS | 43.13 ± 4.86 | 72.47 ± 1.60 | 70.70 ± 1.22 | 54.42 ± 1.93 |
| WFRN | HMD | 85.87 ± 0.88 | 89.05 ± 0.47 | 83.59 ± 0.82 | 0.00 ± 0.00 |
| | LBD | 86.01 ± 0.81 | 88.91 ± 0.53 | 85.96 ± 0.81 | 90.64 ± 0.74 |
| | RRS | 86.59 ± 0.72 | 90.26 ± 0.54 | 88.10 ± 0.52 | 92.95 ± 0.63 |
| WINE | HMD | 96.40 ± 1.48 | 96.40 ± 1.89 | 96.41 ± 1.99 | 93.98 ± 1.82 |
| | LBD | 95.62 ± 1.51 | 96.71 ± 1.21 | 97.15 ± 1.01 | 94.79 ± 2.12 |
| | RRS | 96.40 ± 1.48 | 97.76 ± 1.81 | 97.65 ± 1.01 | 80.46 ± 10.62 |

Table 4.2.: Average values of the geometric mean of accuracies (GACC) on real datasets when kNN, SVM, MLP and ADA are used as base classifier.

## 4.3. Results and Discussion

Experimental tests have been performed using three reconstruction rules, four classification algorithms, thirteen datasets, and running four times the 10-fold cross validation. This produced more than 5000 experiments whose results are summarized and discussed in the following subsections, where we distinguish between those achieved on real and artificial datasets.

### 4.3.1. Experiments on real datasets.

The three reconstruction rules (RRS, LBD and HMD) have been tested over eight real datasets running four times the 10-folds cross validation.

Tables 4.1, 4.2 and 4.3 report the average results in terms of ACC, GACC and $F\,measure$, respectively. Each tabular shows also the 95% confidence interval estimated with the t-student test.

To facilitate the comparisons between the performance of the reconstruction rules, in tables

| Datasets | Metrics | kNN | SVM | MLP | ADA |
|---|---|---|---|---|---|
| FER | HMD | 69 .75 ± 2 .32 | 93 .80 ± 1 .34 | 85 .64 ± 1 .23 | 25 .37 ± 1 .34 |
| | LBD | 76 .02 ± 2 .00 | 93 .24 ± 1 .54 | 85 .82 ± 1 .02 | 29 .26 ± 1 .09 |
| | RRS | 77 .98 ± 1 .80 | 96 .49 ± 0 .89 | 77 .10 ± 6 .58 | 35 .80 ± 2 .43 |
| GLASS | HMD | 55 .44 ± 3 .47 | 50 .40 ± 2 .87 | 53 .81 ± 2 .77 | 46 .09 ± 3 .82 |
| | LBD | 59 .17 ± 3 .86 | 43 .09 ± 2 .27 | 46 .15 ± 2 .28 | 40 .59 ± 2 .83 |
| | RRS | 59 .36 ± 3 .81 | 52 .59 ± 3 .87 | 52 .82 ± 3 .76 | 55 .86 ± 4 .40 |
| IIFI | HMD | 64 .31 ± 2 .28 | 63 .72 ± 1 .68 | 64 .73 ± 1 .85 | 61 .93 ± 2 .70 |
| | LBD | 63 .12 ± 2 .43 | 63 .16 ± 2 .01 | 64 .52 ± 1 .41 | 63 .24 ± 2 .79 |
| | RRS | 68 .54 ± 2 .10 | 71 .16 ± 1 .56 | 67 .22 ± 1 .31 | 58 .25 ± 4 .17 |
| ORHD | HMD | 97 .45 ± 0 .20 | 97 .58 ± 0 .15 | 97 .00 ± 0 .14 | 77 .37 ± 1 .29 |
| | LBD | 97 .88 ± 0 .22 | 98 .65 ± 0 .16 | 98 .45 ± 0 .17 | 87 .9 ± 0 .39 |
| | RRS | 97 .88 ± 0 .22 | 98 .65 ± 0 .16 | 98 .45 ± 0 .17 | 87 .9 ± 0 .39 |
| SAT | HMD | 88 .00 ± 0 .49 | 88 .47 ± 0 .49 | 85 .93 ± 0 .33 | 55 .11 ± 0 .29 |
| | LBD | 88 .97 ± 0 .45 | 88 .71 ± 0 .52 | 86 .39 ± 0 .46 | 66 .64 ± 0 .92 |
| | RRS | 88 .97 ± 0 .45 | 90 .33 ± 0 .43 | 88 .85 ± 0 .31 | 66 .45 ± 4 .12 |
| SUN10 | HMD | 50 .21 ± 0 .99 | 67 .45 ± 1 .10 | 64 .73 ± 1 .85 | 45 .81 ± 0 .92 |
| | LBD | 55 .16 ± 0 .92 | 74 .77 ± 1 .37 | 72.00 ± 1 .10 | 57 .36 ± 1 .58 |
| | RRS | 55 .23 ± 1 .03 | 74 .61 ± 1 .31 | 64 .73 ± 1 .85 | 57 .30 ± 1 .59 |
| WFRN | HMD | 86 .48 ± 0 .73 | 90 .62 ± 0 .39 | 86 .72 ± 0 .54 | 67 .52 ± 0 .21 |
| | LBD | 85 .56 ± 0 .78 | 87 .03 ± 0 .61 | 83 .57 ± 0 .76 | 88 .90 ± 0 .49 |
| | RRS | 86 .59 ± 0 .67 | 91 .13 ± 0 .46 | 87 .80 ± 0 .48 | 94 .35 ± 0 .38 |
| WINE | HMD | 95 .93 ± 1 .63 | 96 .59 ± 1 .76 | 96 .52 ± 1 .72 | 94 .87 ± 1 .53 |
| | LBD | 95 .06 ± 1 .66 | 96 .66 ± 1 .25 | 96 .90 ± 1 .01 | 94 .88 ± 2 .10 |
| | RRS | 95 .93 ± 1 .63 | 97 .72 ± 1 .65 | 97 .60 ± 0 .96 | 83 .38 ± 8 .64 |

Table 4.3.: Average values of the geometric mean of accuracies ($F\,measure$) on real datasets when kNN, SVM, MLP and ADA are used as base classifier.

4.4, 4.5,4.6, 4.7, 4.8,4.9, 4.10 , 4.11 and 4.12 we summarize the results over all folds according to a win/tie/loss scheme. Tables 4.4, 4.7 and 4.10 report results with respect to ACC, tables 4.5 , 4.8 and 4.11 report results with respect to GACC and tables 4.6 , 4.9 and 4.12 report results with respect to $F\,measure$. The win/tie/loss scheme works as follows. Given two methods A and B to be compared, we assign a point to win/tie/loss class every time method A achieves a larger/equal/lower performance than method B on a fold. Each tabular shows the number of win/tie/loss in a relative fashion, since they values have been divided by 40, i.e. the total number of stratified cross validation folds. For instance the value 25/25/50 means that method A against B wins 10 tests (25%), ties 10 tests (25%), and losses 20 tests (50%). Furthermore, the tabulars report in round parentheses a 1 if the performances computed over the 40 folds are statistically different according to t-test, with a significance level of $0.05$. Otherwise in the round parenthesis there is zero.

In the following, we report the results by pairwise comparing the three reconstruction rules. Each comparison is organized in three paragraphs, presenting the results in terms of accuracy, geometric mean and win/tie/loss. In the last paragraph we report the average performance of each classifiers over the datasets.

| Dataset | Classifiers | | | | Average |
|---|---|---|---|---|---|
| | kNN | SVM | MLP | ADA | |
| FER | 100/0/0 (1) | 35/45/20 (0) | 90/0/10 (1) | 50/10/40 (0) | 68.8/13.7/17.5 |
| GLASS | 50/40/10 (0) | 0/7.5/92.5 (1) | 0/10/90 (1) | 0/40/60 (1) | 12.5/24.4/63.2 |
| IIFI | 15/5/80 (0) | 30/17.5/52.5 (0) | 30/0/70 (0) | 52.5/20/27.5 (0) | 31.9/10.6/57.5 |
| ORHD | 85/10/5 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 46.3/2.5/1.2 |
| SAT | 95/5/0 (1) | 100/0/0 (1) | 80/10/10 (1) | 100/0/0 (1) | 93.8/3.7/2.5 |
| SUN10 | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 |
| WFRN | 25/20/55 (0) | 7.5/0/92.5 (1) | 30/0/70 (0) | 100/0/0 (1) | 40.6/5.0/54.4 |
| WINE | 0/85/15 (0) | 10/80/10 (0) | 20/60/20 (0) | 30/50/20 (0) | 15.0/68.7/16.3 |
| Average | 58.8/20.6/20.6 | 47.8/18.8/33.4 | 56.3/10/33.7 | 66.6/15/18.4 | - |

Table 4.4.: Exhaustive comparison between the performances of different classifiers expressed in terms of global accuracy (ACC), considering LBD and HMD reconstruction rules. Each tabular shows the amount of win/tie/loss of LBD comparing versus HMD. Round parentheses reports one if the performances computed over the 40 folds are statistically different according to t-test, with a significance level of $0.05$. Otherwise in the round parenthesis there is zero. Last column shows the average values of win/tie/loss achieved by different classifiers on a given dataset, whereas last row shows the average values achieved by a given classifier using different datasets.

### LBD Vs HMD

We compare now the two most used reconstruction rules in the OpC decomposition, i.e. HMD and LBD. As shown in formulas 2.3 and 2.6, recall that HMD predicts the final labels using the crisp labels only, whereas LBD applies a loss measure on the soft labels provided by each dichotomizer. In particular, for LBD we have always used an exponential loss function, as suggested in [3].

**Accuracy** In table 4.1 we observe that LBD outperforms HMD in 58% of cases, independently of binary learners and datasets used. Furthermore LBD outperforms HMD whatever the dichotomizer in FER, ORHD, SUN10 and WFRN datasets. Looking this table by columns we observe that LBD achieves larger results in 75% of cases using the ADA classifier, in 63% of cases using the SVM and MLP, and in 50% of cases using the kNN.

**Geometric mean** The comparison between LBD and HMD provides similar observations to those reported above for the accuracy. Table 4.2 shows that, independently of datasets and classifiers, LBD outperforms HMD method in the 87% of tests. In particular, LBD provides larger performance than those achieved by HMD using all the dichotomizers over FER, ORHD, SAT and SUN10 datasets. Furthermore, LBD show larger results than HMD in the 87% of cases using the SVM, MLP and ADA classifiers and in 75% of cases using kNN.

**F-measure** Performing the comparisons between LBD and HMD, we observe in Table 4.3 that, independently of datasets and classifiers, LBD outperforms HMD method in the 52% of

| Dataset | Classifiers | | | | Average |
|---|---|---|---|---|---|
| | kNN | SVM | MLP | ADA | |
| FER | 95/5/0 (1) | 55/25/20 (0) | 100/0/0 (1) | 20/80/0 (1) | 67.5/27.5/5 |
| GLASS | 5/90/5 (0) | 10/90/0 (1) | 0/100/0 (0) | 0/100/0 (0) | 3.8/95.0/1.2 |
| IIFI | 35/0/65 (0) | 72.5/0/27.5 (0) | 40/0/60 (0) | 82.5/0/17.5 (0) | 57.5/0.0/42.5 |
| ORHD | 90/5/5 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 97.5/1.2/1.3 |
| SAT | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 20/80/0 (1) | 80.0/20.0/0.0 |
| SUN10 | 90/10/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 97.5/2.5/0.0 |
| WFRN | 40/15/45 (0) | 40/0/60 (0) | 100/0/0 (1) | 100/0/0 (1) | 70.1/3.7/26.2 |
| WINE | 0/85/15 (0) | 10/80/10 (0) | 20/60/20 (0) | 30/50/20 (0) | 15.0/68.7/16.3 |
| Average | 58.9/26.3/ 16.8 | 60.9/24.4/14.7 | 70/20/10 | 56.6/38.7/ 4.7 | - |

Table 4.5.: Exhaustive comparison between the performances of different classifiers expressed in terms of geometric mean of accuracies (GACC), considering LBD and HMD reconstruction rules. Each tabular shows the amount of win/tie/loss of LBD comparing versus HMD. Round parentheses reports one if the performances computed over the 40 folds are statistically different according to t-test, with a significance level of $0.05$. Otherwise in the round parenthesis there is zero. Last column shows the average values of win/tie/loss achieved by different classifiers on a given dataset, whereas last row shows the average values achieved by a given classifier using different datasets.

tests. In particular, LBD provides larger performance than those achieved by HMD using all the dichotomizers over ORHD, SAT and SUN10 datasets. Furthermore, LBD show larger results than HMD in the 87% of cases using the ADA classifier and in 63% of cases using kNN and MLP.

**Win/Tie/Loss** Last column of Table 4.4 averages out over the binary learners win/tie/loss results measured in terms of $acc$. Its values show that LBD outperforms HMD in 50% of cases with a large difference between the number of wins. Indeed, the differences range from 51.2% (FER dataset) up to 100% (SUN10 dataset). In the opposite situation, i.e. when HMD wins, the differences with LBD are smaller and range from 1.2% (WINE dataset) up to 50.6% (GLASS dataset). Last row of the same table, which averages out the results for each dichotomizer over the eight datasets, shows that LBD always outperforms HMD.

Similar considerations hold for win/tie/loss results in case of GACC (Table 4.5). Last column of this table shows that, independently of the classifier used, LBD outperforms HMD in all cases. Similarly, the last row shows that LBD outperforms HMD independently of the datasets.

Finally in case of $F measure$ (Table 4.6) we observe in the last column that LBD show larger results than HMD, independently of the classifier used, in the 50% of the cases. In the last row LBD show larger results than HMD in the 50% of the cases independently of the dataset used. We point out that, when the number of wins of LBD is larger than those obtained from hamming, the difference between these two values is larger than the opposite case, i.e. when the number of wins of HMD is larger than those achieved by LBD.

| Dataset | Classifiers | | | | Average |
|---|---|---|---|---|---|
| | kNN | SVM | MLP | ADA | |
| FER | 100/ 0/0( 1) | 32.5/0/67.5 (0) | 50/0 /50( 0) | 80/0 /20 (1) | 66/0/34 |
| GLASS | 50/35/15 (0) | 7.5/0/92.5 (1) | 0/0/ 100 (1) | 10/0/90 (1) | 17/9/74 |
| IIFI | 20/0/80 (0) | 47.5/0/52.5 (0) | 30/0/70 (0) | 62.5/0/37.5 (0) | 40/0/60 |
| ORHD | 85/0/15 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 96/0/4 |
| SAT | 95/0/5 (1) | 60/0/40 (0) | 70/0/30 (0) | 100/0/0 (1) | 81/0/19 |
| SUN10 | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 |
| WFRN | 0/0/100 (0) | 0/0/100 (1) | 0/0/100 (1) | 100/0/0 (1) | 25/0/75 |
| WINE | 0/85/15 (0) | 10/75/15 (0) | 20/60/20 (0) | 30/30/40 (0) | 15/63/23 |
| Average | 56/15/29 | 45/9/46 | 46/8/46 | 73/4/23 | |

Table 4.6.: Exhaustive comparison between the performances of different classifiers expressed in terms of $F\,measure$, considering LBD and HMD reconstruction rules. Each tabular shows the amount of win/tie/loss of LBD comparing versus HMD. Round parentheses reports one if the performances computed over the 40 folds are statistically different according to t-test, with a significance level of $0.05$. Otherwise in the round parenthesis there is zero. Last column shows the average values of win/tie/loss achieved by different classifiers on a given dataset, whereas last row shows the average values achieved by a given classifier using different datasets.

### RRS Vs HMD

We compare now RRS results with those achieved by HMD. As in the previous comparison, we first present the results in terms of accuracy (Table 4.1), second we introduce the results in terms of geometric mean of accuracies (Table 4.2) and, third, we compare RRS and HMD according to the win/tie/loss scheme (Table 4.7 and 4.8 ).

**Accuracy** Table 4.1 shows that RRS outperforms HMD in the 72% of tabulars. In case of ORHD and SUN10 datasets RRS outperforms HMD for all binary learners.

Furthermore, we notice tha using SVM and MLP dichotomizers, RRS outperforms HMD reconstruction rule in seven 7 out of eight datasets.

**Geometric mean** Similar observations hold looking Table 4.2 where RRS outperforms HMD in 84% of tabulars. It is worth observing that on three datasets, namely ORHD, SAT, SUN10 and WFRN, RRS achieves larger performance than HMD independently of used dichotomizers. Furthermore we note RRS outperforms HMD in all datasets when it uses kNN and SVM binary learners.

**F-measure** Focusing on the results in term of $F\,measure$ in Table 4.3 we note that RRS outperforms HMD in the 65% of the tabulars. Results on ORHD, SAT and ORHD, show that RRS performs better than LBD independently of the base classifier adopted. As in the case of GACC, RRS outperforms HMD in all datasets when it uses kNN and SVM dichotomizers.

| Dataset | Classifiers | | | | Average |
|---|---|---|---|---|---|
| | kNN | SVM | MLP | ADA | |
| FER | 100/0/0 (1) | 100/0/0 (1) | 60/0/40 (0) | 10/90/0 (1) | 67.5/22.5/0.1 |
| GLASS | 5/90/5 (0) | 0/100/0 (0) | 10/90/0 (1) | 10/90/0 (1) | 6.3/92.5/1.2 |
| IIFI | 90/0/10 (1) | 100/0/0 (1) | 70/0/30 (1) | 57.5/7.5/35 (0) | 79.4/18.7/18.8 |
| ORHD | 90/5/5 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 97.5/1.2/1.3 |
| SAT | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 |
| SUN10 | 90/10/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 97.5/2.5/0.0 |
| WFRN | 65/5/30 (0) | 97.5/0/2.5 (1) | 20/0/80 (1) | 100/0/0 (1) | 70.7/1.2/28.1 |
| WINE | 0/100/0 (0) | 17.5/80/2.5 (0) | 20/70/10 (0) | 30/40/30 (1) | 16.9/7.3/10.8 |
| Average | 67.5/26.3/6.25 | 76.9/ 22.5/6 | 60/20/20 | 63.5/28.4/8.1 | |

Table 4.7.: Exhaustive comparison between the performances of different classifiers expressed in terms of global accuracy (ACC), considering RRS and HMD reconstruction rules. Each tabular shows the amount of win/tie/loss of RRS comparing versus HMD. Round parentheses reports one if the performances computed over the 40 folds are statistically different according to t-test, with a significance level of $0.05$. Otherwise in the round parenthesis there is zero. Last column shows the average values of win/tie/loss achieved by different classifiers on a given dataset, whereas last row shows the average values achieved by a given classifier using different datasets.

**Win/Tie/Loss** RRS has a number of wins larger than HMD in 84% of tabulars shown in Table 4.7. These wins are statistically significant in the 82% of cases.

Similar considerations hold for Table 4.8, where we observe that RRS collects a number of wins larger than HMD in the 81% of tabulars, and the differences are statistically significant in the 69% of tests. Last row and last column of the table show that RRS outperforms HMD whatever the dichotomizer and the dataset, with gaps ranging in [25.1%,74.1%] and [6.3%,100%], respectively. In the case of $F measure$ (Table 4.9), RRS otperforms, in number of wins, HMD in all the cases independently of classifiers (last column) and datasets (last row) used.

### RRS Vs LBD

We compare now RRS and LBD reconstruction rules, i.e. our proposal against the other rule setting the final decision using soft labels.

**Accuracy** In Table 4.1 we observe that, RRS outperforms LBD in the 59% of tabulars, whereas in the 16% of them they perform equally. On the one hand, the rate of success of RRS raises up 75% in case of FER, GLASS, IIFI, SAT, WFRN and WINE datasets, independently of the binary learners used. On the other hand, fixed the SVM classifier while the datasets vary, we found that RRS outperforms LBD in the 60% of the tests.

| Dataset | Classifiers | | | | Average |
|---|---|---|---|---|---|
| | kNN | SVM | MLP | ADA | |
| FER | 100/0/0 (1) | 100/0/0 (1) | 60/0/40 (1) | 60/10/30 (1) | 80.0/2.5/17.5 |
| GLASS | 50/40/10 (0) | 37.5/37.5/25 (0) | 20/20/60 (1) | 80/20/0 (1) | 46.9/29.4/23.7 |
| IIFI | 65/20/15 (1) | 95/2.5/2.5 (1) | 40/10/50(1) | 40/0/60 (1) | 60.0/8.1/31.9 |
| ORHD | 85/10/5 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 96.3/2.5/1.2 |
| SAT | 95/5/0 (1) | 100/0/0 (1) | 90/0/10 (1) | 70/0/30 (0) | 88.8/1.2/10.0 |
| SUN10 | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 |
| WFRN | 50/15/35 (0) | 85/2.5/12.5 (0) | 20/0/80 (1) | 100/0/0 (1) | 63.7/4.4/31.9 |
| WINE | 0/100/0 (0) | 17.5/80/2.5 (0) | 20/70/10 (0) | 30/40/30 (1) | 16.9/72.5/10.6 |
| Average | 68.1/23.8/8.1 | 79.4/15.3/5.3 | 56.3/12.5/31.20 | 72.5/8.8/18.7 | |

Table 4.8.: Exhaustive comparison between the performances of different classifiers expressed in terms of geometric mean of accuracies (GACC), considering RRS and HMD reconstruction rules. Each tabular shows the amount of win/tie/loss of RRS comparing versus HMD. Round parentheses reports one if the performances computed over the 40 folds are statistically different according to t-test, with a significance level of 0.05. Otherwise in the round parenthesis there is zero. Last column shows the average values of win/tie/loss achieved by different classifiers on a given dataset, whereas last row shows the average values achieved by a given classifier using different datasets.

**Geometric Mean** Table 4.2 shows that RRS achieves larger results than LBD in the 56% of tabulars, whereas in the 22% of them they perform equally. We observe that on IIFI, SAT, WFRN and WINE datasets RRS outperforms LBD at least in the 75% of cases. Furthermore, looking at the table by columns we notice that using kNN, SVM, MLP classifiers RRS outperforms LBD in the 62% of tabulars.

**F-measure** Table 4.3 shows that RRS achieves larger results than LBD in the 50% of tabulars, whereas in the 13% of them they perform equally. We observe that on FER, GLASS,IIFI, WFRN and WINE datasets RRS outperforms LBD at least in the 75% of cases. Furthermore, looking at the table by columns we notice that using kNN, SVM classifiers RRS outperforms LBD in the 62% of tabulars.

**Win/Tie/Loss** The results of win/tie/loss comparisons between RRS and LBD in terms of ACC and GACC are reported in Tables 4.10 and 4.11, respectively. Last column of Table 4.10, which averages out the win/tie/loss along the various dichotomizer architectures, shows that (i) RRS has a number of wins larger than LBD in six out of eight datasets, with gap ranging in [15.0%, 67.5%]; (ii) in the two other datasets where LBD outperforms RRS, the performance gap is smaller than before and it ranges in [1.3%, 35.6%]. The last row of the same table averages out the results along the datasets and it shows that RRS has a number of wins larger than LBD, with performance gap ranging in [9.9%, 59.4%]. Tests are statistically significant in the 53.12% of cases.

Turning our attention to results expressed in terms of GACC, last column of Table 4.11

| Dataset | Classifiers | | | | Average |
|---|---|---|---|---|---|
| | kNN | SVM | MLP | ADA | |
| FER | 100/0/0 (1) | 100/0/0 (1) | 50/0/50 (1) | 90/0/10 (1) | 85/0/15 |
| GLASS | 55/35/10 (0) | 47.5/2.5/50 (0) | 50/0/50 (0) | 80/0 /20 (1) | 58/9/33 |
| IIFI | 85/0/15 (1) | 97.5/0/2.5 (1) | 40/0/60 (1) | 42.5/0/57.5 (0) | 66/0/34 |
| ORHD | 85/0/15 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 96/0/4 |
| SAT | 95/0/5 (1) | 100/0/0 (1) | 90/0/10 (1) | 70/0/30 (1) | 89/0/11 |
| SUN10 | 95/0/5 (1) | 100/0/0 (1) | 100/0/0 (1) | 100/0/0 (1) | 99/0/1 |
| WFRN | 50/0/50 (0) | 95/0/5 (0) | 20/0/80 (1) | 100/0/0 (1) | 66/0/34 |
| WINE | 0/100/0 (0) | 17.5/80/2.5 (0) | 20/70/10 (0) | 30/30/40 (1) | 17/70/13 |
| Average | 71/17/13 | 82/10/8 | 59/9/33 | 77/4/20 | |

Table 4.9.: Exhaustive comparison between the performances of different classifiers expressed in terms of $F\,measure$, considering RRS and HMD reconstruction rules. Each tabular shows the amount of win/tie/loss of RRS comparing versus HMD. Round parentheses reports one if the performances computed over the 40 folds are statistically different according to t-test, with a significance level of $0.05$. Otherwise in the round parenthesis there is zero. Last column shows the average values of win/tie/loss achieved by different classifiers on a given dataset, whereas last row shows the average values achieved by a given classifier using different datasets.

shows that RRS has a number of wins larger than LBD in six out of eight datasets, with gap ranging in [1.5%, 60.5%]. Last row shows that RRS outperforms LBD in all cases, with a difference between wins and losses cases ranging in [12.5%, 53.4%].

Focusing on the results expressed in terms of $F\,measure$, last column of Table 4.12 shows that RRS has a number of wins larger than LBD in six out of eight datasets, with gap ranging in [18%, 60%]. Last row shows that RRS outperforms LBD in all cases, with a difference between wins and losses cases ranging in [16%, 57%].

## Global Comparison

Figure 4.5 presents global comparison between the 12 tested algorithms. Values are reported in terms of average results obtained on the 8 domains on the three metrics (ACC, GACC and $F\,measure$) and in terms of ranking results. In each plot, tested algorithms'name is reported concatenating the base classifier's name with the reconstruction method's name. As an example considering the SVM, as the base classifier, and the RRS, as the reconstruction method, resulting algorithm's name is SVMRRS. On the left side of the figure we report the mean values and the standard deviation for each algorithm respect each metrics: ACC (top), GACC (middle), $F\,measure$ (bottom). Algorithms are ordered according to the average value of the metric at hand. We note that, using RRS, a classifier rank first compared to when it is adopted using other reconstruction schemes. It is worth nothing that classification algorithm showing larger performance, on all the metrics, is SVM ranking first and second, respectively.

To drill down into these general results, we have also computed the global ranking results of

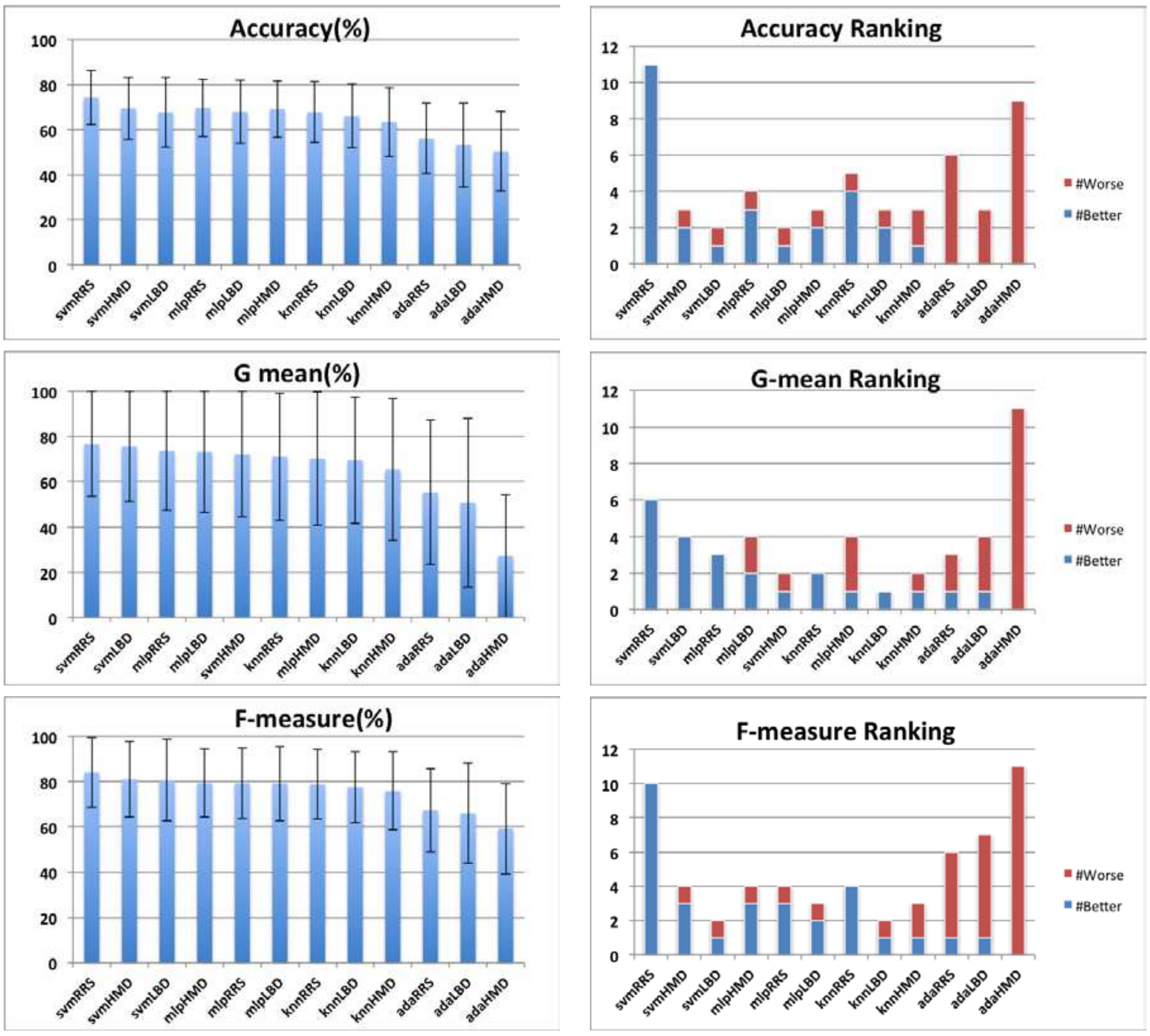

Figure 4.5.: *Left*. Average ± Standard deviation for the accuracy (top), Geometric mean of relative accuracies (middle), $F\,measure$ (bottom) over the 8 domains for all the classification schemes. ***Right***. Ranking results: number of times each algorithm performed significantly better than the others (blue) or worse (red) according to a Student paired t-test ($p = 0.1$). In each plot, algorithms are ordered from left to right in decreasing average of the metric at hand.

| Dataset | Classifiers | | | | Average |
|---|---|---|---|---|---|
| | kNN | SVM | MLP | ADA | |
| FER | 55/45/0 (0) | 90/10/0 (1) | 40/10/50 (1) | 70/10/20 (1) | 63.8/18.7/17.5 |
| GLASS | 0/100/0 (0) | 100/0/0 (1) | 80/20/0 (1) | 90/10/0 (1) | 67.5/32.5/0 |
| IIFI | 90/5/5 (1) | 90/2.5/7.5 (1) | 50/20/30 (1) | 30/12.5/57.5 (1) | 65.0/10.0/25.0 |
| ORHD | 0/100/0 (0) | 0/100/0 (0) | 0/100/0 (0) | 0/100/0 (0) | 0/100/0 |
| SAT | 0/100/0 (0) | 100/0/0 (1) | 80/0/20 (1) | 70/0/30 (1) | 62.5/25.0/12.5 |
| SUN10 | 30/60/10 (0) | 15/55/30 (0) | 0/100/0 (0) | 0/90/10 (0) | 11.3/76.2/12.5 |
| WFRN | 80/20/0 (0) | 100/0/0 (1) | 20/0/80 (1) | 100/0/0 (1) | 75.0/5.0/20.0 |
| WINE | 15/85/0 (0) | 27.5/70/2.5 (0) | 10/90/0 (0) | 30/50/20 (1) | 20.6/73.8/5.6 |
| Average | 33.8/64.4/1.87 | 65.3/29.7/11.9 | 35/42.5/22.5 | 48.8/34.1/17.1 | |

Table 4.10.: Exhaustive comparison between the performance of different classifiers expressed in terms of global accuracy (ACC), considering RRS and LBD reconstruction rules. Each tabular shows the amount of win/tie/loss of RRS comparing versus LBD. Round parentheses reports one if the performances computed over the 40 folds are statistically different according to t-test, with a significance level of $0.05$. Otherwise in the round parenthesis there is zero. Last column shows the average values of win/tie/loss achieved by different classifiers on a given dataset, whereas last row shows the average values achieved by a given classifier using different datasets.

each algorithm, recording the number of times each one ranked first, second, third and so on, over the 8 domains. In figure 4.5, we report these results on the right side. To bring statistical validation to these ranking, we performed Student paired t-test comparison for each algorithm against all others ($12x12 = 144$ comparisons), recording those for which we can reject the null hypothesis for level $p = 0.1$, and then clustering the significant differences as to whether they are *better* (blue), or *worse* (red), of the algorithm at hand. Once again, we notice that learners collect a larger number of significant wins when RRS is used rather than when other reconstruction rules are adopted.

## 4.3.2. Results on artificial datasets

The four artificial datasets highlight performance differences between reconstruction rules in a controlled scenario where only the number of classes vary, whereas the samples are drawn from a normal distribution.

For each run of the stratified cross validation, we perform a 5-fold cross validation.

Preliminarily, we observe that performance differences between RRS and LBD both in terms of ACC and GACC are less than 0.5% in average, and not statistically significant. For this reason in this subsection we will consider RRS and HMD, under the remark that all observations made for RRS hold also for LBD.

Figure 4.6 reports the accuracy for each dichotomizer on the four datasets, where the x-axis shows the number of classes of each artificial datasets and the y-axis reports the value of accuracy. Blue and red lines correspond to RRS and HMD results, respectively. In this figure, RRS always outperforms HMD. In case of kNN classifier, we notice that RRS outperforms

| Dataset | Classifiers | | | | Average |
|---|---|---|---|---|---|
| | kNN | SVM | MLP | ADA | |
| FER | 55/45/0 (0) | 90/10/0 (1) | 40/0/60 (1) | 0/80/20 (0) | 46.25/36.25/20 |
| GLASS | 0/100/0 (0) | 0/90/10 (1) | 10/90/0 (1) | 10/90/0 (1) | 5/92.5/2.5 |
| IIFI | 95/0/5 (1) | 92.5/0/7.5 (1) | 60/10/30 (1) | 35/0/65 (0) | 70.6/2.5/26.9 |
| ORHD | 0/100/0 (0) | 0/100/0 (0) | 0/100/0 (0) | 0/100/0 (0) | 0/100/0 |
| SAT | 0/100/0 (0) | 100/0/0 (1) | 80/0/20 (1) | 90/0/10 (1) | 67.5/25/7.5 |
| SUN10 | 30/55/15 (0) | 15/55/30 (0) | 0/100/0 (0) | 0/90/10 (0) | 11.25/75/13.75 |
| WFRN | 80/20/0 (0) | 100/0/0 (1) | 20/0/80 (1) | 100/0/0 (1) | 75/5/20 |
| WINE | 15/85/0 (0) | 27.5/70/2.5 (0) | 10/90/0 (0) | 30/50/20 (1) | 52.58/73.75/5.62 |
| Average | 34.4/63.1/2.5 | 53.1/40.6/6.3 | 28.8/ 48.7/23.7 | 33.2/51.2/15.6 | |

Table 4.11.: Exhaustive comparison between the performance of different classifiers expressed in terms of geometric mean of accuracies (GACC), considering RRS and LBD reconstruction rules. Each tabular shows the amount of win/tie/loss of RRS comparing versus LBD. Round parentheses reports one if the performances computed over the 40 folds are statistically different according to t-test, with a significance level of $0.05$. Otherwise in the round parenthesis there is zero. Last column shows the average values of win/tie/loss achieved by different classifiers on a given dataset, whereas last row shows the average values achieved by a given classifier using different datasets.

HMD with a difference ranging between $1.7\%$ and $6.3\%$. Using the SVM classifier this difference ranges between $3.2\%$ and $24.4\%$. In case of ADA the gap between the two reconstruction rules ranges between $6.1\%$ and $15.7\%$. Finally, in case of MLP, accuracy improvement of RRS with respect to HMD ranges between $1.5\%$ and $5.4\%$. Furthermore, the charts in Figure 4.6 show that the accuracies decreases as the number of classes increase: this result is expected since a larger number of classes imply a more complex dataset. Nevertheless, it is worth observing that RRS drops the performances less than HMD since in many cases the accuracies gap between RRS and HMD increases with the complexity of the recognition task.

Let us now focus the attention to the performance for each class. To this aim, Figure 4.7 plots the values of accuracies per class. For the sake of brevity and to not overload such graphs with many curves, we report results achieved using one classification architecture for each dataset. Furthermore, in this figure we order class labels so as class with more samples came first. For instance, chart SVM-SIM1 represents the values of $acc_j$ provided by the two reconstruction rules when an SVM is used as a dichotomizer. The x-axis reports the number of classes, and the corresponding ordinates are the values of accuracies for those classes. In general, we observe that RRS outperforms HMD in each class except in the first one. This should be expected since improving the recognition ability on the minority classes usually harms the hit rate on the majority one, as it also been notices in case of binary skewed classification problems [125].

| Dataset | Classifiers | | | | Average |
|---|---|---|---|---|---|
| | kNN | SVM | MLP | ADA | |
| FER | 60/ 40/0 (0) | 100/0/ 0 (1 ) | 50/0/50 (1) | 80/0/20 (1) | 73/10/18 |
| GLASS | 10/85/5 (0) | 82.5/0/17.5 (1 ) | 80/10/10 (1) | 100/0/0 (1) | 68/ 24/8 |
| IIFI | 90/0/10 (1) | 92.5/0/7.5 (1) | 70/0/30 (1) | 35/0/65 (1) | 72/0/28 |
| ORHD | 0/100/0 (0) | 0/100/0 (0 ) | 0/100/0 (0) | 0/100/0 (0) | 0/100/0 |
| SAT | 0/100/0 (0) | 100/0/0 (1 ) | 80/0/20 (1) | 70/0/30 (0) | 63/25/13 |
| SUN10 | 30/45/ 25 (0) | 15 /52,5/32,5 (0) | 0/100/0 (0) | 0/80 /20 (0) | 11/69/19 |
| WFRN | 15 /85/ 0 (0) | 30/67,5/2,5 (0) | 20/80/0 (0) | 30/50/20 (1) | 24/71/6 |
| WINE | 95 /5/0 (1) | 100 /0/0 (1) | 20/0 /80 (1) | 100/0/ 0 (1) | 79/1/20 |
| Average | 38 / 58 / 5 | 65 / 28 / 8 | 40 / 36 / 24 | 52 / 29 / 19 | |

Table 4.12.: Exhaustive comparison between the performance of different classifiers expressed in terms of $F\,measure$, considering RRS and LBD reconstruction rules. Each tabular shows the amount of win/tie/loss of RRS comparing versus LBD. Round parentheses reports one if the performances computed over the 40 folds are statistically different according to t-test, with a significance level of $0.05$. Otherwise in the round parenthesis there is zero. Last column shows the average values of win/tie/loss achieved by different classifiers on a given dataset, whereas last row shows the average values achieved by a given classifier using different datasets.

## 4.4. Discussion

As a first issue, we notice that in imbalance classification tasks the reconstruction rules based on soft labels, i.e. LBD and RRS, provide larger performances than a rule using the crisp labels only, i.e. HMD. Indeed, the former reconstruction rules in most of the experiments provide larger values of both accuracy and geometric mean of accuracies. This therefore suggests us that they are more suited than the latter to tackle with class skew. Indeed our quantitative assessment on real and synthetic datasets confirms the intuition that the use of soft labels enriches the information available to the reconstruction rule, thus permitting to derive more effective criterion. Although this observation should appear straightforward, to the best of our knowledge, this issue has not been discussed so far in the literature where most of the existing works focusing on OpC decomposition report only the accuracy and they do not look at the performances on single and/or under-represented classes. Furthermore, the large number of tests allows also to quantify this improvement, as detailed in previous sections. Generally, experiments on artificial datasets show that RRS reconstruction rule, which uses classification reliabilities and it is therefore based on soft labels, outperforms the reconstruction rule using crisp labels only (HMD) whatever the number of samples and classes in the datasets. This consideration also holds for the experiments in real datasets, where we find out that performances raise on datasets with different degree of imbalance. For instance, LBD and RRS achieve the largest performance improvements in comparison to HMD on FER, ORHD, SAT and SUN10 datasets, although they have very different a-priori sample distributions (Figure 4.4).

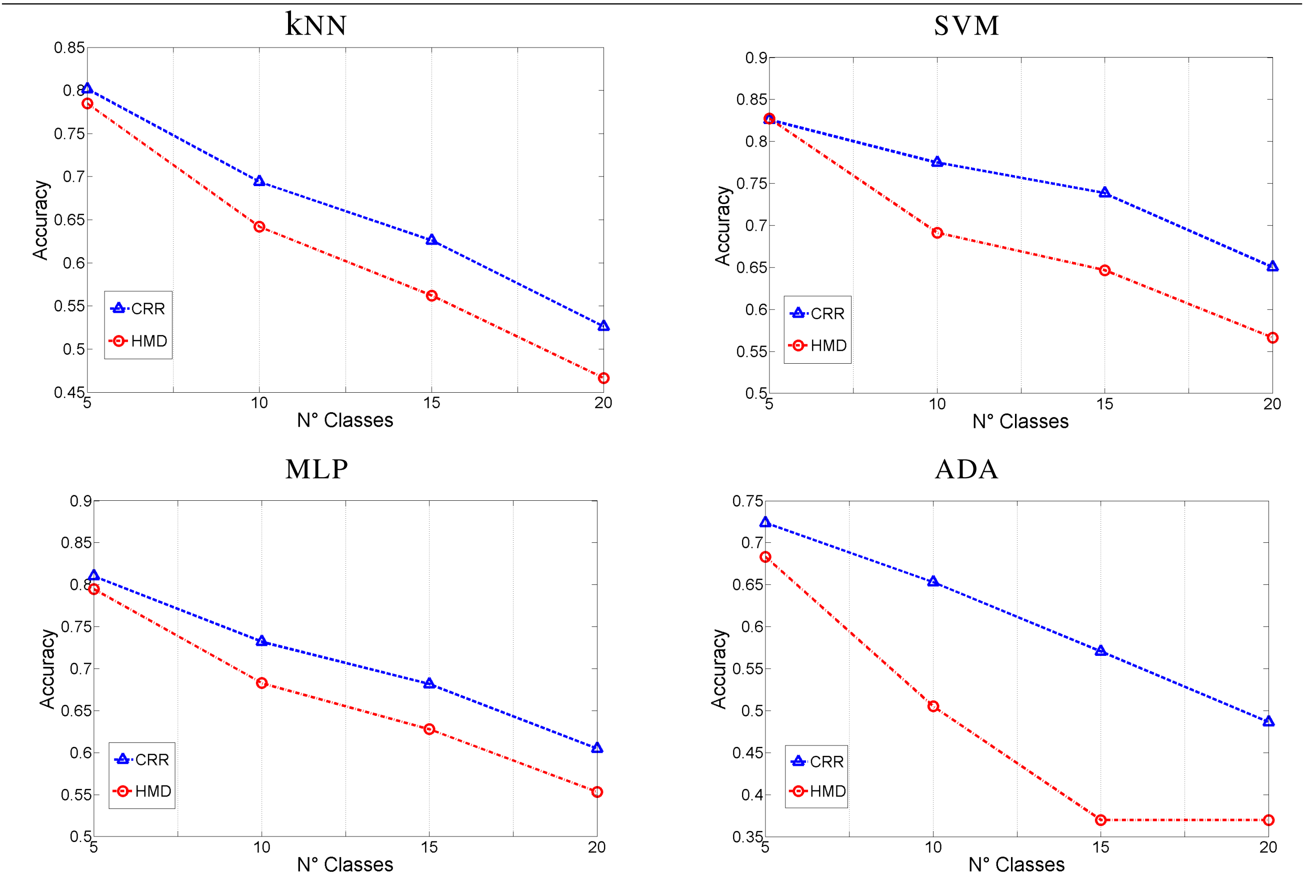

Figure 4.6.: Results of kNN, SVM, MLP andADA classifier measured in terms of accuracy. Blue and red lines correspond to RRS and HMD results, respectively.

As a second issue, we focus on results measured in terms of accuracy per class. The experiments on artificial datasets show that on most of the classes RRS provides values of $acc_j$ larger than HMD. This consideration holds also for tests on real datasets, although we do not burden the manuscript with the corresponding large number of plots. Broadly, RRS achieves more balanced performances among the classes since very often it provides values of GACC and $F\,measure$ larger than HMD and LBD.

The third issue discusses how much the reconstruction rules provide performances which are balanced between ACC and GACC. Indeed, the analysis of the literature on binary imbalance classification task points out that very often the miss rate on the majority class raises when the the hit rate on the minority class raises too. This phenomenon increases the value of $g$ but lowers the value of $acc$ [85, 125, 141]. A similar analysis in case of imbalanced multiclass classification task is missing in the literature. Although its complete description and discussion is out of the scope of this work, we provide a first attempt to analyze this behaviour

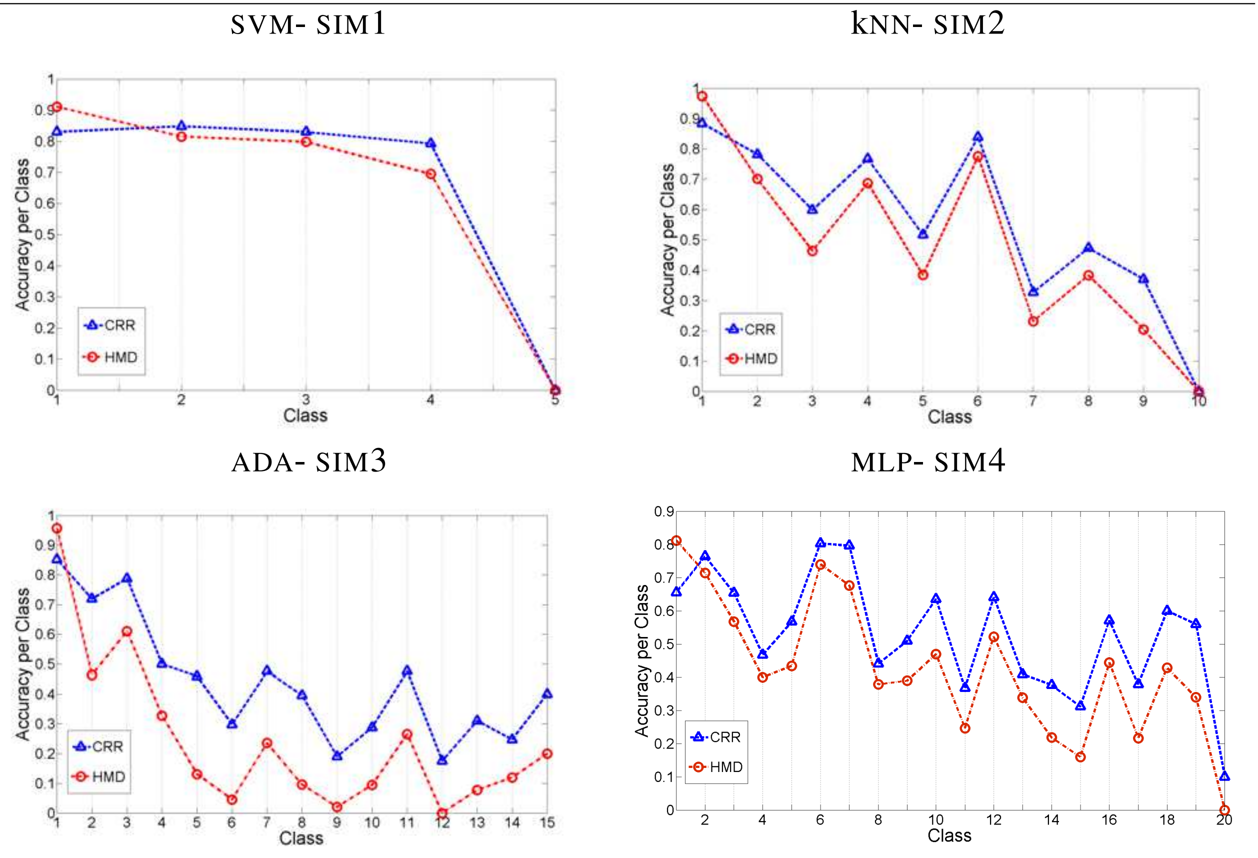

Figure 4.7.: Accuracy per class on the synthetic datasets. The title of each chart reports both the classification architecture and the dataset considered. Class labels are ordered so as class with more samples came first.

by computing the following quantity [125]:

$$\beta = \sqrt{\frac{(1-acc)^2 + (1-g)^2}{2}} \tag{4.6}$$

The measure $\beta$ can be easily interpreted considering the $xy$ plane, where $x$ and $y$ axes correspond to ACC and GACC, respectively. The performances of a classifier measured as $(acc, g)$ pair get one point in $[0,1]$x$[0,1]$ and, hence, $\beta$ ranges in $[0,1]$. Furthermore, the ideal and the worst performance corresponds to points $(1,1)$ and $(0,0)$, respectively. The closer the point representing classifier performance to the ideal point, the more balanced the performance over the classes. On this basis, we compute $\beta$ for the four dichotomizer architectures and the eight datasets used. Then, we normalize values provided by RRS and LBD with respect to values of HMD, achieving $\overline{\beta}$. Such data are graphically represented as follows (Figure 4.8). For each dataset and reconstruction rule (RRS and LBD), we determine the dichotomizer providing the minimum value of $\overline{\beta}$, i.e. the best one, and we report this value in the figure. Moreover, for each point of the figure, we draw also a geometric shape giving information on the consid-

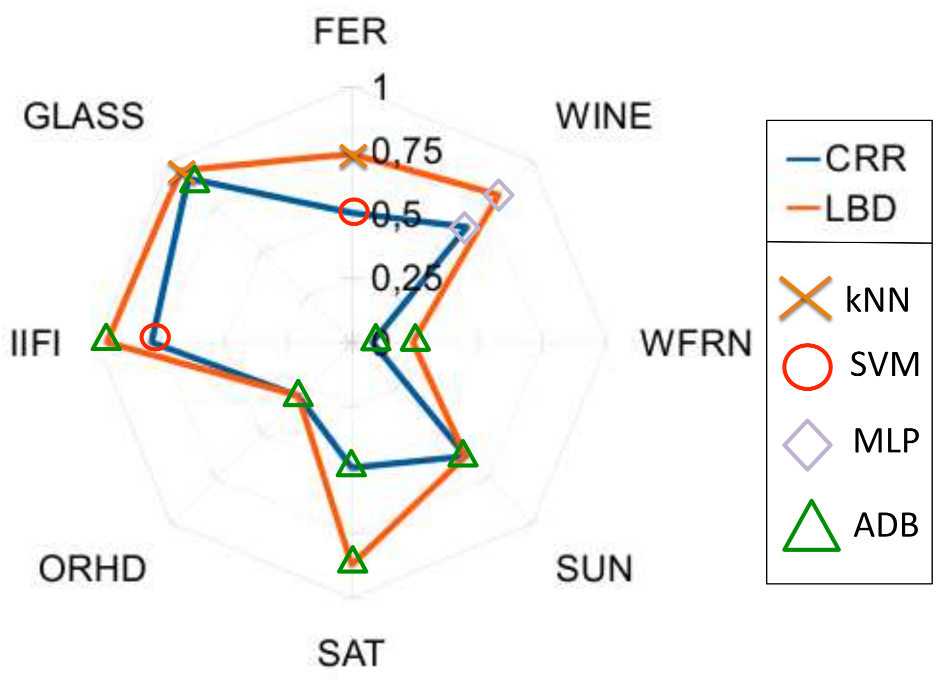

Figure 4.8.: Radar plot of $\overline{\beta}$ values for RRS and LBD reconstruction rules.

ered dichotomizer. For instance, consider the FER dataset. The orange and blue lines represent RRS and LBD performance: the corresponding dichotomizers are the SVM and the kNN, whereas $\overline{\beta}$ values are 0.51 and 0.74, respectively. The figure permits us to derive the following observation. First, values of LBD and RRS are always below 1: this means that they provide values of $\beta$ smaller than HMD, thus being able to improve the recognition capability over the classes. This observation confirms once more the first issue of this section, i.e. soft labels are more suited than crisp labels to tackle with class skew. Second, we notice that the blue line is always inner to the orange one. This implies that $\overline{\beta}$ values provided by RRS are always smaller, and therefore better, than those of LBD, thus confirming that RRS in several cases achieves values of ACC and GACC that are, together, larger than those attained by other rules. Hence, RRS improves the recognition ability on the minority classes affecting the recognition accuracies on majority classes to a lesser extent than the other rules.

As final remark we observe that the SVM architecture seems to be the best suited to work with the proposed reconstruction rule since the overall classification system. Indeed, the performances (ACC GACC and $F\,measure$) achieved when this classifier is used as dichotomizer are larger than those achieved by other classifiers, when RRS is used. In the case of ACC, Table 4.1 shows that SVM classifier achieves larger results than kNN, MLP and ADAin the 62.5% of cases. In particular, using this classifier RRS outperforms HMD and LBD in the 87.5% and 75.0% of cases. Turning our attention to performances measured in terms of GACC and $F\,measure$, Table 4.2 shows that SVM classifier shows larger performances than others in the 75% of cases, when RRS is used. Furthermore, respect to GACC results, RRS outperforms HMD and LBD in the 87.5% and 62.5% of tabulars respectively. Respect to $F\,measure$ results our proposal outperform HMD and LBD in the 100% and 75% of tabulars respectively.

# 5. Reliability-based Softmax reconstruction rule

Results achieved RRS rules, presented in the previous Chapter 4, show that using the classification reliability in the reconstruction stage leads to an improvement of the systems' performances. Motivating by these favourably results we aim to further investigate the effect of this quantity in designing reconstruction rule suited for skewed data. With this aim, we notice that in a decomposition scheme for each input sample a dichotomizer produces a raw output ($\in \Re$) that can be transformed in a reliability value. Hence, after the dichotomizer classifications, we have a vector that collects all these real values, describing each input sample in a new feature space. The task to assign to this vector a label that corresponds to the final decision of the system is a classification problem itself. Considering the problem from this point of view, we investigate the reconstruction rules that address the problem using a statistical approach. Inspired by a statistical reconstruction rule [122] that was designed for Opc and PC decomposition methods, we present here an extension of this method in the case of ECOC decomposition approach. Since our final task is to handle imbalanced datasets, and aware of our study of the use of reliability at reconstruction level, we decide to improve the existing rule using reliabilities instead of raw classifiers outputs. The resulting reconstruction rule is referred to as Reliability-based Softmax reconstruction rule (RBS). Results achieved testing this rule on eight datasets and three classifiers show two main results. The first one is that the proposed rule improves system recognition performance both in therm of accuracy, geometric mean of accuracies and $F measure$ when compared with well established reconstruction rules. The second one, according with the results that we achieved in the other proposals, shows that the reliability improves system performance. The latter result arises from the comparison of the statistical method [122] extended to the ECOC framework when it use reliability and when it use only the raw outputs.

Next section presents RBS method and, at the end of the section, discusses the differences between our proposal and [122]. In the sections 5.2 and 5.3 we present the experimental set-up and the results, respectively. In the last section we discuss results achieved.

## 5.1. Method

When we use RBS reconstruction rule we can considers dichotomizers' outputs as a new feature vector which have to be classified. We present here an approach that solves this classification task using the Softmax regression. According to the ECOC decomposition method a unique codeword, i.e. a binary string, is assigned to each class. Assuming that the string has $L$ bits, the recognition system is composed by L binary classification functions. These binary classifiers provide the binary decision vector $\mathbf{M}(x)$ and the reliability vector $\boldsymbol{\psi}(x) = \{\psi_1(x), \psi_2(x), \dots, \psi_L(x)\}$.

Since we aim at designing a reconstruction rule suited for imbalanced datasets, we want use information owned by classifiers reliability in the reconstruction stage. In order to do this, we introduce the quantity $\chi_j(x)$ that summarizes both the information provided by classifiers. Indeed $\chi_j(x)$ integrates the crisp label and the classification reliability that the $j$-th binary classifier provides for each sample $x$ by multiplying them. Hence, for the whole decomposition we have: $\boldsymbol{\chi}(x) = \boldsymbol{\psi}(x)^T \odot \boldsymbol{M}(x) = \{\chi_1(x), \chi_2(x), \dots, \chi_L(x)\}$, where the symbol $\odot$ represents the element-wise product.

Considering now $\boldsymbol{\chi}(x)$ as second-order features, we have to face with the classification problem $\{\boldsymbol{\chi}(x), \omega(x)\}$, where each sample $\boldsymbol{\chi}(x)$ is a vector described by $L$ features with label $\omega(x)$. The classification task consists in predicting the label $y(x) \in \boldsymbol{\Upsilon}$, where $\boldsymbol{\Upsilon} = \{y_1(x), y_2(x), \dots, y_K(x)\}$, so that $\omega(x) = y(x)$ for each sample. For the sake of clarity we omit in the following the dependence of all symbols from sample $x$.

We solve this classification task by using the softmax regression to estimate the posterior distribution of classification acts. Softmax regression is a natural choice since multiclass problems show multinomial distribution for the output. Defining a set of $K-1$ vectors of parameters, $\Theta = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_{K-1}\}$, to parameterize the multinomial distribution over $K$ different outputs, the conditional distribution of $y$ given $\boldsymbol{\chi}$ is:

$$p(\omega = y_i|\boldsymbol{\chi}; \Theta) = \frac{e^{\boldsymbol{\theta}_i^T \boldsymbol{\chi}}}{\sum_{j=1}^{K} e^{\boldsymbol{\theta}_j^T \boldsymbol{\chi}}} \quad i = 1, 2, \dots, K-1. \tag{5.1}$$

It is straightforward observing that $p(\omega = y_K|\boldsymbol{\chi}; \Theta) = 1 - \sum_{i=1}^{K-1} p(\omega = y_i; \Theta)$. The final label is set by:

$$y = argmax_i(p(\omega = y_i|\boldsymbol{\chi}; \Theta)). \tag{5.2}$$

In order to perform this reconstruction technique we have to estimate $\Theta$. To this aim, consider a training set $tr$ composed of $m_{tr}$ samples. Denoted by $\boldsymbol{\chi^{tr}}$ the values of $\boldsymbol{\chi}$ of samples belonging to $tr$, $\Theta$ can be estimated maximizing the log-likelihood $l$:

$$l(\theta) = \sum_{i=1}^{m_{tr}} log \prod_{l=1}^{K} (\frac{e^{\boldsymbol{\theta}_l^T \boldsymbol{\chi}_i^{tr}}}{\sum_{j=1}^{k} e^{\boldsymbol{\theta}_j^T \boldsymbol{\chi}_i^{tr}}})^{1\{y_i = l\}} \tag{5.3}$$

where $1\{\circ\}$ denotes the index function, which is one if the statement inside the bracket is true, zero otherwise.

To reduce the correlation between classifier outputs when we perform the maximization of eq. 5.1 we use $L2$ penalty, as suggested in [122]. Note that $\boldsymbol{\chi^{tr}}$ is computed performing a stacking procedure, which avoids problem of reusing training samples during parameter estimation. Indeed, we first divide $tr$ into $p$ folds, and then use $p-1$ folds for training and one to estimate $\boldsymbol{\chi_h^{tr}}$, where $h \in [1; p]$. When all folds were considered as test fold, we compute $\boldsymbol{\chi^{tr}} = \{\boldsymbol{\chi_h^{tr}}\}_{h=1}^{p}$.

**Algorithm 1:** Reliability-based Softmax reconstruction rule

```
Input: $\mathcal{Z} = \{(\boldsymbol{x}_i, \omega_i)\}$ with $i = 1, ..., N$ $\boldsymbol{x_i} \in \Re, \omega_i \in \Omega = \{\omega_1, \dots, \omega_K\}\}$;
Input: D : $K \times L$ code matrix.
1: $\forall \boldsymbol{x} \in \mathcal{Z}$ map $\omega \to \{-1, 1\}^L$ using $\boldsymbol{D}$.
2: Split $\mathcal{Z}$ in T folds.
3: Initialize second order feature set: $\mathcal{S} = \emptyset$.
4:
repeat
    Get T-1 folds as training set ($\mathcal{Z}_{Tr}$) and 1 as test set ($\mathcal{Z}_{Te}$).
    for $t = 1$ to $K$ do
        -Train binary classifier $\mathcal{C}_t$ using $\mathcal{Z}_{Tr} = \{(\boldsymbol{x}_j, \boldsymbol{y}_j), \boldsymbol{y_j} \in \{-1, 1\}\}$
        -Test binary classifier $\mathcal{C}_t$ using $\mathcal{Z}_{Te}$
        -Collect hard label and reliability: $\{(M(\boldsymbol{x}), \psi(\boldsymbol{x})),\ \forall \boldsymbol{x} \in \mathcal{Z}_{Te}\}^t$
    end for
    -Compute the $\boldsymbol{\chi}$: $\boldsymbol{\chi}(\boldsymbol{x}) = \boldsymbol{M}(\boldsymbol{x}) \odot \boldsymbol{\psi}(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathcal{Z}_{Te}$.
    -Collect the new features : $\mathcal{S}.add\{\chi(\boldsymbol{x}), \omega), \forall \boldsymbol{x} \in \mathcal{Z}_{Te}\}$.
until all folds tested
5:
repeat
    Get T-1 folds as training set ($\mathcal{S}_{Tr}$) and 1 as test set ($\mathcal{S}_{Te}$) from $\mathcal{S}$.
    -Compute the Softmax regression parameters $\Theta = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_{K-1}\}$ using $\mathcal{S}_{Tr}$
    -Compute the probability $p(\omega = y_i|\boldsymbol{\chi}; \Theta)$ for each sample in $\mathcal{S}_{Te}$ respect each class.
    -Compute sample class $y = argmax_i(p(\omega = y_i|\boldsymbol{\chi}; \Theta))$
until all folds tested
```

**Reliability-based Softmax reconstruction rule pseudo code** The pseudo-code of the proposed method is reported in Algorithm 1. The inputs are the dataset $\mathcal{Z}$ composed of $N$ pairs sample-label $\{(\boldsymbol{x}_i, \omega_i)\}$ and the code matrix $\boldsymbol{D}$. The six steps of the algorithm do the following:

1. We use $\boldsymbol{D}$ to map the multiclass labels of all samples in the new set of binary labels, so that the label of each sample is now a vector of $K$ binary values.

2. Our experiments are performed according to a $T$-fold cross validation: hence, in the second step we divide the original dataset in $T$ folds.

3. We define a new set, named as *Second order feature set* $\mathcal{S}$ representing the new meta-level dataset which will used to compute the Softmax regression. In $\mathcal{S}$ each sample is described by a new feature vector given by $\boldsymbol{\chi}(\boldsymbol{x})$, computed in the following step of the algorithm.

4. For each iteration of the cross validation, we use $T - 1$ folds to train the pool of $L$ classifiers: $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, \dots, \mathcal{C}_j, \dots, \mathcal{C}_L\}$, whereas one fold is used to test $\boldsymbol{\mathcal{C}}$. We first collect the labels and the reliabilities of the classification on all samples of the test set

$\mathcal{Z}_{Te}$ and then we compute the quantity $\chi(\boldsymbol{x})$. This quantity and its true multi-class label are added to $\mathcal{S}$.

5. At this point, after that all the folds were classified in the previous step, in the set $\mathcal{S}$ are collected the second level features for each samples in $\mathcal{Z}$. We divide the new dataset $\mathcal{S}$ in $T$ folds according to the splitting that have been performed on the original set $\mathcal{Z}$. For each iteration of the cross validation, we use $T-1$ folds to estimate parameters $\Theta$ maximizing the log-likelihood 5.1 and the test fold to compute posterior probabilities: $p(\omega = y_i|\boldsymbol{\chi};\Theta)$. Final system outputs are computed as $y = argmax_i(p(\omega = y_i|\boldsymbol{\chi};\Theta))$.

**Remarks** We discuss now the differences between our proposal and the contribution presented in [122]. These differences consist into three main points.

First, in [122] the authors use the raw outputs of the binary classifiers $\boldsymbol{f}(x) = \{f_j(x)\}_{j=1}^{L}$ to compute the features of second order $\boldsymbol{\chi}(x) = \boldsymbol{\psi}(x)^T \odot \boldsymbol{f}(x)$. This choice does not permit to use classifiers providing only crisp labels, e.g. the k-Nearest Neighbour. Conversely, our contribution uses the quantity $\boldsymbol{\chi}(x) = \boldsymbol{\psi}(x)^T \odot \boldsymbol{M}(x)$, which combines the crisps labels with the reliability, i.e. a measure providing us more information about the classification process. Note that this choice permits us to employ any kind of classifiers.

Second, in [122] OpC and PC decomposition are considered, whereas we focus on the ECOC framework.

Third, the performances of RBS reconstruction rule are assessed with particular reference to classification of samples belonging to under-represetend classes.

Note that the novel use of the reliability in the regression not only extends the work of [122], but provides larger classification performance as will be reported in Section 5.3.

## 5.2. Experimental set-up

In this section we give a short description of the specific experimental set-up used to validate our proposal. We first provide the list of datasets used in our tests, then the list of performance metrics chosen and finally we present details of the experimental protocol.

### 5.2.1. Datasets

From the dataset presented in 3.1, we use 8 public datasets (BRTISS, DERM, ECOLI, FER, GLASS, IIFI, SEEDS, WINE) which provide an heterogeneous set of classification tasks in terms of number of samples, features and classes. Datasets shows also different skewness among classes permitting to assess how the classification system performs when a class is under-represented in comparison to others. Their characteristics are summarized in Table 3.1.

As previously done in Chapter 4, in Figure 5.1 we graphically represent the degree of imbalance of the used datasets[88]. The chart represents the prior probability of the minority class as a function of prior probability of the majority class. The feasible space is below the diagonal line of the plot, which also corresponds to equiprobable classes. This line can be therefore thought of as the edge of balanced problems. The balance disappears towards the

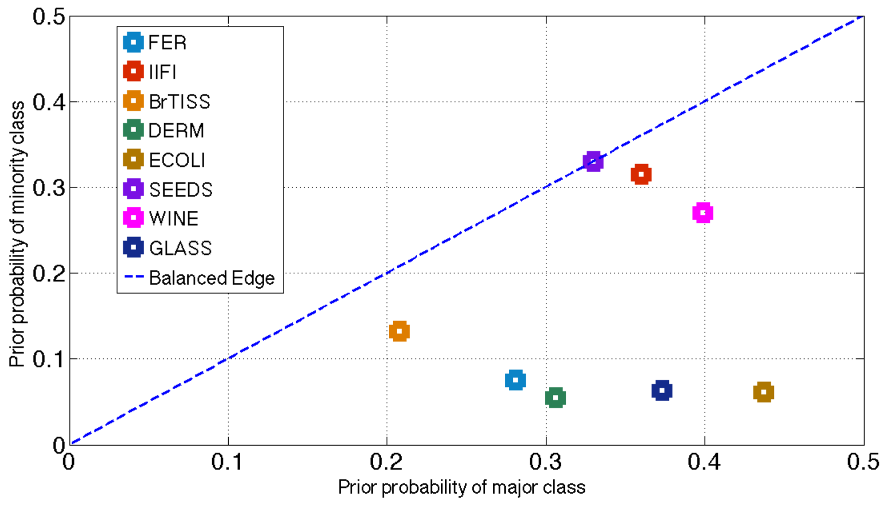

Figure 5.1.: Dataset distribution as function of prior distribution of majority (x-axis) and minority (y-axis) class.

bottom right corner. Point (0.5,0.5) corresponds to two equiprobable classes. This graphical representation helps us to observe how much the datasets are heterogeneous with respect to the imbalance ratio. In the figure we observe that SEEDS dataset is perfectly balanced whereas datasets such as GLASS and DERM have a strong degree of imbalance.

## 5.2.2. Classifiers

We test three different types of classifiers, belonging to different classification para-digms. Therefore, the binary learners are: Adaboost (ADA) as an ensemble of classifiers, Multilayer Perceptron (MLP) as a neural network and Support Vector Machine (SVM) as as a kernel machine.

- **SVM** we use a Gaussian radial basis kernel. Values of regularization parameter $C$ and scaling factor $\sigma$ are selected within $[1, 10^4]$ and $[10^{-4}, 10]$, adopting a $\log$ scale to sample the two intervals. The value of each parameter is selected according to average performance estimated by five fold cross-validation on a validation set. The reliability of a SVM classification is estimated as proposed in [107], where the decision value of the SVM is transformed in a posterior probability.

- **MLP** we use a number of hidden layers equal to half of the sum of features number plus class number. The number of neurons in the input layer is given by the number of the features. The number of neurons in the output layer is always two when the MLP is employed as dichotomizer. To evaluate the reliability of MLP decisions for multiclass classification problems we adopted a method that estimates the test patterns credibility on the basis of their quality in the feature space [27].

- **ADA** we use as the ”Adaboost M1” algorithm proposed in [53], where weak learners are decision stumps. The number of iteration is equal to 100. The reliabilities of ADA classifications are estimated using an extension of method [27], where we compute the difference between the outputs related to winning and losing class.

### 5.2.3. Performance metrics

On the motivations discussed in section 3.3 we use as performance metrics the accuracy ($acc$) and the geometric mean of relative accuracies (GACC).

### 5.2.4. Experimental protocol

We test RBS method to solve multiclass tasks in ECOC framework. We apply ECOC using the method proposed by Dietterich et al. [40] for code generation. In particular, if $3 \leq K \leq 7$ we use exhaustive codes; if $8 \leq K \leq 11$ we generate exhaustive codes and then select a good subset of decomposition matrix columns given by the GSAT algorithm [119]. In this decomposition framework we compare the proposed reconstruction rule with HMD, LBD, and with the method proposed in [122], which is referred to as SHI in the following. In this latter case, according to [122], the softmax regression is applied on the second order features computed starting from classifiers soft labels $\boldsymbol{f}(x) = \{f_j(x)\}_{j=1}^{L}$. Hence $\boldsymbol{\chi}(x)$ is computed as $\boldsymbol{\psi}(x)^T \odot \boldsymbol{f}(x)$. Furthermore, all experiments reported in the following are performed according to a five folds cross validation.

## 5.3. Results

This section presents the results achieved by the three classifiers on the tested datasets varying the reconstruction rule used, as reported in section 5.2.

Tables 5.1, 5.2 and 5.3 report results obtained by SVM, MLP and ADA classifiers, respectively. For each table, the top, middle and right side reports performance measured in term of accuracy, geometric mean of accuracies and $F measure$respectively.

Let us now focus on the results obtained by the SVM classifier (Table 5.1). First we observe that results measured in terms of accuracy show that RBS performs better than others methods for all datasets. Second results expressed in terms of GACC show that RBS outperforms other methods in 7 cases out of 8. Third $F measure$ values show that RBS outperforms other methods in 6 out of 8 datasets. Focusing now on the most imbalanced domains such as ECOLI, FER, GLASS and WINE (Figure 5.1), we observe that RBS achieve for all the three metrics better performance that other methods in all domains ad exception on ECOLI in term of GACC. Consider the GLASS dataset we notice that RBS show largest performance differences with other methods. In this case, the second best method is SHI, but its value of GACC is $32.18\%$ lower than the one provided by RBS.

Turning our attention to Table 5.2 reporting results achieved by MLP classifier, we observe that in terms of ACC RBS achieves the best results in four out of eight datasets. However, there is not another prevalent method, since best performance are attained by SHI and LBD method in one and two datasets, respectively. With respect to GACC , RBS method shows: (i) larger

| Rule | Datasets | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | BRTISS | DERM | ECOLI | FER | GLASS | IIFI | SEEDS | WINE |
| | Accuracy | | | | | | | |
| HMD | 71.67 | 96.96 | 87.75 | 97.48 | 63.88 | 62.50 | 92.85 | 97.79 |
| LBD | 71.58 | 97.23 | 87.76 | 97.60 | 67.87 | 67.33 | 93.33 | 98.33 |
| SHI | 84.34 | 82.74 | 88.06 | 92.80 | 62.66 | 56.00 | 90.95 | 74,79 |
| RBS | 90.43 | 99.45 | 88.98 | 98.97 | 77.11 | 68.00 | 93.81 | 98.87 |
| | G mean | | | | | | | |
| HMD | 40.58 | 96.60 | 80.40 | 96.98 | 23.34 | 56.12 | 92.72 | 97.52 |
| LBD | 40.73 | 96.83 | 87.86 | 97.24 | 36.62 | 64.89 | 93.12 | 98.17 |
| SHI | 80.00 | 80.00 | 81.62 | 80.00 | 41.18 | 25.82 | 79,76 | 64,72 |
| RBS | 80.00 | 99.47 | 79.06 | 99.08 | 73.36 | 65.40 | 93.75 | 98.75 |
| | F measure | | | | | | | |
| HMD | 69.32 | 97.46 | 80.90 | 94.91 | 61.52 | 61.86 | 92.83 | 97.86 |
| LBD | 71.70 | 97.46 | 84.25 | 94.82 | 60.78 | 67.44 | 93.25 | 98.28 |
| SHI | 89.43 | 80.60 | 67.72 | 81.45 | 64.61 | 56.52 | 72.00 | 77.00 |
| RBS | 89.42 | 98.49 | 85.76 | 96.49 | 72.02 | 67.34 | 93.75 | 98.78 |

Table 5.1.: Support Vector Machine results in term of ACC (top), GACC(middle) and $F\,measure$ (bottom), using HMD, LBD, SHI and RBS reconstruction rules.

| Rule | Datasets | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | BRTISS | DERM | ECOLI | FER | GLASS | IIFI | SEEDS | WINE |
| | Accuracy | | | | | | | |
| HMD | 68.09 | 98.36 | 88.06 | 94.52 | 71.25 | 68.84 | 94.28 | 97.19 |
| LBD | 67.17 | 98.36 | 88.03 | 95.32 | 73.08 | 70.14 | 95.23 | 97.14 |
| SHI | 92.72 | 64.19 | 70.06 | 45.54 | 62.27 | 43.51 | 47.61 | 33.78 |
| RBS | 90.00 | 99.17 | 86.88 | 98.62 | 76.95 | 69.31 | 95.23 | 97.74 |
| | G mean | | | | | | | |
| HMD | 13.86 | 98.40 | 80.77 | 92.86 | 14.94 | 65.57 | 94.20 | 96.98 |
| LBD | 27.73 | 98.40 | 82.02 | 93.57 | 14.88 | 68.67 | 95.14 | 97.30 |
| SHI | 80.00 | 40.79 | 53.66 | 35.15 | 22.84 | 35.00 | 37.03 | 8.91 |
| RBS | 80.00 | 99.10 | 78.78 | 98.11 | 55.68 | 67.27 | 95.14 | 97.53 |
| | F measure | | | | | | | |
| HMD | 62.84 | 98.27 | 83.79 | 93.63 | 63.00 | 66.84 | 94.35 | 97.22 |
| LBD | 63.82 | 98.27 | 84.04 | 94.32 | 63.97 | 69.50 | 95.24 | 97.13 |
| SHI | 82.49 | 49.64 | 62.98 | 40.49 | 52.73 | 39.35 | 42.94 | 26.52 |
| RBS | 77.37 | 98.77 | 81.41 | 98.27 | 71.00 | 68.39 | 95.25 | 97.73 |

Table 5.2.: Multilayer Perceptron results in term of ACC (top), GACC(middle) and $F\,measure$ (bottom) using HMD, LBD, SHI and RBS reconstruction rules.

| Rule | Datasets | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | BRTISS | DERM | ECOLI | FER | GLASS | IIFI | SEEDS | WINE |
| | Accuracy | | | | | | | |
| HMD | 68.91 | 97.54 | 85.94 | 53.87 | 68.77 | 60.02 | 90.95 | 96.09 |
| LBD | 69.72 | 98.09 | 74.05 | 57.64 | 71.18 | 66.34 | 92.85 | 96.87 |
| SHI | 87.22 | 94.79 | 82.82 | 60.59 | 69.36 | 56.66 | 43.81 | 40.53 |
| RBS | 94.45 | 98.08 | 88.12 | 64.70 | 74.68 | 66.50 | 92.38 | 94.96 |
| | G mean | | | | | | | |
| HMD | 27.95 | 95.30 | 74.05 | 0.52 | 0.00 | 54.90 | 90.76 | 96.09 |
| LBD | 44.97 | 97.08 | 78.90 | 11.43 | 0.00 | 64.69 | 92.68 | 96.87 |
| SHI | 77.75 | 94.06 | 59.78 | 51.50 | 37.93 | 54.25 | 33.55 | 0.00 |
| RBS | 92.14 | 97.47 | 80.02 | 53.65 | 12.76 | 65.14 | 92.21 | 95.38 |
| | F measure | | | | | | | |
| HMD | 65.50 | 98.27 | 78.99 | 36.89 | 48.92 | 57.93 | 91.01 | 96.24 |
| LBD | 67.92 | 98.27 | 83.46 | 41.17 | 53.69 | 65.58 | 92.84 | 96.76 |
| SHI | 85.86 | 98.77 | 82.17 | 58.02 | 67.56 | 55.52 | 39.71 | 34.09 |
| RBS | 87.77 | 99.08 | 84.02 | 58.39 | 68.83 | 65.88 | 92.37 | 95.19 |

Table 5.3.: AdaBoost classifier results in term of ACC (top), GACC(middle) and $F\,measure$ (bottom), using HMD, LBD, SHI and RBS reconstruction rules.

performance than other methods on four datasets out of eight, (ii) best performance on BRTISS dataset which are also equal to those provided by SHI method, (iii) lower performance than LBD method in the other two cases. Focusing on results in term of $F\,measure$ we observe that RBS performs better than other methods in five cases out of eight. Considering again datasets with highest degree of imbalance (ECOLI, FER, GLASS and WINE), RBS perform better in 3 out of 4 domains in respect of al the three considered metrics. Consider again the GLASS dataset where RBS shows best performance: in case of ACC, the difference with the second best method (LBD) is 3.77%, in case of GACC the difference with respect to SHIis 32.84% and in case of $F\,measure$ the difference with respect to LBD is 7.03%.

Turning our attention to results achieved using the ADA classifier (Table 5.3), we observe that RBS globally has larger performance than other methods. Indeed, considering both the accuracy and the geometric mean of accuracies, RBS shows the largest values in five out of eight datasets, whereas considering $F\,measure$, it overcomes others methods in six out eight datasets. Focusing on the most imbalanced datasets (ECOLI, FER, GLASS and WINE) RBSperforms better than other methods on: i) 3 out of 4 datasets considering the accuracy; ii) on 2 out of 4 datasets considering GACC; iii) on 3 out 4 cases considering $F\,measure$. Furthermore, it is worth observing the value of GACC on the GLASS dataset: only the use of a reconstruction rule based on Softmax regression permit to attains a value of GACC larger than zero. This means that both HMD and LBD misclassifies all samples of one class, at least. This does not occur for SHI and RBS methods.

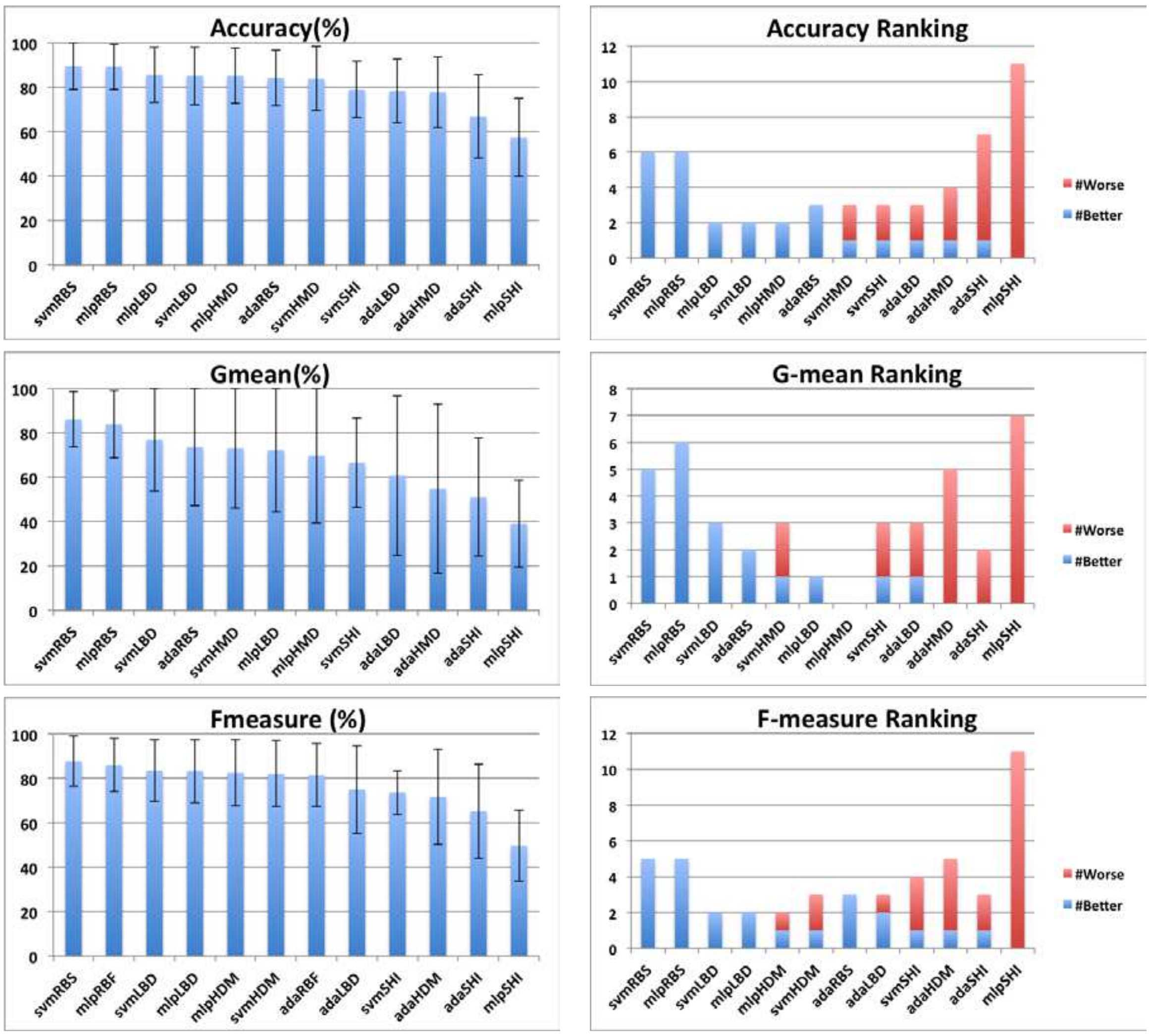

Figure 5.2.: ***Left***. Average $\pm$ Standard deviation for the accuracy (top), Geometric mean of relative accuracies (middle), F-measure (bottom) over the 8 domains for all the classification schemes. ***Right***. Ranking results: number of times each algorithm performed significantly better than the others (blue) or worse (red) according to a Student paired t-test ($p = 0.1$). In each plot, algorithms are ordered from left to right in decreasing average of the metric at hand.

Figure 5.2 presents global comparison between the 12 tested algorithms. Values are reported in terms of average results obtained on the 8 domains on the three metrics (ACC, GACC and $F\,measure$) and in terms of ranking results. In each plot tested algorithms'name is reported concatenating the base classifier's name with the reconstruction method's name. As an example considering the SVM, as the base classifier, and the RBS, as the reconstruction method, resulting algorithm's name is SVMRBF. On the left side of the figure we report the mean values and the standard deviation for each algorithm respect each metrics: ACC (top), GACC (middle), $F\,measure$ (bottom). Algorithms are ordered according to the average value of the metric at hand. We note that, using RBS, a classifier rank first compared to when it is adopted using other reconstruction schemes. It is worth nothing that classification algorithms showing larger performance, on all the metrics, are SVM and MLP ranking first and second, respectively.

To drill down into these general results, we have also computed the global ranking results of each algorithm, recording the number of times each one ranked first, second, third and so on, on the 8 domains. In figure 5.2, we report these results on the right side. To bring statistical validation to these ranking, we performed Student paired t-test comparison for each algorithm against all others ($12x12 = 144$ comparisons), recording those for which we can reject the null hypothesis for level $p = 0.1$, and then clustering the significant differences as to whether they are *better* (blue), or *worse*, of the algorithm at hand. Once again, we notice that learners collect a larger number of significant wins when RBS is used than when other reconstruction rules are adopted.

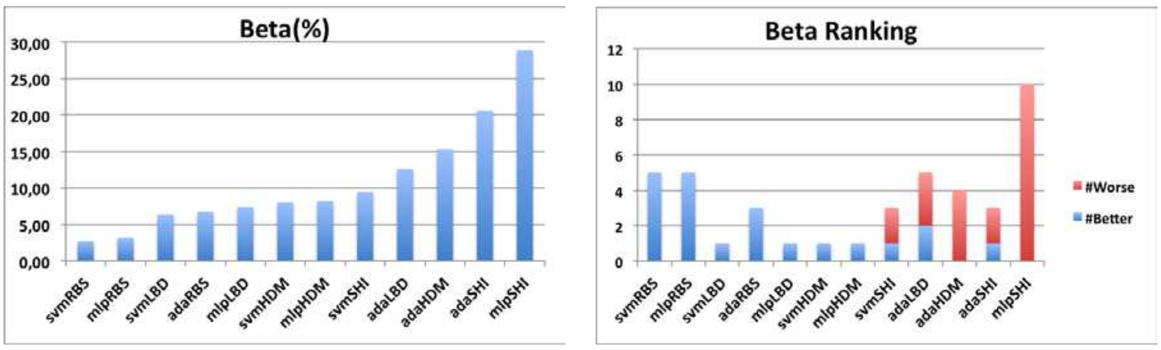

Figure 5.3.: Average $\beta$ values (left) over the 8 domains for all the classification schemes. Ranking results (right): number of times each algorithm performed significantly better than the others (blue) or worse (red) according to a Student paired t-test ($p = 0.1$). In each plot, algorithms are ordered from left to right in decreasing average of $\beta$. Note that lower values of $\beta$ correspond to better classification performance

## 5.4. Discussions

From the analysis of results in tables 5.1, 5.2 and 5.3 and figures 5.2 we prove that RBS method provides performances that are larger that those provided by other methods in the large majority of tests, regardless of the classifier architecture. Furthermore, from rank results in figure 5.2, we observe that, not considering RBS, SHI is not the best performing method. These observations suggest again that the introduction of the reliability in the reconstruction rule provides a significant advantage. Moreover, considering the results achieved on the most imbalanced datasets, namely, ECOLI, FER, GLASS and WINE, we notice that this advantage permits the proposed reconstruction rule to handle imbalanced domains effectively. Indeed the improvements are not limited to the accuracy, but they regard also the GACC and the $F\,measure$. This is important since these two last metrics take in account the behaviour of the classification system with respect to the minority classes (Section 3.3).

As a final issue, we notice that the proposed reconstruction rule provides values of ACC and GACC that are, together, larger than the corresponding ones of other reconstruction rules. This results is summarized in Figure 5.3 in term of the following quantity [125]:

$$\beta = \sqrt{\frac{(1-acc)^2 + (1-g)^2}{2}} \tag{5.4}$$

The measure $\beta$ can be easily interpreted considering the $xy$ plane, where $x$ and $y$ axes correspond to ACC and GACC, respectively. The performancs of a classifier measured as $(acc, g)$ pair get one point in $[0, 1]$x$[0, 1]$ and, hence, $\beta$ ranges in $[0, 1]$. Furthermore, the ideal and the worst performance corresponds to points $(1, 1)$ and $(0, 0)$, respectively. The closer the point representing classifier performance to the ideal point, the more balanced the performance over the classes. The ideal condition implies that $\beta$ is equal to zero and the worst one that $\beta$ is equal to 1. Hence in the figure 5.3 lower is the value better is the corresponding system performance. The notation and plot type of figure 5.3 are the same that we have used in figure 5.2 described above. In this figure we report the average value of $\beta$ for each algorithm on all the datasets (left) and the corresponding rank score (right). Results point out that RBS improves performances of the system, so that SVM, MLP and ADA using RBS rank on the top 4 ranks. We deem that this result is relevant since most of the algorithms coping with class imbalance improve the geometric mean of accuracies harming the global accuracy [125]. Being able to provide larger values of both ACC and GACC implies that RBS improves the recognition ability on the minority class without, or with a small extend, affecting the recognition accuracies on majority classes.

# 6. Boosting Nearest Neighbours for the efficient posterior probability estimation

In this chapter we present an efficient posterior probability estimation by boosting nearest Neighbours. Boosting refers to the iterative combination of classifiers which produces a classifier with reduced true risk (with high probability), while the base classifiers may be weakly accurate [81]. The final, *strong* classifier $h$, satisfies $\mathrm{im}(h) \subseteq \mathbb{R}$. Such an output carries out two levels of information. The simplest one is the sign of the output. This discrete value is sufficient to classify an unknown observation $\boldsymbol{x}$: $h(\boldsymbol{x})$ predicts that $\boldsymbol{x}$ belongs to a class of interest iff it is positive. The most popular boosting results typically rely on this sole information [98, 114, 115] (and many others). The second level is the real value itself, which carries out as additional information a magnitude which can be interpreted, applying some suitable transformation, as a "confidence" or reliability in the classification. This continuous information may be fit into a link function $f : \mathbb{R} \rightarrow [0, 1]$ to estimate conditional class probabilities, thus lifting the scope of boosting to that of Bayes decision rule [54]:

$$\hat{\mathbf{Pr}}[y = 1|\boldsymbol{x}] \quad = \quad f(h(\boldsymbol{x})) \ . \tag{6.1}$$

To date, estimating posteriors with boosting has not met the same success as predicting (discrete) labels. It is widely believed that boosting and conditional class probability estimation are, up to a large extent, in conflict with each other, as boosting iteratively improves classification at the price of progressively overfitting posteriors [19, 54]. Experimentally, limiting overfitting is usually obtained by tuning the algorithms towards early stopping [18].

We analyse, in the light of this problem, a recent algorithm has been proposed to leverage the famed nearest neighbor ( $NN_k$) rules [105], UNN. This algorithm, UNN, fits real-valued coefficients for examples in order to minimize a surrogate risk [12, 98]. These leveraging coefficients are used to balance the votes in the final $k$- $NN_k$ rule. It is proven that, as the number of iterations $T \rightarrow \infty$, UNN achieves the global optimum of the surrogate risk at hand for a wide class of surrogates called strictly convex surrogates [99, 98].

Perhaps the simplest road towards computing the conditional class probabilities for each class $c$, also called (estimated) posteriors estimators consists in adopting a OpC decomposition approach. In such a way we have $C$ problems with corresponding sample $\mathcal{S}^{(c)} = \{(\boldsymbol{x}_i, y_{ic}), i = 1, 2, ..., m\}$. For each of these problems, we learn from $\mathcal{S}$ a classifier $h : \mathcal{O} \rightarrow \mathbb{R}$ out of which we may accurately compute (6.1), typically with $\hat{p}_c(\boldsymbol{x}) = f(h(\boldsymbol{x}))$ for some relevant function $f$.

There exists a convenient approach to carry out this path as a whole, for each class $c =$

| | $\psi$ | $f(x)$ | $\phi$ |
|---|---|---|---|
| A | $(1-x)^2$ | $\frac{1}{2}(1+x)$ | $x(1-x)$ |
| B | $\log_2(1+\exp(-x))$ | $[1+\exp(-x)]^{-1}$ | $-x\ln x$ $-(1-x)\ln(1-x)$ |
| C | $\log_2(1+2^{-x})$ | $[1+2^{-x}]^{-1}$ | $-x\log_2 x$ $-(1-x)\log_2(1-x)$ |
| D | $-x+\sqrt{1+x^2}$ | $\frac{1}{2}\left(1+\frac{x}{\sqrt{1+x^2}}\right)$ | $\sqrt{x(1-x)}$ |
| E | $\frac{1}{2}x(\mathrm{sign}(x)-1)$ | $\begin{cases} 1 & \text{if} \quad x>0 \\ 0 & \text{if} \quad x<0 \end{cases}$ | $2\min\{x, 1-x\}$ |
| F | $\exp(-x)$ | $[1+\exp(-2x)]^{-1}$ | N/A |
| G | $\left(1+\frac{1-\alpha^2}{4}x\right)^{-\frac{1+\alpha}{1-\alpha}}$ | $\left[1+\left(\frac{4-(1-\alpha^2)x}{4+(1-\alpha^2)x}\right)^{\frac{2}{1-\alpha}}\right]^{-1}$ | N/A |

Table 6.1.: Examples of surrogates $\psi$ (Throughout the work, we let $\ln$ denote the base-$e$ logarithm, and $\log_z(x) \doteq \ln(x)/\ln(z)$ denote the base-$z$ logarithm). From top to bottom, the losses are known as: squared loss, (normalized) logistic loss, binary logistic loss, Matsushita loss [99, 98], linear Hinge loss, exponential loss, Amari's $\alpha$-loss, for $\alpha \in (-1,1)$ [98]. Strictly convex losses are A, B, C, D, F, G. Balanced convex losses are A, B, C, D (E corresponds to a limit behavior of balanced convex losses [98]). For each $\psi$, we give the corresponding estimators $\hat{p}_c(\boldsymbol{x}) = f(h(\boldsymbol{x}))$. (Theorem A.2 and Eqs (A.6, A.8) below: replace $x$ in $f(x)$ by $h_{\mathrm{opt}}(\boldsymbol{x})$), and if they are balanced convex losses, the corresponding concave signature $\phi$ (See text for details).

$1, 2, ..., C$: learn $h$ by minimizing a *surrogate risk* over $\mathcal{S}$ [12, 98, 99]. A surrogate risk has general expression:

$$\varepsilon^{\psi}_{\mathcal{S}}(h, c) \quad \doteq \quad \frac{1}{m}\sum_{i=1}^{m}\psi(y_{ic}h(\boldsymbol{x})) \ , \tag{6.2}$$

for some function $\psi$ that we call a *surrogate loss*. Quantity $y_{ic}h(\boldsymbol{x}) \in \mathbb{R}$ is called the *edge* of classifier $\boldsymbol{h}$ on example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ for class $c$. The demonstration that exist a subclass of surrogate losses, whose minimization brings simple and efficient estimators for Bayes (true) posteriors ($\hat{p}_c(\boldsymbol{x}) \doteq \hat{\mathbf{Pr}}[y_c = 1|\boldsymbol{x}]$ ), can be found by interested reader in the appendix A.

In the following of this section we show explicit convergence rates towards these estimators for UNN, for any such surrogate loss, under a Weak Learning Assumption which parallels that of classical boosting results. We provide also experiments and comparisons on synthetic and real datasets. displaying that boosting nearest neighbours brings very good results from the conditional class probabilities estimation standpoint, *without* the over fitting problem of classical boosting approaches.

**Algorithm 2:** Algorithm UNIVERSAL NEAREST NEIGHBORS, UNN($\mathcal{S}, \psi, k$)

```
Input: $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m,\ \boldsymbol{x}_i \in \mathcal{O},\ \boldsymbol{y}_i \in \{-1, 1\}^C\}$, $\psi$ strictly convex loss (Definition A.1), $k \in \mathbb{N}_*$;
Let $\boldsymbol{\alpha}_j \leftarrow \boldsymbol{0}, \forall j = 1, 2, ..., m$;
for $c = 1, 2, ..., C$ do
    Let $\boldsymbol{w} \leftarrow -\nabla_\psi(0)\boldsymbol{1}$;
    for $t = 1, 2, ..., T$ do
        [I.0] Let $j \leftarrow$ WIC($\mathcal{S}, \boldsymbol{w}$);
        [I.1] Let $\delta_j \in \mathbb{R}$ solution of:
            $$\sum_{i:j\sim_k i} y_{ic} y_{jc} \nabla_\psi \left(\delta_j y_{ic} y_{jc} + \nabla_\psi^{-1}(-w_i)\right) = 0 \ ; \qquad (6.3)$$
        [I.2] $\forall i : j \sim_k i$, let
            $$w_i \leftarrow -\nabla_\psi \left(\delta_j y_{ic} y_{jc} + \nabla_\psi^{-1}(-w_i)\right) \ , \qquad (6.4)$$
        [I.3] Let $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$;
Output: $\mathcal{H}(\boldsymbol{x}) \doteq \sum_{j\sim_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j$
```

## 6.1. Leveraging and boosting Nearest Neighbors

The nearest neighbor rule belongs to the oldest, simplest and most widely studied classification algorithms [28, 38]. We denote by $k\mathrm{NN}(\boldsymbol{x})$ the set of the $k$-nearest neighbors (with integer constant $k > 0$) of an example $(\boldsymbol{x}, \boldsymbol{y})$ in set $\mathcal{S}$ with respect to a non-negative real-valued "distance" function. This function is defined on domain $\mathcal{O}$ and measures how much two observations differ from each other. This dissimilarity function thus may not necessarily satisfy the triangle inequality of metrics. For the sake of readability, we let $j \sim_k \boldsymbol{x}$ denote the assertion that example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ belongs to $k\mathrm{NN}(\boldsymbol{x})$. We shall abbreviate $j \sim_k \boldsymbol{x}_i$ by $j \sim_k i$. To classify an observation $\boldsymbol{x} \in \mathcal{O}$, the $k$- $NN_k$ rule $\mathcal{H}$ over $\mathcal{S}$ computes the sum of class vectors of its nearest neighbors, that is: $\mathcal{H}(\boldsymbol{x}) = \sum_{j\sim_k \boldsymbol{x}} \boldsymbol{1} \circ \boldsymbol{y}_j$, where $\circ$ is the Hadamard product. $\mathcal{H}$ predicts that $\boldsymbol{x}$ belongs to each class whose corresponding coordinate in the final vector is positive. A *leveraged* $k$- $NN_k$ rule is a generalization of this to:

$$\mathcal{H}(\boldsymbol{x}) \ = \ \sum_{j\sim_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j \ , \qquad (6.5)$$

where $\boldsymbol{\alpha}_j \in \mathbb{R}^C$ is a leveraging vector for the classes in $\boldsymbol{y}_j$. Leveraging approaches to nearest neighbors are not new [117, 118], yet to the best of our knowledge no convergence results or rates were known, at least until the algorithm UNN [105]. Algorithm 2 gives a simplified version of the UNN algorithm of [105] which learns a leveraged $k$- $NN_k$. Oracle WIC($\mathcal{S}, \boldsymbol{w}$) is the analogous for $NN_k$ of the classical weak learners for boosting: it takes learning sample $\mathcal{S}$ and weights $\boldsymbol{w}$ over $\mathcal{S}$, and returns the index of some example in $\mathcal{S}$ which is to be leveraged. [105] prove that for any strictly convex loss $\psi$, UNN converges to the global optimum of the surrogate risk at hand. However, they prove boosting-compliant convergence rates only for the exponential loss. For all other strictly convex losses, there is no insight on the rates with which UNN may converge towards the optimum of the surrogate risk at hand. We now provide such explicit convergence rates under the following *Weak Learning Assumption*:

**WLA**: There exist some $\vartheta > 0, \varrho > 0$ such that, given any $k \in \mathbb{N}_*$, $c = 1, 2, ..., C$ and any distribution $\boldsymbol{w}$ over $\mathcal{S}$, the weak index chooser oracle WIC returns an index $j$ such that the following two statements hold:

(i) $\mathbf{Pr}_{\boldsymbol{w}}[j \sim_k i] \geq \varrho$;

(ii) $\mathbf{Pr}_{\boldsymbol{w}}[y_{jc} \neq y_{ic} | j \sim_k i] \leq 1/2 - \vartheta$ or $\mathbf{Pr}_{\boldsymbol{w}}[y_{jc} \neq y_{ic} | j \sim_k i] \geq 1/2 + \vartheta$.

Requirement (i) is a weak *coverage* requirement, which "encourages" WIC to choose indexes in dense regions of $\mathcal{S}$. Before studying the boosting abilities of UNN, we focus again on surrogate risks. So far, the surrogate risk (6.2) has been evaluated with respect to a single class. In a multiclass multilabel setting, we may compute the *total* surrogate risk over all classes as:

$$\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H}) \doteq \frac{1}{C}\sum_{c=1}^{C} \varepsilon_{\mathcal{S}}^{\psi}(h_c, c) , \tag{6.6}$$

where $\mathcal{H}$ is the set of all $C$ classifiers $h_1, h_2, ..., h_C$ that have been trained to minimize each $\varepsilon_{\mathcal{S}}^{\psi}(., c), c = 1, 2, ..., C$. We split classifiers just for convenience in the analysis: if one trains a single classifier $H : \mathcal{O} \times \{1, 2, ..., C\} \rightarrow \mathbb{R}$ like for example [115], then we define $h_c$ to be $H$ in which the second input coordinate is fixed to be $c$. Minimizing the total surrogate risk is not only efficient to estimate posteriors (Appendix A.2): it is also useful to reduce the error in label prediction, as the total surrogate risk is an upperbound for the *Hamming risk* [115]: $\varepsilon_{\mathcal{S}}^{\mathrm{H}}(\mathcal{H}) \doteq (1/(mC)) \sum_{c=1}^{C} \sum_{i=1}^{m} \mathrm{I}[y_{ic} h_c(\boldsymbol{x}_i) < 0]$, where $\mathrm{I}[.]$ denotes the indicator variable. It is indeed not hard to check that for any strictly convex surrogate loss $\psi$, we have $\varepsilon_{\mathcal{S}}^{\mathrm{H}}(\mathcal{H}) \leq (1/\psi(0)) \times \varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H})$. We are left with the following question about UNN:

"are there sufficient conditions on the surrogate loss $\psi$ that guarantee, under the sole **WLA**, a *convergence rate* towards the optimum of (6.6) with UNN ?"

We give a positive answer to this question when the surrogate loss meets the following smoothness requirement.

**definition** [78] $\psi$ is said to be $\omega$ strongly smooth iff there exists some $\omega > 0$ such that, for all $x, x' \in \mathrm{int}(\mathrm{dom}(\psi))$, $D_{\psi}(x' \| x) \leq \frac{\omega}{2}(x' - x)^2$, where

$$D_{\psi}(x' \| x) \doteq \psi(x') - \psi(x) - (x' - x)\nabla_{\psi}(x) \tag{6.7}$$

denotes the Bregman divergence with generator $\psi$ [98].

Denote $n_j \doteq |\{i : j \sim_k i\}|$ the number of examples in $\mathcal{S}$ of which $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ is a nearest neighbor, and $n_* \doteq \max_j n_j$. Denote also $\mathcal{H}_{\mathrm{opt}}$ the leveraged $k$- $NN_k$ which minimizes $\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H})$; it corresponds to the set of classifiers $\hat{h}_{\mathrm{opt}}$ of Appendix A.2 that would minimize (6.2) over each class. We are now ready to state our main result (remark that $\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H}_{\mathrm{opt}}) \leq \psi(0)$).

**theorem** Suppose (**WLA**) holds and choose as $\psi$ is any $\omega$ strongly smooth, strictly convex loss. Then for any fixed $\tau \in [\varepsilon^{\psi}_{\mathrm{S}}(\mathcal{H}_{\mathrm{opt}}), \psi(0)]$, UNN has fit a leveraged $k$- $NN_k$ classifier $\mathcal{H}$ satisfying $\varepsilon^{\psi}_{\mathrm{S}}(\mathcal{H}) \leq \tau$ provided the number of boosting iterations $T$ in the inner loop satisfies:

$$T \quad \geq \quad \frac{(\psi(0) - \tau)\omega m n_*}{2\vartheta^2 \varrho^2} \ . \tag{6.8}$$

Theorem proof can be found in Appendix B.

Appendix A.2 has underlined the importance of balanced convex losses in obtaining simple efficient estimators for conditional class probabilities. Coupled with Theorem 6.1, we now show that UNN may be a fast approach to obtain such estimators.

**corollary** Consider any permissible $\phi$ that has been scaled without loss of generality so that $\phi(1/2) = 1$, $\phi(0) = \phi(1) = 0$. Then for the corresponding balanced convex loss $\psi = \psi_\phi$ and under the **WLA**, picking

$$T \quad > \quad \frac{m n_*}{2\vartheta^2 \varrho^2 \min_{x\in(0,1)} \left|\frac{\partial^2 \phi}{\partial x^2}\right|} \tag{6.9}$$

in the inner loop of UNN, for each $c = 1, 2, ..., C$, guarantees to yield an optimal leveraged $k$-$NN_k$ $\mathcal{H}$, satisfying $\varepsilon^{\psi}_{\mathrm{S}}(\mathcal{H}) = \varepsilon^{\psi}_{\mathrm{S}}(\mathcal{H}_{\mathrm{opt}})$. This leveraged $k$- $NN_k$ yields efficient estimators for conditional class probabilities, for each class, by computing:

$$\hat{p}_c(\boldsymbol{x}) \quad = \quad \nabla^{-1}_{\tilde{\phi}}(h_c(\boldsymbol{x})) \ . \tag{6.10}$$

(Proof omitted) For the most popular permissible functions (Table 6.1), quantity $\min_{x\in(0,1)} \left|\frac{\partial^2 \phi}{\partial x^2}\right|$ does not take too small value: its values are respectively $8$, $4/\ln 2$, $4$ for the permissible functions corresponding to the squared loss, logistic loss, Matsushita loss. Hence, in these cases, the bound for $T$ in (6.9) is not significantly affected by this term.

## 6.2. Experiments

In this section we present tests performed in order to validate our proposal. We perform three different kinds of tests. The first one, performed on simulated datasets, aim at evaluating the goodness-of-fit of the posterior estimator. The second one tests UNN against SVM on the task of classify challenging SUN computer vision database. The last one, performed on heterogeneous datasets, evaluates the benefit of using posterior probabilities to set the final decision.

We have tested three flavors of UNN: with the exponential loss (F in Table 6.1), the logistic loss (B in Table 6.1) and Matsushita's loss (D in Table 6.1). All three are respectively referred to as UNN(exp), UNN(log) and UNN(Mat). It is the first time this last flavor is tested, even from the classification standpoint. For all these algorithms, we compute the estimation of posteriors as follows: we use (A.8) for UNN(exp), (6.10) for UNN(log) and UNN(Mat). Leveraging

| | $\delta_{jc}$, see (6.11) | $g : w_i \leftarrow g(w_i)$ |
|---|---|---|
| A | $2W_{jc} - 1$ | $w_i - 2\delta_{jc}y_{ic}y_{jc}$ |
| B | $\ln \frac{W_{jc}}{1-W_{jc}}$ | $\frac{w_i}{w_i \ln 2+(1-w_i \ln 2)\times\exp(\delta_{jc}y_{ic}y_{jc})}$ |
| C | $\log_2 \frac{W_{jc}}{1-W_{jc}}$ | $\frac{w_i}{w_i+(1-w_i)\times 2^{\delta_{jc}y_{ic}y_{jc}}}$ |
| D | $\frac{2W_{jc}-1}{2\sqrt{W_{jc}(1-W_{jc})}}$ | $1 - \frac{1-w_i+\sqrt{w_i(2-w_i)}\delta_{jc}y_{ic}y_{jc}}{\sqrt{1+\delta_{jc}^2 w_i(2-w_i)+2(1-w_i)\sqrt{w_i(2-w_i)}\delta_{jc}y_{ic}y_{jc}}}$ |
| E | N/A | N/A |
| F | $\frac{1}{2}\ln \frac{W_{jc}}{1-W_{jc}}$ | $\exp(-\delta_{jc}y_{ic}y_{jc})$ |
| G | $\frac{4}{1-\alpha^2}\left(\frac{(W_{jc})^{\frac{2}{1-\alpha}}-(1-W_{jc})^{\frac{2}{1-\alpha}}}{(W_{jc})^{\frac{2}{1-\alpha}}+(1-W_{jc})^{\frac{2}{1-\alpha}}}\right)$ | $\frac{4}{1-\alpha^2}\times\left(\frac{1-\alpha^2}{4}\delta_{jc}y_{ic}y_{jc}+\left(\frac{1+\alpha}{2\sqrt{w_i}}\right)^{1-\alpha}\right)^{-\frac{2}{1-\alpha}}$ |

Table 6.2.: Computation of $\delta_{jc}$ and the weight update rule of our implementation of UNN, for the strictly convex losses of Table 6.1. UNN leverages example $j$ for class $c$, and the weight update is that of example $i$ (See text for details and notations).

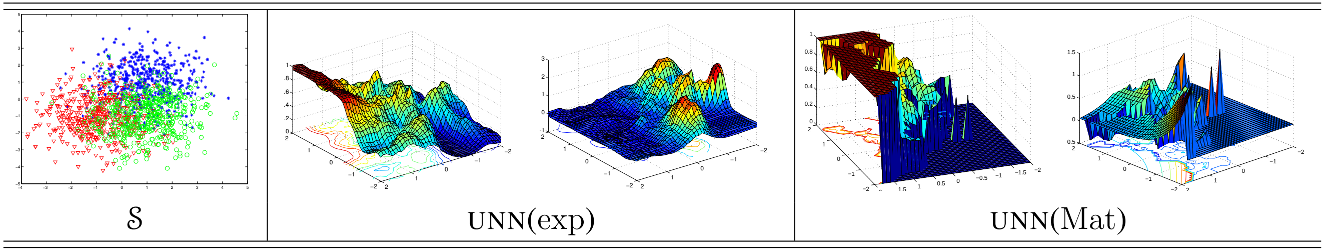

Figure 6.1.: From left to right: example of simulated dataset with $\sigma = 1.1$; the estimated posterior for class 1 obtained by UNN(exp); the corresponding gridwise KL divergence for class 1; the estimated posterior for class 1 obtained by UNN(Mat); the corresponding gridwise KL divergence for class 1 (see (6.13) and text for details).

coefficients estimation and weights update strategies for these three method are reported in the following paragraph.

**Computing leveraging coefficients and weights update** Fix for short $\mathcal{S}_{jb}^{(c)} \doteq \{i : j \sim_k i \wedge y_{ic} = by_{jc}\}$ for $b \in \{+,-\}$. (6.3) may be simplified as $\sum_{i\in\mathcal{S}_{j+}^{(c)}} \nabla_\psi\left(\delta + \nabla_\psi^{-1}(-w_i)\right) = \sum_{i\in\mathcal{S}_{j-}^{(c)}} \nabla_\psi\left(-\delta + \nabla_\psi^{-1}(-w_i)\right)$. There is no closed form solution to this equation in the general case. While it can be simply approximated with dichotomic search, it buys significant computation time, as this approximation has to be performed for each couple $(c,t)$. We tested a much faster alternative which produces results that are in general experimentally quite competitive, consisting in solving instead: $\sum_{i\in\mathcal{S}_{j+}^{(c)}} w_i\nabla_\psi(\delta) = \sum_{i\in\mathcal{S}_{j-}^{(c)}} w_i\nabla_\psi(-\delta)$. We get equivalently that $\delta$ satisfies:

$$\frac{\nabla_\psi(-\delta)}{\nabla_\psi(\delta)} = \frac{W_{jc}}{1-W_{jc}}\ , \tag{6.11}$$

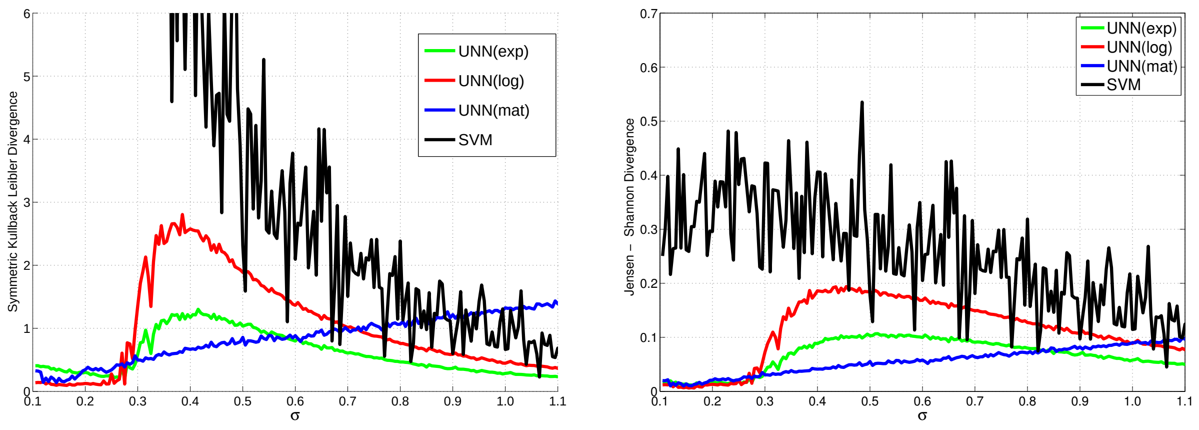

Figure 6.2.: Average Symmetric KL-divergence (left) and JensenShannon divergence (right) as a function of $\sigma$ on simulated datasets, for UNN(exp), UNN(log), UNN(Mat) (left, $k = 10$) and SVM .

with $W_{jc} \doteq (\sum_{i \in \mathcal{S}^{(c)}_{j+}} w_i)/(\sum_{i \in \mathcal{S}^{(c)}_{j+}} w_i + \sum_{i \in \mathcal{S}^{(c)}_{j-}} w_i)$. Remark the similarity with (A.5). Table 6.2 gives the corresponding expressions for $\delta$ and the weight updates.

### 6.2.1. Results on simulated data

In order to evaluate the goodness-of-fit of the proposed posterior probability estimator, we perform tests using simulated data and adopting specific performances metrics.

#### Datasets

We crafted a general domain consisting of $C = 3$ equiprobable classes, each of which follows a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \sigma \mathrm{I})$, for $\sigma \in [0.1, 1.1]$ with steps of $0.005$, and $\boldsymbol{\mu}$ remains the same. For each value of $\sigma$, we compute the average over ten simulations, each of which consists of 1500 training examples and 4500 testing examples. We get overall several thousands datasets, on which all algorithms are tested.

#### Classifier

The competitor that we chose to validate UNN posterior estimation performance is the SVM. On synthetic datasets SVM performs equally using both linear and radial basis function kernel. Therefore, in the following we indicate simply with SVM the linear Support Vector Machine. Values of regularization parameter $C$ is selected within $[1, 10^4]$ and $[10^{-4}, 10]$, adopting a $\log$ scale to sample the two intervals. The value of each parameter is selected according to average performance estimated by five fold cross-validation on a validation set. For SVM, we use the method of [107], which, given a SVM output $f$ for class $c$, forms the posterior:

$$\hat{p}_c(\boldsymbol{x}) \doteq \frac{1}{1 + \exp(af(\boldsymbol{x}) + b)} , \tag{6.12}$$

| | $k$ | UNN(exp) | UNN(log) | UNN(Mat) | SVM |
|---|---|---|---|---|---|
| $Symm\hat{D}_{\mathrm{KL}}$ | 10 | 0.599 | 1.029 | 0.848 | 3.533 |
| | 20 | 0.372 | 0.760 | 0.687 | |
| | 30 | 0.293 | 0.610 | 0.646 | |
| | 40 | 0.254 | 0.534 | 0.632 | |
| $\hat{D}_{\mathrm{JS}}$ | 10 | 0.067 | 0.113 | 0.0562 | 0.256 |
| | 20 | 0.045 | 0.086 | 0.045 | |
| | 30 | 0.036 | 0.072 | 0.043 | |
| | 40 | 0.032 | 0.065 | 0.043 | |
| F-measure | 10 | 90.32 | 89.59 | 90.58 | 91.02 |
| | 20 | 90.62 | 89.53 | 90.81 | |
| | 30 | 90.70 | 89.26 | 90.84 | |
| | 40 | 90.72 | 88.82 | 90.88 | |

Table 6.3.: Average results over simulated data, for UNN(exp), UNN(log), UNN(Mat) with four different values of $k$, and for support vector machines with linear (SVM).

where $a$ and $b$ are estimated by maximizing the log-likelihood of the training sample with a five-fold cross validation.

### Metrics

We use three metrics to evaluate the algorithms. We compute first Kullback-Leibler (KL) divergences between the true and estimated posterior and after their mean obtaining the Symmetric Kullback-Leibler divergences:

$$D_{\mathrm{KL}}(\hat{p}||p) \doteq \sum_c \mathbf{Pr}[c] \int \mathbf{Pr}[\boldsymbol{x}]\hat{p}_c(\boldsymbol{x}) \ln \frac{\hat{p}_c(\boldsymbol{x})}{p_c(\boldsymbol{x})} \mathrm{d}\mu \quad , \quad D_{\mathrm{KL}}(p||\hat{p}) \doteq \sum_c \mathbf{Pr}[c] \int \mathbf{Pr}[\boldsymbol{x}]\, p_c(\boldsymbol{x}) \ln \frac{p_c(\boldsymbol{x})}{\hat{p}_c(\boldsymbol{x})} \mathrm{d}\mu \tag{6.13}$$

$$SymmD_{\mathrm{KL}}(\hat{p}||p) \doteq \frac{1}{2}(D_{\mathrm{KL}}(\hat{p}||p) + D_{\mathrm{KL}}(p||\hat{p})) \tag{6.14}$$

and also we compute JensenShannon (JS) divergence:

$$D_{\mathrm{JS}}(\hat{p}||p) \doteq \frac{1}{2}(D_{\mathrm{KL}}(\hat{p}||q) + D_{\mathrm{KL}}(p||q)) \tag{6.15}$$

where $q$ is the average of the two distribution. Our estimate, $Symm\hat{D}_{\mathrm{KL}}D_{\mathrm{KL}}(\hat{p}||p)$ and $\hat{D}_{\mathrm{JS}}D_{\mathrm{KL}}(\hat{p}||p)$ rely on a simple fine-grained grid approximation of the integral over the subsets of $\mathcal{O}$ of sufficient mass according to $\mu$. We use also the F-$measure$ to evalutate classification performance.

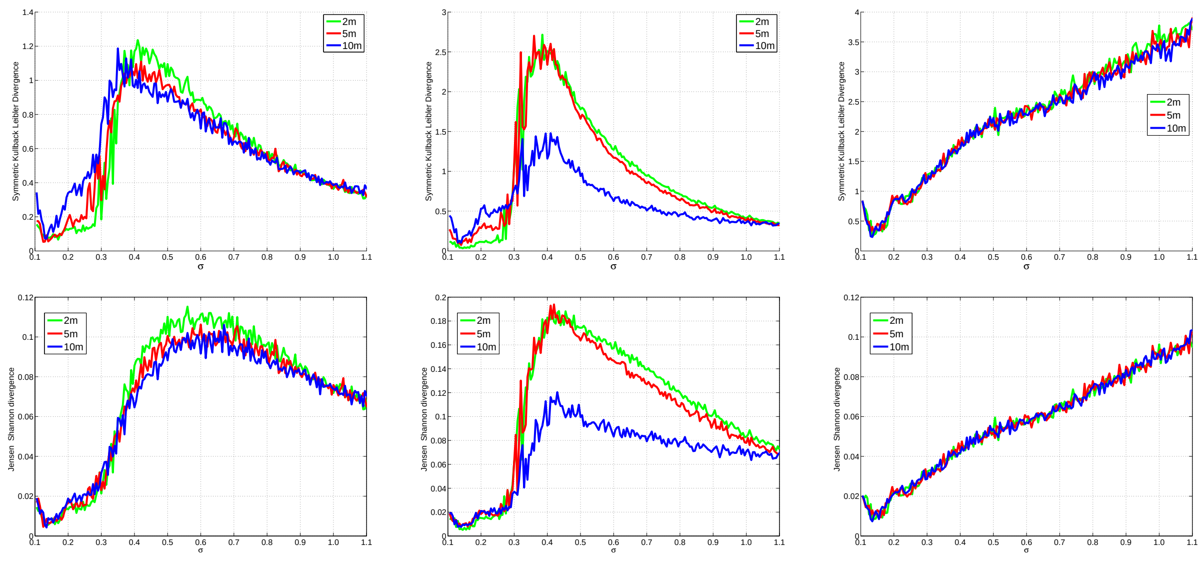

Figure 6.3.: Average symmetric KL-divergence (top) and JensenShannon divergence (bottom) as a function of $\sigma$ on simulated datasets, for UNN$(\exp)$ (left), UNN$(\log)$ (center), UNN$(\mathrm{Mat})$ (right), when the number of boosting iterations $T$ varies in $\{2m, 5m, 10m\}$. The color code in the same on each plot. Notice the differences in the $y$-scale for UNN$(\mathrm{Mat})$ (see text for details).

## Results

Figure 6.1 presents an example of simulated datasets, along with results obtained by UNN$(\exp)$ and UNN$(\mathrm{Mat})$ from the standpoints of the posterior estimates and KL-divergence on the same class. The estimators are rather good, with the largest mismatches (KL-divergence) located near the frontiers of classes. Also, UNN$(\mathrm{Mat})$ tends to outperform UNN$(\exp)$.

Figure 6.2 synthesizes the results from the KL and JS divergence standpoints. Two clear conclusions can be drawn from these results. First, UNN is the clear winner over SVM for the posteriors estimation task. The results of each flavor of UNN is indeed better than those of SVM by orders of magnitude. This is all the more important as the kernels we used are the theoretical kernels of choice given the way we have simulated data. The second conclusion is that UNN$(\mathrm{Mat})$ is the best of all flavors of UNN, a fact also confirmed by the synthetic results of Table 6.3. The KL and JS divergences of UNN$(\mathrm{Mat})$ are in general of minute order with respect to the others. Its behavior (Figure 6.2) is also monotonous: it is predictable that it increases with the degree of overlap between classes, that is, with $\sigma$. From the *classification* standpoint, the average F-measure metrics display a very slight advantage to SVM.

The most important conclusion that can be drawn from the simulated data is shown in Figure 6.3: as the number of boosting iterations $T$ increase, UNN does *not* overfit posteriors in general. The only hitch — not statistically significant — is the case $\sigma > 0.7$ for UNN$(\mathrm{Mat})$, but the differences are of very small order compared to the standard deviations of the KL-divergence.

### 6.2.2. Results on the SUN database domains

We present now results on SUN dataset. We described this dataset in section 3.1. Our interest in this dataset rely on the fact that SUN is one of the most challenging dataset in the field of large scale image classification task.

| | UNN(exp) | | UNN(log) | | UNN(Mat) | | SVM$_l$ | |
|---|---|---|---|---|---|---|---|---|
| | F | R | F | R | F | R | F | R |
| SUN10 | **89.91** | 21.35 | 84.46 | 5.18 | 72.47 | **3.39** | 87.99 | 22.32 |
| SUN20 | **82.82** | 36.64 | 72.34 | 8.51 | 55.46 | **2.51** | 74.60 | 33.25 |
| SUN30 | **73.39** | 49.92 | 61.02 | 14.99 | 40.83 | **5.99** | 62.81 | 39.95 |

Table 6.4.: Area under the $F\, measure$ (in percentage) and (R)ejection rate on the SUN databases. For each database, the best F and R are written in **bold faces**.

#### Datasets

We have crafted, out of SUN computer vision database [149], three datasets, consisting in taking all pictures from the first ten (SUN10), twenty (SUN20) or thirty (SUN30) classes.

#### Classifier

For experiments on SUN dataset we use the same classifiers configuration that we used on simulated data described above.

#### Metrics

On these data, we compute a couple of metrics. First, we compute the F-measure of the classifiers (the harmonic average of precision and recall), based on thresholding the probabilistic output and deciding that $\boldsymbol{x}$ belong to class $c$ iff $\hat{p}_c(\boldsymbol{x}) \geq \kappa$, for varying $\kappa \in ({}^1\!/\!{}_2, 1)$. Second, we compute the rejection rate, that is, the proportion of observations for which $\hat{p}_c(\boldsymbol{x}) < \kappa$. Either we plot couples of curves for the F-measure and rejection rates, or we summarize both metrics by their average values as $\kappa$ ranges through $({}^1\!/\!{}_2, 1)$, which amounts to compute the area under the corresponding curves.

#### Results

Table 6.4 summarizes the results obtained. This table somehow confirms that classification and posterior estimation may be conflicting goals when it comes to boosting [54, 19], as UNN(Mat) achieves very poor results compared to the other algorithms. Furthermore, UNN(exp) appears to the clear winner over all algorithms for this classification task. These results have to be appreciated in the light of the rejection rates: in comparison with the other algorithms, UNN(Mat) rejects a very small proportion of the examples, this indicating a high recall for the algorithm. Figure 6.4 completes the picture by detailing F-measure and rejection

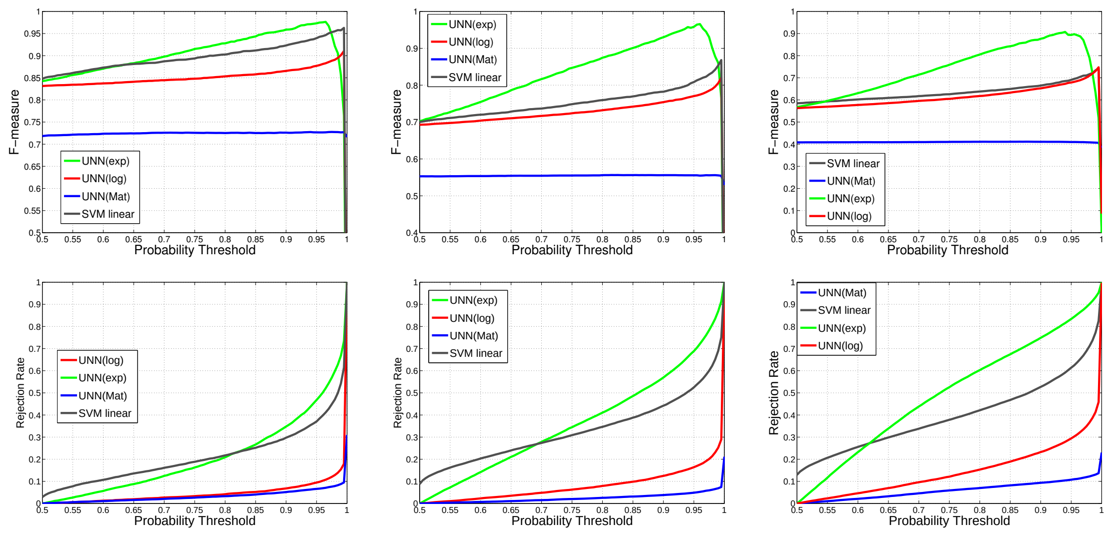

Figure 6.4.: $F\ measure$ (top row) and rejection rates (bottom row) on the SUN domains, with $C = 10$ (left), $C = 20$ (center) and $C = 30$ (right, see Table 6.3 for notations).

rates plots. The F-measure plots clearly display the better performances of UNN(exp) compared to the other algorithms, and the fact that UNN(Mat) displays very stable performances. The rejection rates plots show that UNN(Mat) indeed rejects a very small proportion of examples, even for large values of $\kappa$.

## 6.2.3. Results on heterogeneous domains

We present here results achieved using several heterogeneous datasets belonging to real world domains. In these experiments we verify the ability of the estimated posterior probability to improved classification performances when it is used to set final label. In order to reach this goal we use two different reconstruction rules. The first one estimates the final label according to the *Hamming decoding* (HMD), whereas the second one sets the final label according to the largest probability $\arg\{\max_c(\hat{p}_c(\boldsymbol{x}))\}$ among those computed by dichotomizers that output a crisp label $\hat{y}_c = 1$. When there are no dichotomizers that output a crisp label $\hat{y}_c = 1$, we consider the class associated to $\arg\{\min_c(1 - \hat{p}_c(\boldsymbol{x}))\}$, i.e. the class associated with the lowest probability to predict a $\hat{y}_c = 0$. In the following, this rule is referred to as MDS.

Experiments are performed using a 10-fold cross validation scheme. Each fold is randomly generated maintaining the a-priori distribution of the original dataset. Reported results are computed averaging out the results obtained for each fold.

### Datasets

We used one private and five public datasets, belonging to images classification problems of different biomedical domains (BIOCELLS, DERM, IIFI,YEAST, ICPR$_{BOF}$, ICPR$_{BIF}$). They are characterized by a large variability with respect to the number and type of features, classes and samples, allowing the assessment of classifiers' performances in different conditions. Synthetic data about the used datasets are reported in Table 3.1.

### Classifier

Competitor chosen to asses the proposed approach are the SVM and the kNN. We tested both the SVM with a linear kernel (SVM$_l$) and the SVM with Gaussian kernel (SVMr). For SVM$_l$ values of regularization parameter $C$ is selected within $[1, 10^4]$ and $[10^{-4}, 10]$, adopting a $\log$ scale to sample the two intervals. SVMr's values of regularization parameter $C$ and scaling factor $\sigma$ are selected within $[1, 10^4]$ and $[10^{-4}, 10]$, adopting a $\log$ scale to sample the two intervals. The value of each parameter is tuned using a five fold cross-validation on a validation set. The reliability of a SVM (both linear and Gaussian) classification is estimated as proposed in [107], where the decision value of the classifier is transformed in a posterior probability.

The kNN require no specific set-up. We test values of $k$ equal to $\{1, 3, 5, 7\}$ and choose the value providing the best performances on a validation set according to a five-fold cross validation. We estimate the reliability of each classification act following the method presented in Section 2.4.

### Metrics

As measure of classifier performance, we compute the accuracy and the F-$measure$ described in Section 3.3.

### Results

We report in Table 6.5 the classification performance provided by UNN, SVM (with linear and Gaussian kernel) and kNN classifiers on the six datasets. For each classification task, we report the results obtained using both MDS and HMD reconstruction rules. In order to provide a global comparison among the results, we calculate the relative performance of each experimental configurations with respect to the others (Figure 6.5). We record the number of times each ranked first, second, third and so on, on the 6 domains. To bring statistical validation to these ranking, we performed Student paired t-test comparison for each algorithm against all others ($12x12 = 144$ comparisons), recording those for which we can reject the null hypothesis (that the per-domain difference has zero expectation) for level $p = 0.1$, and then clustering the significant differences as to whether they are in favor, or not, of the algorithm at hand. The six ranks for each classification method are then summed up to give a measure of the overall dominance among the methods in terms of accuracy. An analogous procedure has been carried out in case of F-measure. The analysis of data reported both in Table 6.5 and Figure 6.5 permits us to derive the following three considerations. The first one concerns the comparison between MDS and HMD reconstruction rules. Independently of the classifier and of performance metric considered, the former improves classification results in comparison with the latter over $90\%$. We deem that such performance improvement is mainly due to the fact that MDS rule uses not only predicted crisps labels, as HMD does, but also the corresponding classification reliability. The second consideration focuses on UNN, observing

| Datasets | Metrics (%) | Classifiers | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UNN(exp) | | UNN(log) | | UNN(Mat) | | SVM$_l$ | | SVMr | | kNN | |
| | | MDS | HMD | MDS | HMD | MDS | HMD | MDS | HMD | MDS | HMD | MDS | HMD |
| BIOCELLS | ACC | 87.1 | 87.1 | 86.5 | 86.5 | **87.3** | **87.3** | 74.3 | 74.3 | **87.7** | **87.7** | 85.2 | 85.2 |
| | $F\,measure$ | 77.7 | 77.7 | 76.9 | 76.9 | **78.2** | **78.9** | 66.9 | 66.9 | 76.9 | 76.9 | 75.2 | 75.2 |
| DERM | ACC | 97.5 | **97.6** | 96.5 | 96.4 | **97.7** | 97.1 | 97.1 | 87.7 | 96.9 | 95.5 | 95.9 | 95.5 |
| | $F\,measure$ | **97.3** | **97.3** | 96.1 | 95.7 | **97.3** | **96.6** | 96.5 | 81.2 | **96.6** | 95.1 | 95.4 | 95.2 |
| IIFI | ACC | 69.5 | 69.3 | 68.8 | 69.1 | **70.8** | 68.8 | 67.2 | 66.7 | **71.5** | 67.4 | 70.3 | 68.7 |
| | $F\,measure$ | **69.0** | 68.5 | 68.4 | 68.4 | **70.3** | 68.0 | 66.8 | 64.8 | **70.3** | 65.5 | 69.6 | 67.7 |
| YEAST | ACC | **59.1** | 58.0 | 57.1 | 55.5 | 53.9 | 53.5 | 52.8 | 48.3 | **58.4** | 54.5 | 54.1 | 54.3 |
| | $F\,measure$ | **50.7** | 46.3 | 47.5 | 45.4 | 41.5 | 40.9 | 41.2 | 24.2 | **47.8** | 41.7 | 46.1 | 44.5 |
| ICPR$_{BOF}$ | ACC | **88.1** | 85.3 | **87.1** | 84.6 | 85.9 | 80.5 | 65.4 | 66.0 | 86.3 | 81.6 | 25.1 | 26.6 |
| | $F\,measure$ | **87.4** | 84.9 | 86.2 | 83.3 | **85.8** | 81.1 | 72.3 | 55.1 | 85.2 | 79.8 | 21.5 | 21.2 |
| ICPR$_{BIF}$ | ACC | **95.7** | **95.6** | 94.9 | 95.5 | 95.4 | 94.9 | 91.8 | 89.8 | 95.3 | 94.4 | 95.1 | 93.91 |
| | $F\,measure$ | **95.6** | **95.4** | 94.4 | 95.4 | **95.6** | 95.1 | 90.7 | 85.5 | 95.2 | 94.0 | 94.8 | 93.7 |

Table 6.5.: Average values (%) of accuracy and F-measure of the different classifiers. We mark highest value (blue) and the second one (green) in each row.

that its performance improve using posterior based reconstruction rule. Indeed, MDS scheme equals or improves UNN performance with HMD scheme in 85% of the cases, at least. For instance, focusing on ICPR$_{BOF}$ dataset, MDS improves UNN performance for all the three configurations of 2%, at least, in terms of both accuracy and F-measure. The third observation concerns how UNN performance compares with those provided by other classifiers. From a general point of view, turn our attention to Figure 6.5 where we notice that the value of UNN(exp) rank is larger than the ones of other classifiers. Focusing now on recognition performance we note that UNN classifiers with MDS scheme always overcome performance of SVM$_l$. UNN also overcome kNN results with at least one configuration among the three tested. Comparing performance of UNN with those of SVMr we note that results are quite similar.

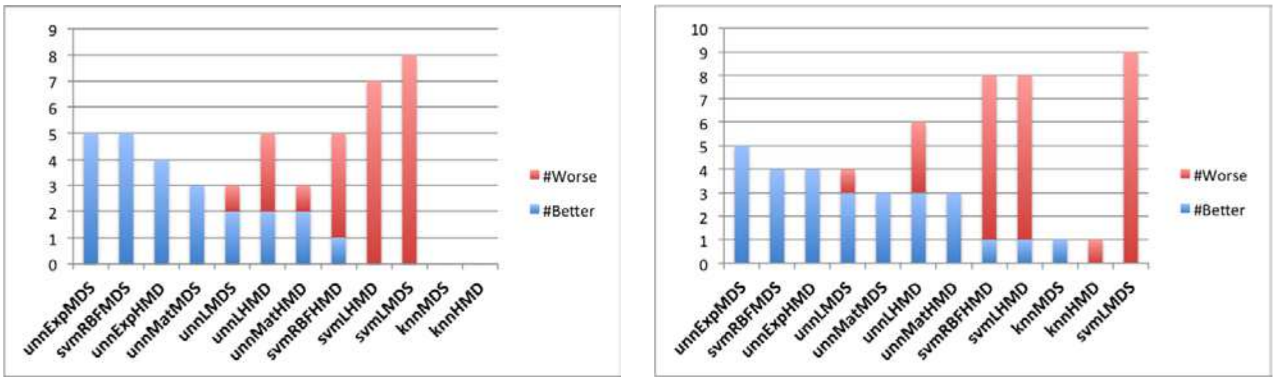

Figure 6.5.: Ranking results: number of times each algorithm performed significantly better than the others (blue) or worse (red) according to a Student paired t-test ($p = 0.1$), for the accuracy (left), Fmeasure (right) over the 6 domains for all the classification schemes. In each plot, algorithms are ordered from left to right in decreasing average of the metric at hand.

## 6.3. Discussions

Boosting algorithms are remarkably simple and efficient from the classification standpoint, and are being used in a rapidly increasing number of domains and problems [18]. In some sense, it would be too bad that such successes be impeded when it comes to posterior estimation [19]. Experimental results display that this estimation is possible, but it necessitates a very fine tuning of the algorithms [18].
The point of our work is that estimating class conditional probabilities may be possible, without such tedious tunings, and sometimes even *without overfitting*, if we boost topological approaches to learning like nearest neighbors. There is a simple explanation to this fact. For any classifier, the conditional class probability estimation for some $\boldsymbol{x}$ in (A.4) is be the same as for any other observation in the vicinity of $\boldsymbol{x}$, where the "vicinity" is to be understood from the *classifier* standpoint. When boosting decision trees, the vicinity of $\boldsymbol{x}$ corresponds to observations classified by the same leaf as $\boldsymbol{x}$. As the number of leaves of the tree increases, the vicinity gets narrowed, which weakens the estimation in (A.4) and thus overfits the corresponding estimated density. Ultimately, linear combinations of such trees, such as those performed in AdaBoost, make such a fine-grained approximation of the local topology of data that the estimators get irreparably confined to the borders of the interval $[0, 1]$ [19]. Nearest neighbors do not have such a drawback, as the set of $k$-nearest neighbors in $\mathcal{S}$ of some observation $\boldsymbol{x}$ spans a region of $\mathcal{O}$ which does not change throughout the iterations. Nearest neighbor rules exploit a topology of data which, under regularity conditions about the true posteriors, also carries out information about these posteriors. For these reasons, nearest neighbors might be a key entry for a reliable estimation of posteriors with boosting. Because of the wealth of "good" surrogates, this opens avenues of research to *learn* the most accurate surrogate on a data-dependent way, such as when it is parameterized (Amari's $\alpha$-loss, see Table 6.1 ).
Furthermore with this contribution we have shown that using posteriors to set the final label improves UNN performances. Indeed on heterogeneous datasets it achieves larger results when using a reconstruction rule based on classifiers' reliability rather than a reconstruction rule based on classifiers' row outputs. In this section we have shown also that thanks to this efficient posterior estimation UNN can compete with SVM, one of the most powerful classifier especially on dataset such as SUN.

We finally want notice that there is, an analytical and computational bottleneck in UNN, as the leveraging coefficients are solutions to non-linear equations with no closed form expression in the general case. Boosting compliant approximations are possible, but in the context of NN rules, they are computationally far too expensive to be performed at each boosting iteration on large datasets. Hence, in appendix C we present "Gentle Nearest Neighbors Boosting that performs adaptive Newton-Raphson steps to minimize any balanced convex surrogate with guaranteed convergence rates avoiding this drawback.

# 7. Conclusions

In this thesis we focused on the issues related to the classification of samples belonging to multiclass imbalanced datasets. To the best of our knowledge, existing works in this field can be divided in two groups. In the first one there are methods tackling directly the polychotomy. In the other one there are approaches that decompose the problem in binary tasks. In the latter case, existing solutions are proposed only at level of the single dichotomies. furthermore, the analysis of the literature show that no attempts exist to solve this problem using reconstruction rules specifically tailored to skewed data.

We therefore aim at designing a reconstruction rule addressing this open issue. We also decided to use the classification reliability into the decision stage. This measure indicates the classifier "confidence" towards its prediction taking into account elements such as dataset noise, borderline samples, outliers, etc. Hence, it owns useful information related to the learners predictions.

Considering the characteristics of the reliability, we designed a novel heuristic reconstruction rule. This rule copes with multiclass imbalance problems using classifiers' reliability in the One-per-Class reconstruction rule. The proposed rule has been compared also with other two well-established reconstruction criteria on a set of benchmark real and artificial datasets, testing four classification architectures. Our results showed that the proposed reconstruction rule provides larger performances than those provided by other criteria. In particular, in several cases it attained values of accuracy, geometric mean of accuracies and $F\,measure$ that were, together, larger than those attained by other rules. Hence, our proposals improved the recognition ability on the minority classes affecting the recognition accuracies on majority classes to a lesser extent than the others. Furthermore, the large number of experiments we carried out showed and that employing reliability in the reconstruction rule permits to achieve larger values of accuracy, geometric mean of accuracies and $F\,measure$ than using only the crisp labels.

Aiming at further exploring the use of reconstruction rules to handle imbalanced datasets, we presented a reconstruction rule in the ECOC decomposition scheme. We considered the outputs of binary classifiers as a new feature vector. Indeed, each dichotomizer output can be transformed in a reliability value and the collection of these values maps the input sample in a new feature space. From this point of view the reconstruction stage is similar to a classification task. Hence, we have proposed an extension of a statistical rule suited for the OpC and PC decomposition scheme in the ECOC case. According to this rule the final label is set choosing the class with the highest posterior probability estimated by the softmax regression. Beside to the extension to a new decomposition scheme, we modified also the rule in order to use classifiers' reliability. Indeed, the original approach uses directly the dichotomizers' soft label without using the reliability. Given eight heterogeneous datasets, our proposal was satisfactory compared with two popular reconstruction rules using three different classification algorithms.

We also provided comparison with the original method [122] extended to the ECOC case. This comparison, together with results obtained against other rules, shows that the use of reliability at the reconstruction stage improves system performance in term of accuracy, geometric mean of accuracy, as well as $F\,measure$. Hence, we can conclude that the rule shows satisfactory performances when it deals with imbalanced domains.

Exploring the issue of reliability and posterior estimation, we noticed that, several methods exist to derive this quantity from classifier soft outputs. In some cases, when the classification function satisfied the requirements of sufficient regularity, the reliability can be estimated in terms of posterior probability. Nevertheless, we noticed also that, up to now, efficient estimators for posterior probability for boosting algorithms have not met enough attention. Indeed, even if boosting algorithms are efficient for classification and are being used in a rapidly increasing number of domains, it is widely believed that boosting and conditional class probability estimation are in conflict with each other. Existing experimental results on this subject show that this estimation is possible, but it necessitates a very fine tuning of the algorithms. On these reasons, we developed an efficient method that permits to estimate class conditional probabilities without such tuning. That is possible boosting topological approaches to learning like nearest neighbors. As shown in Chapter 6 we achieved this result using UNN, which leverages nearest neighbors while minimizing a convex loss. Our contribution is threefold. First, we showed that there exists a subclass of surrogate losses whose minimization brings simple and statistically efficient estimators for Bayes posteriors. Second, we showed explicit convergence rates towards these estimators for UNN, for any such surrogate losses, under a Weak Learning Assumption which parallels that of classical boosting results. Third and last, we provided experiments and comparisons on synthetic and real datasets, including the challenging SUN computer vision database. Results clearly display that boosting nearest neighbors may provide highly accurate estimators, sometimes more than a hundred times more accurate than those of other contenders like support vector machines. It is worth noting that UNN does not over-fit posteriors increasing the number of boosting iteration. This is an interesting results since it is widely believed that boosting and posterior estimation are in conflict with each other, as boosting iteratively improves classification at the price of progressively overfitting posteriors [19, 54]. Furthermore, we observed that using the estimated posterior in the decision making process we improved UNN classification performances.

Summarizing, in this thesis we presented two reconstruction rules based on the classification reliability and an efficient posterior estimator for boosting algorithm by Nearest Neighbors. Results achieved on the proposed reconstruction rules show two main results. The first one is that overall performances of the tested systems improve when reliability-based reconstruction rules are used. The second one is that the proposed rules improve the performance with respect of both the geometric mean of accuracies and $F\,measure$ proving that the rules are suited for skewed domains. Furthermore, in this thesis we showed that the use of classifiers' reliability in reconstruction rule is an useful instrument to improve performance with respect to minority classes.

Future works are directed towards a further exploration of the role of classification reliability in the reconstruction rule. In this work, we extend the [122] in the ECOC scheme proving that the use of reliability improves the performance with respect of the original method. For this reason a future contribution will be to verify if the use of this quantity can improve also the performance of the softmax reconstruction in the original One-per-Class and Pair Wise Coupling schemes. Finally, another future work is provide a full comparisons between all the reconstruction rules suited for skewed data across the decomposition schemes. This future work could show the existence of a rule suited for imbalanced domains independently of the datasets, of the learners, and of the decomposition schemes.

# Bibliography

[1] R. Alejo, J. Sotoca, and G. Casañ. An empirical study for the multi-class imbalance problem with neural networks. *Progress in Pattern Recognition, Image Analysis and Applications*, pages 479–486, 2008.

[2] H. Alhammady and K. Ramamohanarao. The application of emerging patterns for improving the quality of rare-class classification. In *Advances in Knowledge Discovery and Data Mining*, pages 207–211. Springer, 2004.

[3] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001.

[4] S.-I. Amari and H. Nagaoka. *Methods of Information Geometry*. Oxford University Press, 2000.

[5] F. Angiulli. Fast condensed nearest neighbor rule. In *Proceedings of the 22nd international conference on Machine learning*, pages 25–32. ACM, 2005.

[6] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007.

[7] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[8] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *J. of Mach. Learn. Res.*, 6:1705–1749, 2005.

[9] R. Barandela, J. S. Sanchez, V. Garca, and E. Rangel. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3):849–851, 2003.

[10] H. B. Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[11] A.-R. Barron, A. Cohen, W. Dahmen, and R.-A. DeVore. Approximation and learning by greedy algorithms. *Annals of Statistics*, 26:64–94, 2008.

[12] P. Bartlett, M. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the Am. Stat. Assoc.*, 101:138–156, 2006.

[13] G. E. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.

[14] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[15] A. Blum. Random projection, margins, kernels, and feature-selection. In *Subspace, Latent Structure and Feature Selection*, volume 3940 of *LNCS*, pages 52–68. Springer Verlag, 2006.

[16] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proc. of COMP-STAT*, pages 177–186. Springer, 2010.

[17] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic. Training text classifiers with svm on very few positive examples. *Microsoft Research*, 2003.

[18] P. Bühlmann and T. Hothorn. Boosting algorithms: regularization, prediction and model fitting. *Statistical Science*, 22:477–505, 2007.

[19] A. Buja, D. Mease, and A.-J. Wyner. Comment: Boosting algorithms: regularization, prediction and model fitting. *Statistical Science*, 22:506–512, 2007.

[20] P. K. Chan and S. J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *KDD*, volume 1998, pages 164–168, 1998.

[21] K. Chatfield, V.-S. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. pages 1–12, 2011.

[22] N. Chawla and L. Hall. Smote: Synthetic minority over-sampling technique. *J. of Artificial Intelligence Research*, (16):321–357, 2002.

[23] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119. Springer, 2003.

[24] K. Chen, B. Lu, and J. Kwok. Efficient classification of multi-label and imbalanced data using min-max modular classifiers. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 1770–1775. IEEE, 2006.

[25] Y. Chen, X. S. Zhou, and T. S. Huang. One-class svm for learning in image retrieval. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 34–37. IEEE, 2001.

[26] W. W. Cohen. Fast effective rule induction. In *ICML*, volume 95, pages 115–123, 1995.

[27] L. P. Cordella et al. Reliability parameters to improve combination strategies in multi-expert systems. *Pattern Analysis & Applications*, 2(3):205–214, 1999.

[28] T.-M. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. on Information Theory*, 13:21–27, 1967.

[29] L. Cucala, J.-M. Marin, C.-P. Robert, and D.-M. Titterington. A Bayesian reassessment of nearest-neighbor classification. *Journal of the Am. Stat. Assoc.*, 104:263–273, 2009.

[30] R. D'Ambrosio, G. Iannello, and P. Soda. A one-per-class reconstruction rule for class imbalance learning. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 1310–1313. IEEE, 2012.

[31] R. D'Ambrosio, G. Iannello, and P. Soda. Biomedical classification tasks by softmax reconstruction in ecoc framework. In *6th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2013)*, 2013.

[32] R. D'Ambrosio, G. Iannello, and P. Soda. Softmax regression for ecoc reconstruction. In *Image Analysis and Processing–ICIAP 2013*, pages 682–691. Springer Berlin Heidelberg, 2013.

[33] R. D'Ambrosio, R. Nock, W. B. H. Ali, F. Nielsen, and M. Barlaud. Boosting nearest neighbors for the efficient estimation of posteriors. In *Machine Learning and Knowledge Discovery in Databases*, pages 314–329. Springer Berlin Heidelberg, 2012.

[34] R. D'Ambrosio and P. Soda. Polichotomies on imbalanced domains by one-per-class compensated reconstruction rule. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 301–309. Springer Berlin Heidelberg, 2012.

[35] R. D'Ambrosio, P. Soda, and G. Iannello. Automatic facial expression recognition using statistical-like moments. In *ICIAP 2011*, volume 6978 of *LNCS*, pages 585–594. 2011.

[36] C. De Stefano, C. Sansone, and M. Vento. To reject or not to reject: that is the question: an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics–Part C*, 30(1):84–93, February 2000.

[37] M. D. Del Castillo and J. I. Serrano. A multistrategy approach for digital text categorization from imbalanced documents. *ACM SIGKDD Explorations Newsletter*, 6(1):70–79, 2004.

[38] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer, 1996.

[39] T. G. Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[40] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[41] P. Domingos. Metacost: a general method for making classifiers cost-sensitive. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164. ACM, 1999.

[42] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 43–52. ACM, 1999.

[43] C. Drummond, R. C. Holte, et al. C4. 5, class imbalance, and cost sensitivity: why undersampling beats over-sampling. In *Workshop on Learning from Imbalanced Datasets II*, volume 11. Citeseer, 2003.

[44] A. Estabrooks, T. Jo, and N. Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[45] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. Adacost: misclassification cost-sensitive boosting. In *ICML*, pages 97–105. Citeseer, 1999.

[46] A. Fernández, M. del Jesus, and F. Herrera. Multi-class imbalanced data-sets with linguistic fuzzy rule based classification systems based on pairwise learning. *Computational Intelligence for Knowledge-Based Systems Design*, pages 89–98, 2010.

[47] A. Fernández, M. J. del Jesus, and F. Herrera. On the 2-tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data-sets. *Information Sciences*, 180(8):1268–1291, 2010.

[48] A. Fernández, S. García, M. J. del Jesus, and F. Herrera. A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets. *Fuzzy Sets and Systems*, 159(18):2378–2398, 2008.

[49] A. Fernández, S. García, and F. Herrera. Addressing the classification with imbalanced data: Open problems and new challenges on class distribution. In *Hybrid Artificial Intelligent Systems*, pages 1–10. Springer, 2011.

[50] A. Fernández, V. López, M. Galar, M. José del Jesus, and F. Herrera. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, 2013.

[51] P. Foggia et al. *MCS*, chapter On Rejecting Unreliably Classified Patterns, pages 282–291. 2007.

[52] D. Fragoudis, D. Meretakis, and S. Likothanassis. Integrating feature and instance selection for text classification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 501–506. ACM, 2002.

[53] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Machine Learning-International Workshop Then Conference*, pages 148–156. MORGAN KAUFMANN PUBLISHERS, INC., 1996.

[54] J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression : a Statistical View of Boosting. *Annals of Statistics*, 28:337–374, 2000.

[55] J. Fürnkranz. Round robin classification. *The Journal of Machine Learning Research*, 2:721–747, 2002.

[56] S. García, A. Fernández, and F. Herrera. Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems. *Applied Soft Computing*, 9(4):1304–1314, 2009.

[57] S. García and F. Herrera. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary Computation*, 17(3):275–306, 2009.

[58] N. García-Pedrajas and D. Ortiz-Boyer. Boosting $k$-nearest neighbor classifier by means of input space projection. 36(7):10570–10582, 2009.

[59] T. Gneiting and A. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the Am. Stat. Assoc.*, 102:359–378, 2007.

[60] M. Gonen, A. G. Tanugur, and E. Alpaydm. Multiclass posterior probability support vector machines. *Neural Networks, IEEE Transactions on*, 19(1):130–139, 2008.

[61] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[62] P. Grünwald and P. Dawid. Game theory, maximum entropy, minimum discrepancy and robust Bayesian decision theory. *Annals of Statistics*, 32:1367–1433, 2004.

[63] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou. On the class imbalance problem. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, volume 4, pages 192–201. IEEE, 2008.

[64] H. Han, W.-Y. Wang, and B.-H. Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *Advances in intelligent computing*, pages 878–887. Springer, 2005.

[65] S. Han, B. Yuan, and W. Liu. Rare class mining: progress and prospect. In *Pattern Recognition, 2009. CCPR 2009. Chinese Conference on*, pages 1–5. IEEE, 2009.

[66] D. O. Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2002.

[67] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(1):66–75, 1994.

[68] K. Huang, H. Yang, I. King, and M. R. Lyu. Learning classifiers from imbalanced data based on biased minimax probability machine. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–558. IEEE, 2004.

[69] G. Iannello et al. On the use of classification reliability for improving performance of the one-per-class decomposition method. *Data & Knowledge Eng.*, 68:1398–1410, 2009.

[70] N. Japkowicz. The class imbalance problem: Significance and strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI 2000)*, volume 1, pages 111–117. Citeseer, 2000.

[71] N. Japkowicz. Supervised learning with unsupervised output separation. In *International Conference on Artificial Intelligence and Soft Computing*, volume 3, pages 321–325, 2002.

[72] J. Jelonek and J. Stefanowski. Experiments on solving multiclass learning problems by $n^2$ classifier. In *10th European Conference on Machine Learning*, pages 172–177. Springer-Verlag Lecture Notes in Artificial Intelligence, 1998.

[73] T. Jo and N. Japkowicz. Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter*, 6(1):40–49, 2004.

[74] T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM, 2005.

[75] M. V. Joshi, R. C. Agarwal, and V. Kumar. Mining needle in a haystack: classifying rare classes via two-phase rule induction. In *ACM SIGMOD Record*, volume 30, pages 91–102. ACM, 2001.

[76] M. V. Joshi, V. Kumar, and R. C. Agarwal. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 257–264. IEEE, 2001.

[77] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *arXiv preprint cs/9605103*, 1996.

[78] S. Kakade, S. Shalev-Shwartz, and A. Tewari. Applications of strong convexity–strong smoothness duality to learning with matrices. Technical Report CoRR abs/0910.0610, Computing Res. Repository, 2009.

[79] T. Kanade, J. Cohn, and Y. Tian. Comprehensive database for facial expression analysis. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 46–53. IEEE, 2000.

[80] M. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. *Journal of Comp. Syst. Sci.*, 58:109–128, 1999.

[81] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. M.I.T. Press, 1994.

[82] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(3):226–239, 1998.

[83] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[84] S. Kotsiantis and P. Pintelas. Mixture of expert agents for handling imbalanced data sets. *Annals of Mathematics, Computing and Teleinformatics*, 1(1):46–55, 2003.

[85] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *Machine Learning-International Workshop Then Conference*, pages 179–186. Morgan Kaufmann Publishers, Inc., 1997.

[86] S. Kumar, J. Ghosh, and M. M. Crawford. Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis & Applications*, 5(2):210–220, 2002.

[87] L. I. Kuncheva. Using diversity measures for generating error-correcting output codes in classifier ensembles. *Pattern Recognition Letters*, 26(1):83–90, 2005.

[88] L. I. Kuncheva and W. J. Faithfull. Pca feature extraction for change detection in multidimensional unlabelled streaming data. In *21st International Conference on Pattern Recognition*, 2012.

[89] L. Lam and C. Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16(9):945–954, 1995.

[90] T. Liao. Classification of weld flaws with imbalanced class data. *Expert Systems with Applications*, 35(3):1041–1052, 2008.

[91] H. Liu and H. Motoda. On issues of instance selection. *Data Mining and Knowledge Discovery*, 6(2):115–130, 2002.

[92] X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory undersampling for class-imbalance learning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(2):539–550, 2009.

[93] D. MacKay. *The epistemological problem for automata*. Automata Studies, Princeton, NJ: Princeton University Press, 1956.

[94] L. M. Manevitz and M. Yousef. One-class svms for document classification. *The Journal of Machine Learning Research*, 2:139–154, 2002.

[95] D. Marr. A theory for cerebral neocortex. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, pages 161–234, 1970.

[96] F. J. Martínez-Estudillo, P. A. Gutiérrez, C. Hervás, and J. Fernández. Evolutionary learning by a sensitivity-accuracy approach for multi-class problems. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1581–1588, 2008.

[97] U. Müller-Funk, F. Pukelsheim, and H. Witting. On the attainment of the Cramér-Rao bound in $L_r$-differentiable families of distributions. *Annals of Statistics*, pages 1742–1748, 1989.

[98] R. Nock and F. Nielsen. On the efficient minimization of classification-calibrated surrogates. In *NIPS*21*, pages 1201–1208, 2008.

[99] R. Nock and F. Nielsen. Bregman divergences and surrogates for learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(11):2048–2059, 2009.

[100] R. Nock and F. Nielsen. Information-Geometric Lenses for multiple Foci+Contexts interfaces. 2013.

[101] R. Nock, F. Nielsen, and E. Briys. Non-linear book manifolds: learning from associations the dynamic geometry of digital libraries. pages 313–322, 2013.

[102] R. Nock, P. Piro, F. Nielsen, W. B. H. Ali, and M. Barlaud. Boosting $k$-NN for categorization of natural scenes. 100:294–314, 2012.

[103] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid. Towards good practice in large-scale learning for image classification. pages 3482–3489, 2012.

[104] C. Phua, D. Alahakoon, and V. Lee. Minority report in fraud detection: classification of skewed data. *ACM SIGKDD Explorations Newsletter*, 6(1):50–59, 2004.

[105] P. Piro, R. Nock, F. Nielsen, and M. Barlaud. Leveraging $k$-NN for generic classification boosting. *Neurocomputing*, 80:3–9, 2012.

[106] W. B. P. Piro, L. Crescence, D. Giampaglia, O. Ferhat, J. Darcourt, T. Pourcher, and M. Barlaud. A bio-inspired learning and classification method for subcellular localization of a plasma membrane protein. In *VISAPP 2012*, 2012.

[107] J.-C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. pages 61–74, 1999.

[108] F. Provost. Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI 2000 workshop on imbalanced data sets*, 2000.

[109] J. R. Quinlan. *C4.5 : programs for machine learning*. Morgan Kaufmann, 1993.

[110] P. Riddle, R. Segal, and O. Etzioni. Representation design and brute-force induction in a boeing manufacturing domain. *Applied Artificial Intelligence an International Journal*, 8(1):125–147, 1994.

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

[112] A. Rigon, P. Soda, D. Zennaro, G. Iannello, and A. Afeltra. Indirect immunofluorescence in autoimmune diseases: Assessment of digital images for diagnostic purpose. *Cytometry B (Clinical Cytometry)*, 72:472–477, 2007.

[113] C. Sansone, F. Tortorella, and M. Vento. A classification reliability driven reject rule for multi-expert systems. *International journal of pattern recognition and artificial intelligence*, 15(06):885–904, 2001.

[114] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin : a new explanation for the effectiveness of voting methods. *Annals of statistics*, 26:1651–1686, 1998.

[115] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning Journal*, 37:297–336, 1999.

[116] F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[117] M. Sebban, R. Nock, and S. Lallich. Boosting Neighborhood-Based Classifiers. In *Proc. of the 18 $^{th}$ International Conference on Machine Learning*, pages 505–512. Morgan Kauffman, 2001.

[118] M. Sebban, R. Nock, and S. Lallich. Stopping criterion for boosting-based data reduction techniques: from binary to multiclass problems. *J. of Mach. Learn. Res.*, 3:863–885, 2003.

[119] B. Selman, H. Levesque, D. Mitchell, et al. A new method for solving hard satisfiability problems. In *Proceedings of the tenth national conference on Artificial intelligence*, pages 440–446, 1992.

[120] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML'07*, pages 807–814. ACM, 2007.

[121] J. G. Shanahan and N. Roma. Boosting support vector machines for text classification through parameter-free threshold relaxation. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 247–254, New York, NY, USA, 2003. ACM.

[122] Y. Shiraishi and K. Fukumizu. Statistical approaches to combining binary classifiers for multiclass classification. *Neurocomputing*, 74(5):680–688, 2011.

[123] P. Simeone, C. Marrocco, and F. Tortorella. Design of reject rules for ecoc classification systems. *Pattern Recognition*, 45(2):863–875, 2012.

[124] P. Soda. Experiments on solving multiclass recognition tasks in the biological and medical domains. In *IEEE International Conference on Bio-inspired Systems and Signal Processing*, pages 64–71, 2008.

[125] P. Soda. A multi-objective optimisation approach for class-imbalance learning. *Pattern Recognition*, 44:1801–1810, 2011.

[126] P. Soda and G. Iannello. Decomposition methods and learning approaches for imbalanced dataset: An experimental integration. In *20th International Conference on Pattern Recognition*, pages 3117–3120, 2010.

[127] P. Soda, G. Iannello, and M. Vento. A multiple experts system for classifying fluorescence intensity in antinuclear autoantibodies analysis. *Pattern Analysis & Applications*, 12(3):215–226, 2009.

[128] J. Stefanowski. Multiple and hybrid classifiers. pages 174–188, 2001.

[129] C. Stone. Consistent nonparametric regression. *Annals of Statistics*, 5:595–645, 1977.

[130] A. Sun, E.-P. Lim, B. Benatallah, and M. Hassan. Fisa: feature-based instance selection for imbalanced text classification. In *Advances in Knowledge Discovery and Data Mining*, pages 250–254. Springer, 2006.

[131] A. Sun, E.-P. Lim, and Y. Liu. On strategies for imbalanced text classification using svm: A comparative study. *Decision Support Systems*, 48(1):191–201, 2009.

[132] A. Sun, E.-P. Lim, W.-K. Ng, and J. Srivastava. Blocking reduction strategies in hierarchical text classification. *Knowledge and Data Engineering, IEEE Transactions on*, 16(10):1305–1308, 2004.

[133] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[134] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

[135] A. Tan, D. Gilbert, and Y. Deville. Multi-class protein fold classification using a new ensemble machine learning approach. *Genome Informatics*, 14:206–217, 2003.

[136] E. Vernet, R.-C. Williamson, and M.-D. Reid. Composite multiclass losses. pages 1224–1232, 2011.

[137] K. Veropoulos, C. Campbell, and N. Cristianini. Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on artificial intelligence*, volume 1999, pages 55–60. Citeseer, 1999.

[138] J. Wang, J. Yand, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition*, pages 3360–3367, 2010.

[139] S. Wang and X. Yao. Multiclass Imbalance Problems: Analysis and Potential Solutions. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, 42(4):1119–1130, 2012.

[140] G. M. Weiss. Mining with rarity: a unifying framework. *ACM SIGKDD Explorations Newsletter*, 6(1):7–19, 2004.

[141] G. M. Weiss and F. Provost. Learning when training data are costly: the effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.

[142] D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on*, (3):408–421, 1972.

[143] T. Windeatt and R. Ghaderi. Binary labelling and decision-level fusion. *Information Fusion*, 2(2):103–112, 2001.

[144] G. Wu and E. Y. Chang. Class-boundary alignment for imbalanced dataset learning. In *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*, pages 49–56, 2003.

[145] G. Wu and E. Y. Chang. Kba: Kernel boundary alignment considering imbalanced data distribution. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):786–795, 2005.

[146] J. Wu, H. Xiong, P. Wu, and J. Chen. Local decomposition for rare class analysis. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 814–823. ACM, 2007.

[147] M. Wu and J. Ye. A small sphere and large margin approach for novelty detection using training data with outliers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(11):2088–2092, 2009.

[148] T.-F. Wu, C.-J. Lin, and R.-C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. of Mach. Learn. Res.*, 5:975–1005, 2003.

[149] J. Xiao, J. Hays, K.-A. Ehringer, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010.

[150] L. Xu, A. Krzyzak, and C. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(3):418–435, 1992.

[151] R. Yan, Y. Liu, R. Jin, and A. Hauptmann. On predicting rare classes with svm ensembles in scene classification. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 3, pages III–21. IEEE, 2003.

[152] Y. Yang. A study of thresholding strategies for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 137–145. ACM, 2001.

[153] X. Ye, M. Cheriet, and C. Y. Suen. StrCombo: combination of string recognizers. *Pattern Recognition Letters*, 23(4):381–394, 2002.

[154] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699. ACM, 2002.

[155] X. Zhao, X. Li, L. Chen, and K. Aihara. Protein classification with imbalanced data. *Proteins: Structure, Function, and Bioinformatics*, 70(4):1125–1132, 2008.

[156] Z. Zhou and X. Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):63–77, 2006.

# A. Surrogates Losses

In this Appendix we provide the demonstration that exist a subclass of surrogate losses, whose minimization brings simple and efficient estimators for Bayes (true) posteriors. These demonstrations have been provided together with Richard Nock, Wafa Bel Haj Ali, Frank Nielsen, Michel Barlaud. [33].

## A.1. Surrugate Losses

The surrogate risk is an estimator of the *true surrogate risk* computed over $\mathcal{D}$:

$$\varepsilon^{\psi}_{\mathcal{D}}(h, c) \doteq \mathbf{E}_{\mathcal{D}}[\psi(y_{ic}h(\boldsymbol{x}))] \ . \tag{A.1}$$

Any surrogate loss relevant to classification [12] has to meet $\mathrm{sign}(h_{\mathrm{opt}}(\boldsymbol{x}^{*})) = \mathrm{sign}(2\mathbf{Pr}_{\mathcal{D}}[y_c = 1|\boldsymbol{x} = \boldsymbol{x}^{*}] - 1)$, where $h_{\mathrm{opt}}$ minimizes $\mathbf{E}_{\mathcal{D}}[\psi(y_c h(\boldsymbol{x}))|\boldsymbol{x} = \boldsymbol{x}^{\star}]$. Hence, the sign of the optimal classifier $h_{\mathrm{opt}}$ is as accurate to predict class membership as Bayes decision rule. This Fisher consistency requirement for $\psi$ is called *classification calibration* [12]. We focus in this work on the subclass of classification calibrated surrogates that are strictly convex and differentiable.

**definition** [98] A **strictly convex loss** is a strictly convex function $\psi$ differentiable on $\mathrm{int}(\mathrm{dom}(\psi))$ satisfying (i) $\mathrm{im}(\psi) \subseteq \mathbb{R}^{+}$, (ii) $\mathrm{dom}(\psi)$ symmetric around 0, (iii) $\nabla_{\psi}(0) < 0$.

Definition A.1 is extremely general: should we have removed conditions (i) and (ii), Theorem 6 in [12] brings that it would have encompassed the intersection between strictly convex differentiable functions and classification calibrated functions. Conditions (i) and (ii) are mainly conveniences for classification: in particular, it is not hard to see that modulo scaling by a positive constant, the surrogate risk (C.2) is an upperbound of the empirical risk for any strictly convex loss. Minimizing the surrogate risk amounts thus to minimize the empirical risk up to some extent. We define the Legendre conjugate of any strictly convex loss $\psi$ as $\psi^{\star}(x) \doteq x\nabla_{\psi}^{-1}(x) - \psi(\nabla_{\psi}^{-1}(x))$. There exists a particular subset of strictly convex losses of independent interest [98]. A function $\phi : [0, 1] \to \mathbb{R}^{+}$ is called *permissible* iff it is differentiable on $(0, 1)$, strictly concave and symmetric around $x = 1/2$ [80, 98]. We adopt the notation $\overline{\phi} = -\phi$ [98].

**definition** [98] Given some permissible $\phi$, we let $\psi_{\phi}$ denote the **balanced convex loss** with signature $\phi$ as:

$$\psi_{\phi}(x) \doteq \frac{\overline{\phi}^{\star}(-x) - \phi(0)}{\phi\left(1/2\right) - \phi(0)} \ . \tag{A.2}$$

Balanced convex losses have an important rationale: up to differentiability constraints, they match the set of symmetric lower-bounded losses defining proper scoring rules [98], that is, basically, the set of losses that fit to classification problems without class-dependent misclassification costs. Table 6.1 provides examples of surrogate losses, most of which are strictly convex surrogates, some of which are balanced convex surrogates. We have derived Amari's $\alpha$-loss from Amari's famed $\alpha$ divergences [4] (proof omitted). The linear Hinge loss is *not* a balanced convex loss, yet it figures the limit behavior of balanced convex losses [98]. Remark that all signatures $\phi$ are well-known in the domain of decision-tree induction : from the top-most to the bottom-most, one may recognize Gini criterion, the entropy (two expressions), Matsushita's criterion and the empirical risk [80, 99].

A (regular) one dimensional *exponential family* [4] is a set of probability density functions whose elements admit the following canonical form:

$$p[x|\theta] \doteq \exp\left(x\theta - \psi(\theta)\right) p_0(x) \ , \tag{A.3}$$

where $p_0(x)$ normalizes the density, $\psi$ is a strictly convex differentiable function that we call the *signature* of the family, and $\theta$ is the density's natural parameter. It was shown in [98] that the efficient minimization of any balanced convex surrogate risk — *i.e.* a surrogate risk with a balanced convex loss — amounts to a maximum likelihood estimation $\hat{\theta} = H(\boldsymbol{x})$ at some $\boldsymbol{x}$ for an exponential family whose signature depends solely on the permissible function $\phi$. [98] suggest to use the corresponding *expected* parameter of the exponential family as the posterior:

$$\hat{\mathbf{Pr}}[y = 1|\boldsymbol{x}] = \hat{\mathbf{Pr}}_\phi[y = 1|\boldsymbol{x}; H] \doteq \nabla_{\bar{\phi}}^{-1}(H(\boldsymbol{x})) \in [0, 1] \ . \tag{A.4}$$

$\nabla_{\bar{\phi}}^{-1}$ plays the role of the link function (6.1). The quality of such an estimator shall be addressed in the following Section.

## A.2. Strictly convex losses and the efficient estimation of posteriors

There is a rationale to use (A.4) as the posterior: the duality between natural and expectation parameters of exponential families, via Legendre duality [12, 98], and the fact that the domain of the expectation parameter of one dimensional exponential families whose signature is (minus) a permissible function is the interval $[0, 1]$ [98]. We improve below this rationale, with the proof that *Bayes posteriors* satisfy (A.4) for the classifier which is the population minimizer of (A.4).

**theorem** Suppose $\psi$ strictly convex differentiable. The true surrogate risk $\mathbf{E}_{\mathbb{D}}[\psi(y_{ic}h(\boldsymbol{x}))]$ is minimized at the unique $h_{\text{opt}}(\boldsymbol{x})$ satisfying:

$$\frac{\nabla_\psi(-h_{\text{opt}}(\boldsymbol{x}))}{\nabla_\psi(h_{\text{opt}}(\boldsymbol{x}))} = \frac{p_c(\boldsymbol{x})}{1 - p_c(\boldsymbol{x})} \ . \tag{A.5}$$

Furthermore, is $\psi$ is a balanced convex loss, then the population minimizer $h_{\mathrm{opt}}$ of $\mathbf{E}_{\mathcal{D}}[\psi_{\phi}(y_{ic}h(\boldsymbol{x}))]$ satisfies:

$$p_c(\boldsymbol{x}) \; = \; \nabla_{\overline{\phi}}^{-1}(h_{\mathrm{opt}}(\boldsymbol{x})) \; , \tag{A.6}$$

for which

$$\mathbf{E}_{\mathcal{D}}[\psi_{\phi}(y_{ic}h_{\mathrm{opt}}(\boldsymbol{x}))] \; = \; \frac{\phi(p_c(\boldsymbol{x})) - \phi(0)}{\phi(1/2) - \phi(0)} \; . \tag{A.7}$$

(Proof omitted) Table 6.1 provides examples of expressions for $p_c(\boldsymbol{x})$ as in (A.6). Eq. (A.5) in Theorem (A.2) brings that we may compute an estimator $\hat{p}_c(\boldsymbol{x})$ as:

$$\hat{p}_c(\boldsymbol{x}) \; = \; \frac{\nabla_{\psi}(-h(\boldsymbol{x}))}{\nabla_{\psi}(h(\boldsymbol{x})) + \nabla_{\psi}(-h(\boldsymbol{x}))} \; . \tag{A.8}$$

This simple expression is folklore, at least for the logistic and exponential losses [18, 54]. The essential contribution of Theorem A.2 relies on bringing a strong rationale to the use of (A.4), as the estimators converge to Bayes posteriors in the infinite sample case. Let us give some finite sample properties for the estimation (A.4). We show that the sample-wise estimators of (A.6) are efficient estimators of (A.6); this is not a surprise, but comes from properties of exponential families [97]. What is perhaps more surprising is that the corresponding aggregation of classifiers is not a linear combination of all estimating classifiers, but a generalized $\nabla_{\overline{\phi}}^{-1}$-mean.

**theorem** Suppose we sample $n$ datasets $\mathcal{S}_j^{(c)}, j = 1, 2, ..., n$. Denote $\hat{h}_{\mathrm{opt},j}$ the population minimizer for $\mathbf{E}_{\mathcal{S}_j^{(c)}}[\psi_{\phi}(y_{ic}h(\boldsymbol{x}))]$. Then each $\hat{p}_{c,j}(\boldsymbol{x}) \doteq \nabla_{\overline{\phi}}^{-1}(\hat{h}_{\mathrm{opt},j}(\boldsymbol{x}))$ is the only efficient estimator for $p_c(\boldsymbol{x})$. The corresponding classifier $\hat{h}_{\mathrm{opt}}$ aggregating all $\hat{h}_{\mathrm{opt},j}$, is: $\hat{h}_{\mathrm{opt}}(\boldsymbol{x}) \doteq \nabla_{\overline{\phi}}\left(\frac{1}{n_{\boldsymbol{x}}}\sum_{j:(\boldsymbol{x},.)\in\mathcal{S}_j^{(c)}} \nabla_{\overline{\phi}}^{-1}(\hat{h}_{\mathrm{opt},j}(\boldsymbol{x}))\right), \forall \boldsymbol{x} \in \cup_j \mathcal{S}_j$, where $1 \leq n_{\boldsymbol{x}} \leq n$ is the number of subsets containing $\boldsymbol{x}$.

**proof** Let us pick $\psi = \overline{\phi}^{\star}$ in (A.3) and condition $p[x|\theta] \doteq p[x|\theta; \boldsymbol{x}^*]$ for each $\boldsymbol{x}^* \in \mathcal{O}$. We let $\mu \doteq p_c(\boldsymbol{x}^*)$ (remark that $\mu \in \mathrm{dom}(\phi) = [0, 1]$ because $\phi$ is permissible) the expectation parameter of the exponential family, and thus $\theta = \nabla_{\overline{\phi}}(\mu)$. Using the fact that $\nabla_{\overline{\phi}^{\star}} = \nabla_{\overline{\phi}}^{-1}$, we get the score:

$$s(x|\theta) \; \doteq \; \frac{\partial \ln p[x|\theta]}{\partial \theta} = x - \nabla_{\overline{\phi}^{\star}}(\theta) \; ,$$

and so $x$ is an efficient estimator for $\nabla_{\overline{\phi}^{\star}}(\theta) = \mu$; in fact, it is the only efficient estimator [97]. Thus, $\hat{p}_c(\boldsymbol{x}^*)$ is an efficient estimator for $p_c(\boldsymbol{x}^*)$. There remains to use (A.6) to complete the proof of Theorem A.2. □

# B. Universal Nearest Neigbours convergence

In this Appendix we provide the proof o the UNN convergence given any $\omega$ strongly smooth, strictly convex loss under the Weak Learning Assumption. This proof has been developed together with Richard Nock, Wafa Bel Haj Ali, Frank Nielsen, Michel Barlaud. [33].

**theorem** Suppose (**WLA**) holds and choose as $\psi$ is any $\omega$ strongly smooth, strictly convex loss. Then for any fixed $\tau \in [\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}_{\text{opt}}), \psi(0)]$, UNN has fit a leveraged $k$- $NN_k$ classifier $\mathcal{H}$ satisfying $\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}) \leq \tau$ provided the number of boosting iterations $T$ in the inner loop satisfies:

$$T \geq \frac{(\psi(0) - \tau)\omega m n_*}{2\vartheta^2 \varrho^2} . \tag{B.1}$$

**Proof sketch:** To fit UNN to the notations of (C.1), we let $h_c$ represent the leveraged $k$-$NN_k$ in which each $\boldsymbol{\alpha}_j$ is restricted to $\alpha_{jc}$. We first analyze $\varepsilon^{\psi}_{\mathcal{S}}(h_c, c)$ for some fixed $c$ in the outer loop of Algorithm 2, after all $\alpha_{jc}$ have been computed in the inner loop. We adopt the following notations in this proof: we plug in the weight notation the iteration $t$ and class $c$, so that $w^{(c)}_{ti}$ denotes the weight of example $\boldsymbol{x}_i$ at the beginning of the "**for** $c$" loop of Algorithm 2.

$\psi$ is $\omega$ strongly smooth is equivalent to $\tilde{\psi}$ being strongly convex with parameter $\omega^{-1}$ [78], that is,

$$\tilde{\psi}(w) - \frac{1}{2\omega} w^2 \text{ is convex,} \tag{B.2}$$

where we use notation $\tilde{\psi}(x) \doteq \psi^{\star}(-x)$. Any convex function $h$ satisfies $h(w') \geq h(w) + \nabla_h(w)(w' - w)$. We apply this inequality taking as $h$ the function in (C.28). We obtain, $\forall t = 1, 2, ..., T, \forall i = 1, 2, ..., m, \forall c = 1, 2, ..., C$:

$$D_{\tilde{\psi}}\left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti}\right) \geq \frac{1}{2\omega}\left(w^{(c)}_{(t+1)i} - w^{(c)}_{ti}\right)^2 . \tag{B.3}$$

On the other hand, Cauchy-Schwartz inequality and (C.10) yield:

$$\forall j \in \mathcal{S}, \sum_{i:j\sim_k i}\left(\mathrm{r}^{(c)}_{ij}\right)^2 \sum_{i:j\sim_k i}(w^{(c)}_{(t+1)i} - w^{(c)}_{ti})^2 \geq \left(\sum_{i:j\sim_k i}\mathrm{r}^{(c)}_{ij} w^{(c)}_{ti}\right)^2 . \tag{B.4}$$

**lemma** Under the **WLA**, index $j$ returned by WIC at iteration $t$ satisfies $\left|\sum_{i:j\sim_k i} w_{ti}^{(c)} \mathrm{r}_{ij}^{(c)}\right| \geq 2\vartheta\varrho$.

(proof omitted) Letting $e(t) \in \{1, 2, ..., m\}$ denote the index of the example returned at iteration $t$ by WIC in Algorithm 2, we obtain:

$$
\begin{aligned}
\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w_{(t+1)i}^{(c)} || w_{ti}^{(c)}\right) &\geq \frac{1}{2\omega m}\sum_{i:e(t)\sim_k i}\left(w_{(t+1)i}^{(c)} - w_{ti}^{(c)}\right)^2 && \text{(B.5)}\\
&\geq \frac{1}{2\omega m}\frac{\left(\sum_{i:e(t)\sim_k i} \mathrm{r}_{ie(t)}^{(c)} w_{ti}^{(c)}\right)^2}{\sum_{i:e(t)\sim_k i}\left(\mathrm{r}_{ie(t)}^{(c)}\right)^2} && \text{(B.6)}\\
&\geq \frac{2\vartheta^2\varrho^2}{\omega m} \times \frac{1}{\sum_{i:e(t)\sim_k i}\left(\mathrm{r}_{ie(t)}^{(c)}\right)^2} && \text{(B.7)}\\
&= \frac{2\vartheta^2\varrho^2}{\omega m n_{e(t)}} \geq \frac{2\vartheta^2\varrho^2}{\omega m n_*} \ . && \text{(B.8)}
\end{aligned}
$$

Here, (B.5) follows from (C.29), (B.6) follows from (B.4), (B.7) follows from Lemma B, and (B.8) follows from the fact that $\mathrm{r}_{ie(t)}^{(c)} = \pm 1$ when $e(t) \sim_k i$. Summing these inequalities for $t = 1, 2, ..., T$ yields:

$$
\sum_{t=1}^{T}\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w_{(t+1)i}^{(c)} || w_{ti}^{(c)}\right) \geq \frac{2T\vartheta^2\varrho^2}{\omega m n_*} \ . \qquad \text{(B.9)}
$$

Now, UNN meets the following property ([105], A.2):

$$
\varepsilon_{\mathcal{S}}^{\psi}(h_{(t+1)c}, c) - \varepsilon_{\mathcal{S}}^{\psi}(h_{tc}, c) = -\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w_{(t+1)i}^{(c)} || w_{ti}^{(c)}\right), \qquad \text{(B.10)}
$$

where $h_{(t+1)c}$ denotes $h_c$ after the $t^{th}$ iteration in the inner loop of Algorithm 2. We unravel (B.10), using the fact that all $\boldsymbol{\alpha}$ are initialized to the null vector, and obtain that at the end of the inner loop, $h_c$ satisfies:

$$
\varepsilon_{\mathcal{S}}^{\psi}(h_c, c) = \psi(0) - \sum_{t=1}^{T}\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w_{(t+1)i}^{(c)} || w_{ti}^{(c)}\right) \leq \psi(0) - \frac{2T\vartheta^2\varrho^2}{\omega m n_*} \ , \qquad \text{(B.11)}
$$

from (B.9). There remains to compute the minimal value of $T$ for which the right hand side of (B.11) becomes no greater than some user-fixed $\tau \in [0, 1]$ to obtain that $\varepsilon_{\mathcal{S}}^{\psi}(h_c, c) \leq \tau$.

The aggregation of the bounds for each $c = 1, 2, ..., C$ in $\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H})$ is immediate as it is an average of $\varepsilon_{\mathcal{S}}^{\psi}(h_c, c)$ over all classes. Hence, this minimal value of $T$, used for each $c = 1, 2, ..., C$, also yields $\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H}) \leq \tau$. This ends the proof of Theorem 6.1. □

# C. Gentle Nearest Neighbors Boosting over Proper Scoring Rules

This work has been developed in collaboration with Richard Nock[1], Wafa Bel Haj Ali[2], Frank Nielsen[3], Michel Barlaud[4]. Note that the work has been submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI).

## C.1. Introduction

Iterative approaches to learn classifiers have been playing a major role in machine learning and statistical learning for many decades. The most common high-level scheme consists in gradually combining from scratch classifiers obtained at each iteration, with the objective to minimize throughout iterations a convex differentiable risk called a *surrogate risk*, sometimes amended with a structural part based on data [11]. Unlike so-called greedy algorithms, that repeatedly perform fine-grained optimization steps [11], *boosting* algorithms rely on weak optimization stages much less demanding from the statistical and computational standpoints [54, 29, 102, 115]. In fact, the boosting theory involves at each iteration weak classifiers slightly different from pure random, *but* requires that the final combination be probably as close as required from optimum, in polynomial time.

Nearest neighbors ( $NN_k$) rules are a non-trivial field of choice for boosting algorithms [29, 102], as examples ideally play weak classifiers. In this case, we treat the boosting problem in its simplest form: the accurate leveraging of examples that vote among nearest neighbors. In particular, we compute nearest neighbors in the ambient space of data, *i.e.* as described over their initial features. There have been other approaches to boost nearest neighbors by learning features with (Ada)boosting algorithms, prior to computing nearest neighbor rules on these new sets of features [58] (and references therein). No boosting results are known for these algorithms, and it is in fact not known whether they achieve convergence to the optimum of Adaboost's exponential risk. A previous approach in our line of works is algorithm UNN (for "Universal Nearest Neighbors"), which brings boosting guarantees for merely all

[1]Université Antilles-Guyane, CEREGMIA-UFR DSE, Campus de Schoelcher, B.P. 7209, Schoelcher 97275, France

[2]CNRS - U. Nice, France.

[3]Sony Computer Science Laboratories, Inc., Tokyo, Japan

[4]Institut Universitaire de France and CNRS - U. Nice, France.

| | $\psi_\phi$ | $\phi$ | $\pi^*$ | $\delta_j(1/2)$ | weight update, $f : w_i \leftarrow f(w_i)$ |
|---|---|---|---|---|---|
| A | $(1-x)^2$ | $-x(1-x)$ | $\frac{1}{16}$ | $\frac{\eta(c,j)}{2n_j}$ | $w_i - 2\delta_{jc}y_{ic}y_{jc}$ |
| B | $\log_2(1+\exp(-x))$ | $x\ln x$ $+(1-x)\ln(1-x)$ | $\frac{\ln 2}{8}$ | $\frac{4\ln(2)\eta(c,j)}{n_j}$ | $\frac{w_i}{w_i\ln 2+(1-w_i\ln 2)\times\exp(\delta_{jc}y_{ic}y_{jc})}$ |
| C | $\log_2(1+2^{-x})$ | $x\log_2 x$ $+(1-x)\log_2(1-x)$ | | $\frac{4\eta(c,j)}{\ln(2)n_j}$ | $\frac{w_i}{w_i+(1-w_i)\times 2^{\delta_{jc}y_{ic}y_{jc}}}$ |
| D | $-x+\sqrt{1+x^2}$ | $-\sqrt{x(1-x)}$ | $\frac{1}{8}$ | $\frac{\eta(c,j)}{n_j}$ | $1-\frac{1-w_i+\sqrt{w_i(2-w_i)}\delta_{jc}y_{ic}y_{jc}}{\sqrt{1+\delta_{jc}^2 w_i(2-w_i)+2(1-w_i)\sqrt{w_i(2-w_i)}\delta_{jc}y_{ic}y_{jc}}}$ |
| E | $\frac{1}{2}x(\mathrm{sign}(x)-1)$ | $-\min\{x,1-x\}$ | N/A | | |

Table C.1.: From left to right: examples of balanced convex losses $\psi_\phi$ (A, B, C, D; we let $\ln$ denote the base-$e$ logarithm, and $\log_z(x) \doteq \ln(x)/\ln(z)$); permissible functions $\phi$; value of $\pi^*$ as defined in (C.41); expression of update $\delta_j$ in (C.10) for $\varepsilon = 1/2$; expression of the weight update in (C.11) (See text for details).

strictly convex differentiable surrogates relevant to classification [12, 102]. For a wide subset of surrogates, it yields simple and efficient estimators of posteriors [33].

There is, however, an analytical and computational bottleneck in UNN, as the leveraging coefficients are solutions to non-linear equations with no closed form expression in the general case. Boosting compliant approximations are possible, but in the context of $NN_k$ rules, they are computationally far too expensive to be performed at *each* boosting iteration on large datasets. Computationally affordable coarse-grained approximations are also possible, that yield compelling experimental results, but it is not known if they always lie within the boosting regime [102].

In this work, we propose a simple boosting compliant solution to this computational bottleneck. Our algorithm, GNNB for "Gentle Nearest Neighbors Boosting", performs adaptive Newton-Raphson steps to minimize any *balanced convex surrogate* [99] with guaranteed convergence rates. This class, which comprises the popular logistic and squared surrogates [54], match the set of even, twice differentiable *proper scoring rules* [59]. This is a proof of generality of our approach as being "proper" is the bare minimum one can request from a score — it roughly states that forecasting the right output yields the optimal score. Our main theoretical result establishes, for any of these surrogates, convergence rates towards global optimum that surprisingly compete with those known for UNN [102] — thus proving that a complex, time consuming leveraging procedure is not necessary for fast convergence towards the optimum. To the best of our knowledge, these are the first convergence rates under the boosting framework for Newton-Raphson approaches to general surrogate risk minimization, a set whose most prominent member is Gentle Adaboost [54]. The link with balanced convex surrogates optimization allows to show that GNNB equivalently fits class posteriors in a way that complies with weak universal consistency requirements. Experiments are provided on a dozen domains, including small domains from the UCI repository of machine learning database [7] and large computer vision domains: the Caltech [61] and SUN domains [149]. They display that GNNB outperforms UNN, both in terms of convergence rate and quality of the solutions obtained. They also display that, on large domains for which complex learning approaches like non-linear support vector machines or boosting with deep trees are ruled out for computational considerations, GNNB offers a simple, lightweight and competing alternative to heuristic methods like stochastic gradient descent. Our experiments come with an improvement of GNNB aimed at reducing the weak point represented by the curse of dimensionality for nearest neighbor algorithms on large domains. We provide a low-cost divide-and-conquer scheme which makes a partition of the description variables before running GNNB, and exploits links with density estimation in proper scoring rules to craft, out of all predictions, an aggregated score which is shown experimentally to outperform very significantly the vanilla approach without splitting.

The remaining of the work is organized as follows: Section 2 provides definitions. Section 3 presents GNNB. Section 4 and Section 5 respectively state and discuss its theoretical properties. Section 6 presents experiments, and Section 7 concludes the work.

## C.2. Definitions

### C.2.1. General setting

Our setting is multiclass, multilabel classification [115]. We have access to an input set of $m$ examples (or prototypes), $\mathcal{S} \doteq \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$. Vector $\boldsymbol{y}_i \in \{-1, 1\}^C$ encodes class memberships, assuming $y_{ic} = 1$ means that observation $\boldsymbol{x}_i$ belongs to class $c$. We let $\mathcal{H} : \mathcal{O} \rightarrow \mathbb{R}^C$ denote a classifier, $\mathcal{O}$ being the observations domain to which all $\boldsymbol{x}_i$ belong. The $c^{th}$ coordinate of the output of $\mathcal{H}$, $h_c \doteq \mathcal{H}_c$, is a classifier which segregates observations according to their membership to class $c$. We learn $\mathcal{H}$ by the minimization of a *total surrogate risk*:

$$\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}) \doteq \frac{1}{C}\sum_{c=1}^{C} \varepsilon^{\psi}_{\mathcal{S}}(h_c, c) \ , \tag{C.1}$$

where

$$\varepsilon^{\psi}_{\mathcal{S}}(h_c, c) \doteq \frac{1}{m}\sum_{i=1}^{m} \psi(y_{ic} h_c(\boldsymbol{x}_i)) \tag{C.2}$$

is a surrogate risk associated to class $c$, simply named surrogate risk hereafter [54, 99, 98, 115] (and many others). Quantity $y_{ic}h_c(\boldsymbol{x}) \in \mathbb{R}$ is the *edge* of classifier $h$ on example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, for class $c$.

### C.2.2. Proper scoring rules and surrogate losses

There exists numerous choices for the (surrogate) *loss* $\psi$. In this subsection, we motivate the analysis of a subset of particular interest, called balanced convex losses [99, 98]. For the sake of clarity, we assume in this Subsection that we have two classes ($C = 2$), and reduce the class vector to real $y \in \{-1, 1\}$ encoding membership to a so-called "positive" class ("1"). "$-1$" means observation does not belong to the positive class, or similarly belongs to a "negative"

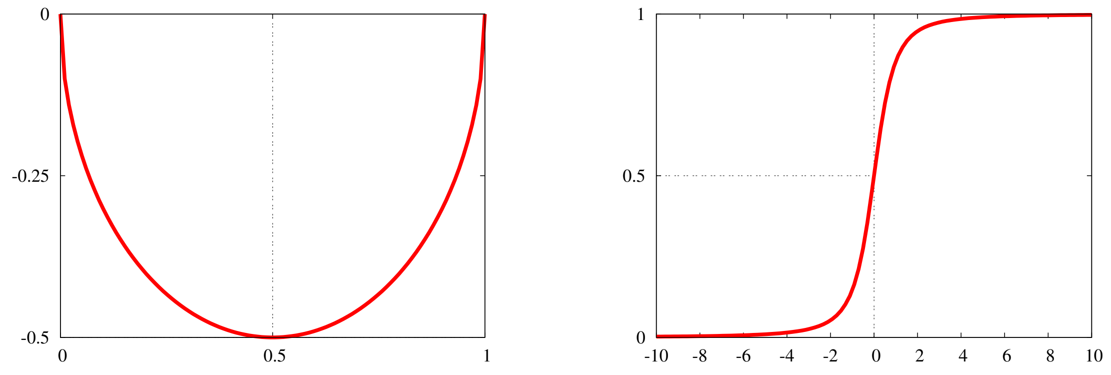

Figure C.1.: Plot $\phi$ of row D in Table C.1 (left) and its matching posterior estimate $\hat{p}_{\phi,h}$ as a function of $h \in \mathbb{R}$ (right).

class. In this case, a classifier $h$ outputs a single real value.

More general than the problem of predicting labels is the problem of estimating posteriors [59, 136]: let $p \doteq \hat{p}[y = 1|\boldsymbol{x}]$ define for short the unknown true posterior for observation $\boldsymbol{x}$. The discrepancy between an estimator $\hat{p}$ of $p$ and $p$ is measured by a loss $\ell_{[0,1]}(p\|\hat{p})$. The interval $[0, 1]$ in index recalls that its arguments are probabilities, and "$\|$" means that it is not assumed to be symmetric. There are three requirements one can put on a loss to fit it to statistical requirements of the estimation task while making it suited to convenient algorithmic minimization. The most important one, requirement **R1**, is fundamental in estimation, as it states that $\ell_{[0,1]}$ defines a (strictly) *proper scoring rule*: $0 = \ell_{[0,1]}(p\|p) < \ell_{[0,1]}(p\|q)$, for any $q$ and $p \neq q$ [59, 62, 98, 136]. This requirement is fundamental in that it encourages reliable estimations. Second, requirement **R2** states that the loss is *even* as $\ell_{[0,1]}(p\|\hat{p}) = \ell_{[0,1]}(1-p\|1-\hat{p})$, and thus there is no class-dependent mis-estimation cost, a common assumption in machine learning or classification. Third and last, requirement **R3** states that $\ell_{[0,1]}$ is twice differentiable. The following Theorem, whose proof can be found in [99, 98], exhibits the true shape of $\ell_{[0,1]}$.

**Theorem 1** *[99, 98] Any loss $\ell_{[0,1]}$ satisfies requirements* **R1**–**R3** *iff it is a Bregman divergence: $\ell_{[0,1]}(p\|q) = D_\phi(p\|q)$, for some permissible $\phi$.*

Theorem 1 makes use of two important definitions: a permissible $\phi$ satisfies: $\phi : [0, 1] \rightarrow \mathbb{R}^+$, it is differentiable on $(0, 1)$, strictly convex, twice differentiable on $(0, 1)$ and symmetric around $x = 1/2$. Also, for any strictly convex differentiable $\psi$, the *Bregman divergence* of (strictly convex differentiable) *generator* $\psi$ is:

$$D_\psi(x'\|x) \doteq \psi(x') - \psi(x) - (x' - x)\nabla_\psi(x) \ , \tag{C.3}$$

where "$\nabla$" denotes first order derivative. The Legendre convex conjugate of any strictly convex differentiable function $\psi$ is $\psi^\star(x) \doteq x\nabla_\psi^{-1}(x) - \psi(\nabla_\psi^{-1}(x))$.

**Definition 1** *[98] Given some permissible $\phi$, the* ***balanced convex loss*** *(BCL) with signature $\phi$, $\psi_\phi$, is:*

$$\psi_\phi(x) \doteq \frac{\phi^\star(-x)+\phi(0)}{\phi(0)-\phi\left(1/2\right)} \ . \tag{C.4}$$

We then have the following Theorem.

**Theorem 2** *[99, 98] The following identity holds true for any permissible $\phi$ and any classifier $h$:*

$$D_\phi(p(y)\|\hat{p}_{\phi,h}(\boldsymbol{x})) = (\phi(0)-\phi(1/2))\psi_\phi(yh(\boldsymbol{x})) \ ,$$

*where*

$$\hat{p}_{\phi,h}(\boldsymbol{x}) \doteq \nabla_\phi^{-1}(h(\boldsymbol{x})) \in [0,1] \ , \tag{C.5}$$

$$p(y) \doteq p[y=1|\boldsymbol{x}] \doteq \begin{cases} 0 & \textit{iff} \quad y=-1 \\ 1 & \textit{otherwise} \end{cases} \ . \tag{C.6}$$

Let us call $\hat{p}_{\phi,h}$ the matching posterior estimate for classifier $h$, as it represents an estimate $\hat{p}_{\phi,h}[y=1|\boldsymbol{x}]$. Figure C.1 plots $\hat{p}_{\phi,h}[y=1|\boldsymbol{x}]$ for choice D in Table C.1. It comes from Theorems 1 and 2 that balanced convex losses (for real valued classification) match a wide set of proper scoring rules (for estimation). Thus, they characterize a very important set of losses. We shall see in the following Section how to achieve the optimum of the score through a gentle optimization procedure with nearest neighbor classifiers.

Table C.1 includes popular examples of BCLs: squared loss (row A), (normalized) logistic loss (B), binary logistic loss (C), Matsushita's loss (D). Hinge loss (E) is not a BCL, yet it defines the asymptotes of any BCL [99], and its $\phi$ is the empirical loss [99]. Adaboost's exponential loss is not a BCL [54]. We finish by stating properties of $\phi$ and $\psi_\phi$. Let us assume that:

$$\min_{[0,1]} \mathrm{H}_\phi(x) > 0 \ ; \tag{C.7}$$

this is the case for all examples in Table C.1. Otherwise, we may replace $\phi$ by $\phi+\phi_2$ where $\phi_2$ is permissible and meets assumption (C.7). Since permissibility is closed by linear combinations, function $\phi+\phi_2$ is also permissible and satisfies (C.7). Since $\mathrm{H}_{\psi_\phi}(x) = 1/[(\phi(0)-\phi\left(1/2\right))\times \mathrm{H}_\phi(\nabla_\phi^{-1}(x))]$, assumption (C.7) implies:

$$\mathrm{H}^*_{\psi_\phi} \doteq \sup_{\mathbb{R}} \mathrm{H}_{\psi_\phi}(x) \ll \infty \ , \tag{C.8}$$

and in fact $\mathrm{H}^*_{\psi_\phi} = \mathrm{H}_{\psi_\phi}(0)$ for all examples in Table C.1, and is very small (Cf column $\pi^*$, (C.41) and Section C.4). The following Lemma states properties shown in [98].

**Lemma 1** *[98] For any permissible $\phi$, the following properties hold true: $\phi^\star(x)=\phi^\star(-x)+x, \forall x$; $\nabla_{\psi_\phi}(0)<0$, $\psi_\phi(0)=1$, $\mathrm{im}(\psi_\phi)\subseteq\mathbb{R}^+$.*

**Algorithm 3:** Algorithm GENTLE $NN_k$ BOOSTING, GNNB($\mathcal{S}, \phi, \varepsilon, k$)

```
Input: $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1,2,...,m,\ \boldsymbol{x_i} \in \mathcal{O},\ \boldsymbol{y_i} \in \{-1,1\}^C\}$, permissible $\phi$, $\varepsilon \in (0,1)$, $k \in \mathbb{N}_*$;
Let $\boldsymbol{\alpha}_j \leftarrow \boldsymbol{0}, \forall j = 1,2,...,m$;
for $c = 1,2,...,C$ do
    Let $\boldsymbol{w} \leftarrow \frac{1}{2(\phi(0)-\phi(1/2))}\boldsymbol{1}$;
    for $t = 1,2,...,T$ do
        [I.0]//Choice of the example to leverage
        Let $j \leftarrow \text{WIC}(\mathcal{S}, \boldsymbol{w})$;
        [I.1]//Computation of the gentle leveraging coefficient update, $\delta_j$
        Let
            $\eta(c,j) \leftarrow \sum_{i:j\sim_{\mathcal{S},k} i} w_{ti} y_{ic} y_{jc}$ ;   (C.9)
            $\delta_j \leftarrow \frac{2(1-\varepsilon)\eta(c,j)}{\text{H}^*_{\psi_\phi} n_j}$ , with $n_j \doteq |\{i : j \sim_k i\}|$ ;   (C.10)
        [I.2]//Weights update
        $\forall i : j \sim_k i$, let
            $w_i \leftarrow \frac{\nabla_\phi^{-1}\left(-\delta_j y_{ic} y_{jc} + \nabla_\phi((\phi(0)-\phi(1/2))w_i)\right)}{\phi(0)-\phi(1/2)}$ ;   (C.11)
        // we have $w_i \in \left[0, (\phi(0)-\phi(1/2))^{-1}\right]$
        [I.3]//Leveraging coefficient update
        Let $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$;
Output: $\mathcal{H}(\boldsymbol{x}) \doteq \sum_{j\sim_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j$
```

## C.2.3. Empirical risk and its minimization

Lemma 1 makes that surrogate risk minimization may be used as an approximate primer to the minimization of the empirical risk, as the total surrogate risk (C.1) upperbounds the empirical (Hamming) risk [115]:

$$\varepsilon^{\text{H}}_{\mathcal{S}}(\mathcal{H}) \doteq \frac{1}{C}\sum_{c=1}^{C} \varepsilon^{0/1}_{\mathcal{S}}(h_c, c) \ \le\ \frac{1}{\psi(0)}\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}) \ , \tag{C.12}$$

where

$$\varepsilon^{0/1}_{\mathcal{S}}(h_c, c) \ \doteq\ \frac{1}{m}\sum_{i=1}^{m} \text{I}[y_{ic} h_c(\boldsymbol{x}_i) < 0] \tag{C.13}$$

is the usual empirical risk associated to class $c$. To quantify the performance of the best possible classifier, we respectively define:

$$(\varepsilon^{\psi_\phi}_{\mathcal{S}})^*_c \ \doteq\ \inf_h \varepsilon^{\psi_\phi}_{\mathcal{S}}(h, c) \ , \tag{C.14}$$

$$(\varepsilon^{0/1}_{\mathcal{S}})^*_c \ \doteq\ \inf_h \varepsilon^{0/1}_{\mathcal{S}}(h, c) \ , \tag{C.15}$$

as the respective Bayes surrogate risks and Bayes empirical risks for class $c$. Averaging these expressions following (C.1) and (C.12), we respectively define $(\varepsilon^{\psi_\phi}_{\mathcal{S}})^*$ and $(\varepsilon^{0/1}_{\mathcal{S}})^*$ as the opti-

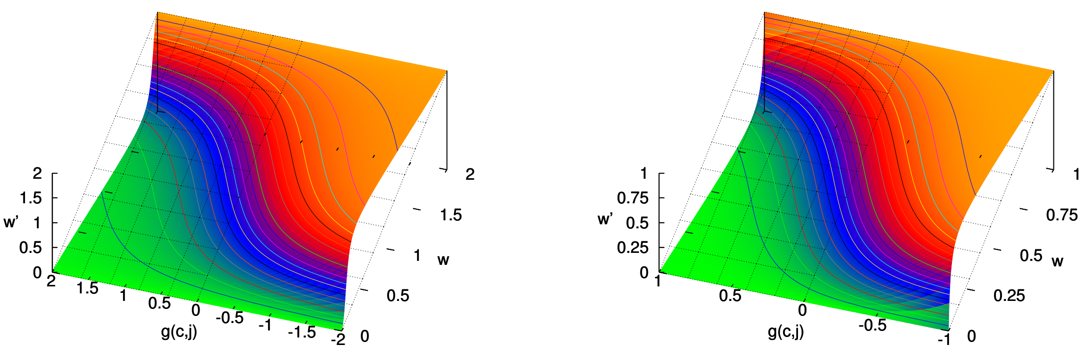

Figure C.2.: Weight update $w'$ computed as a function of $w$ and $g(c,j) \doteq \eta(c,j)/n_j$, when $y_{ic}y_{jc} = 1$ and $\varepsilon = 1/2$ (see Table C.1). The corresponding BCLs are the binary logistic loss (left) and Matsushita's loss (right). The black grid depicts the plane of equation $w' = w$.

mal total surrogate risk and empirical (Hamming) risk on $\mathcal{S}$. As a last remark, our minimization problems on the learning sample may be useful as well to minimize the *true (surrogate) risks*, that is, expectations of (C.1, C.12) in generalization, according to some unknown distribution from which $\mathcal{S}$ is supposed i.i.d. sampled. We refer to [12] and the references therein for details, not needed here.

## C.3. Gentle boosting for $NN_k$ rules

The nearest neighbors ( $NN_k$s) rule belongs to the simplest classification algorithms [38]. It relies on a non-negative real-valued "distance" function. This function, defined on domain $\mathcal{O}$, measures how much two observations differ from each other. It may not be a metric. We let $j \sim_k \boldsymbol{x}$ denote the assertion that example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, or simply example $j$, belongs to the $k$ $NN_k$s of observation $\boldsymbol{x}$. We abbreviate $j \sim_k \boldsymbol{x}_i$ by $j \sim_k i$ — and we say that example $i$ belongs to the *inverse neighborhood* of example $j$. To classify an observation $\boldsymbol{x} \in \mathcal{O}$, the $k$- $NN_k$ rule $\mathcal{H}$ over $\mathcal{S}$ computes the sum of class vectors of its nearest neighbors, that is: $\mathcal{H}(\boldsymbol{x}) = \sum_{j \sim_k \boldsymbol{x}} \mathbf{1} \circ \boldsymbol{y}_j$, where $\circ$ is the Hadamard product. $\mathcal{H}$ predicts that $\boldsymbol{x}$ belongs to each class whose corresponding coordinate in the final vector is positive. A *leveraged* $k$- $NN_k$ rule generalizes this to:

$$\mathcal{H}(\boldsymbol{x}) \quad = \quad \sum_{j \sim_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j \ , \tag{C.16}$$

where $\boldsymbol{\alpha}_j \in \mathbb{R}^C$ is a leveraging vector for the classes in $\boldsymbol{y}_j$. Leveraging approaches to nearest neighbors are not new [118], yet to the best of our knowledge no convergence rates were known, at least until the algorithm UNN [102]. Algorithm 3 presents our gentle boosting algorithm for the nearest neighbor rules, GNNB. It differs with UNN on the key part of (C.16):

the computation and update of the leveraging vectors. Instead of the repetitive solving of nonlinear equations — time consuming and with the risk, for approximations, of lying outside the boosting regime —, we prefer a simple scheme linear on the *weighted edge* $\eta(c, j)$ (see Algorithm 3). The scheme of UNN [102] is nonlinear in this parameter. Our updates also depend on integer $n_j$, the cardinality of the inverse neighborhood of example $j$, where $|.|$ denotes the cardinality (see Algorithm 3). Table C.1 gives the expressions of the weight update (C.11) for various choices of permissible $\phi$, and the expression of $\delta_j$ for the particular choice $\varepsilon = 1/2$. Figure C.2 plots examples of the weight update (C.11). The ranges of values, used in Figure C.2, are respectively $[-(\phi(0)-\phi(1/2))^{-1}, (\phi(0)-\phi(1/2))^{-1}]$ for $g(c, j)$, and $[0, (\phi(0)-\phi(1/2))^{-1}]$ for $w$ and $w'$. The two plots, similar, exemplify two important remarks valid for any BCL. First, when classes match for example $i$ and $j$, the weight of example $i$ decreases iff $\delta_j > 0$. This is a common behavior for boosting algorithms. Second, the regime of weight variations for extreme values of $g(c, j)$ appear to be very important, despite the fact that leveraging update $\delta_j$ is linear in the weighted edge. Thus, "gentle" updates do not prevent significant variations in weights.

## C.4. Properties of GNNB

### C.4.1. GNNB is Newton-Raphson

Our first result establishes that GNNB performs Newton-Raphson updates to optimize its surrogate risk, like Gentle Adaboost [54]. If we pick example $i$ in the inverse neighborhood of example $j$ to be updated for class $c$, we have $\partial\psi_\phi(y_{ic}h_c(\boldsymbol{x}_i))/\partial\delta_j = -w_i y_{ic} y_{jc}$, and $\partial^2\psi_\phi(y_{ic}h_c(\boldsymbol{x}_i))/\partial\delta_j^2 = \mathrm{H}_{\psi_\phi}(y_{ic}h_c(\boldsymbol{x}_i))$, so that the Newton-Raphson update for $\delta_j$ reads:

$$\delta_j \quad \leftarrow \quad \rho \times \frac{\eta(c,j)}{\sum_{i:j\sim_{\mathrm{S},k} i} \mathrm{H}_{\psi_\phi}(y_{ic}h_c(\boldsymbol{x}_i))} \ , \tag{C.17}$$

for some small learning rate $\rho$, typically with $0 < \rho \leq 1$. Comparing with (C.10), we get the following result.

**Theorem 3** GNNB *uses adaptive Newton-Raphson steps to minimize the surrogate risk at hand, $\varepsilon_{\mathrm{S}}^{\psi_\phi}$, with adaptive learning rate $\rho \doteq \rho(c, j, \varepsilon)$:*

$$\rho(c,j,\varepsilon) \quad = \quad \frac{2(1-\varepsilon)\sum_{i:j\sim_{\mathrm{S},k} i} \mathrm{H}_{\psi_\phi}(y_{ic}h_c(\boldsymbol{x}_i))}{\mathrm{H}^*_{\psi_\phi} n_j} \ . \tag{C.18}$$

*Furthermore,* $0 < \rho(c, j, \varepsilon) < 2(1-\varepsilon)$.

The Newton-Raphson flavor of GNNB might be useful to prove its convergence to the optimum of the surrogate risk at hand ($\varepsilon_{\mathrm{S}}^{\psi_\phi}$), yet the original boosting theory is more demanding than "mere" convergence to global optimum: it requires guaranteed convergence rates under weak assumptions about each iteration.

## C.4.2. GNNB boosts the surrogate risks

We consider the following *Weak Learning Assumption* about GNNB:

> (**WLA**) There exist constants $\varrho > 0, \vartheta > 0$ such that at any iterations $c, t$ of GNNB, index $j$ returned by WIC is such that the following holds:
>
> $$\frac{\sum_{i:j\sim_{\mathcal{S},k}i} w_i}{n_j} \geq \frac{\varrho}{\phi(0)-\phi(1/2)} \ ; \quad \textbf{(i)}$$
>
> $$|\hat{p}_{\boldsymbol{w}}[y_{jc} \neq y_{ic} | j \sim_{\mathcal{S},k} i] - 1/2| \geq \vartheta \ . \quad \textbf{(ii)}$$

Requirement (ii) corresponds to the usual weak learning assumption of boosting [99, 98, 115]: it postulates that the current *normalized* weights in the inverse neighborhood of example $j$ authorize a classification different from random by at least $\vartheta$. GNNB uses unnormalized weights that satisfy $(1/n_j)\sum_{i:j\sim_{\mathcal{S},k}i} w_i \in [0, 1/(\phi(0)-\phi(1/2))]$: requirement (i) thus implies that the unnormalized weights in the inverse neighborhood must not be too small. Intuitively, such a condition is necessary as unnormalized weights of minute order would not necessarily prevent (ii) to be met, but would impair the convergence of GNNB given the linear dependence of $\delta_j$ in the unnormalized weights. Notice also that unnormalized weights are all the smaller as examples receive the right labels: the fact that requirement (i) becomes harder to be met simply means that GNNB approaches the optimum sought. At the beginning of GNNB, the initialization with the null leveraging vectors ($\boldsymbol{\alpha}_j = \boldsymbol{0}, \forall j$) guarantees that we can pick in (i) $\varrho = 1/2$ everywhere.

The analysis we carry out is a bit more precise than usual boosting results: instead of giving, under the **WLA**, a lowerbound on the number of iterations needed to drive down the surrogate or empirical risks down some user-fixed threshold $\tau$, we rather provide a lowerbound on the total number of *weight updates*, for each class $c$. This number, $\ell(T, c)$, integrates the total number of boosting iterations and the size of inverse neighborhoods used. It is important to integrate these sizes since there is obviously a big difference for convergence between leveraging an example which votes for many others in "dense" parts of the data, and leveraging one which votes for none. Our main result is split in two. The first focuses on the surrogate risk, the second on the empirical risk. Let us define:

$$\phi_c(\mathcal{S}) \doteq \sum_{\boldsymbol{x}} \hat{p}_{\mathcal{S}}[\boldsymbol{x}]\phi(\hat{p}_{\mathcal{S}}[y_c = 1|\boldsymbol{x}]) \ , \tag{C.19}$$

$$\Delta_\phi(\mathcal{S}, \tau, c) \doteq \phi_c(\mathcal{S}) - ((1-\tau)\phi(1/2) + \tau\phi(0)) \ , \tag{C.20}$$

$$\Delta'_\phi(\mathcal{S}, \tau, c) \doteq \phi_c(\mathcal{S}) - \phi\left((1-\tau)/2\right) \ . \tag{C.21}$$

$\Delta_\phi(\mathcal{S}, \tau, c)$ and $\Delta'_\phi(\mathcal{S}, \tau, c)$ are differences between average values of $\phi$ taking values within $\pm(\phi(0) - \phi(1/2))$. We now state our main result on GNNB.

**Theorem 4** *Assume the **WLA** holds, and let $\tau \in [0,1]$. Suppose we run* GNNB *so that, $\forall c$, $\ell(T, c)$ meets:*

$$\ell(T, c) \ \geq \ \frac{\Delta_\phi(\mathcal{S}, \tau, c)(\phi(0) - \phi(1/2))H^*_{\psi_\phi}}{8\varepsilon(1-\varepsilon)\vartheta^2\varrho^2} \times m \ . \tag{C.22}$$

*Then the leveraged $k$- $NN_k$ $\mathcal{H}$ learned by* GNNB *satisfies:*

$$\varepsilon^{\psi_\phi}_{\mathcal{S}}(\mathcal{H}) \quad \leq \quad (\varepsilon^{\psi_\phi}_{\mathcal{S}})^* + \tau \ . \tag{C.23}$$

**Proof:** We craft a negative upperbound for the variation of the surrogate risk at hand (C.2) between two successive iterations, say $t$ and $t+1$. To keep references clear, we replace the index $j$ of the example returned by WIC by $e(t)$. We have:

$$\begin{aligned}
&\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{(t+1)c}, c) - \varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{tc}, c) \\
&= \frac{1}{m}\sum_{i:e(t)\sim_{\mathcal{S},k} i} \psi_\phi(y_{ic}h_{(t+1)c}(\boldsymbol{x})) - \frac{1}{m}\sum_{i:e(t)\sim_{\mathcal{S},k} i} \psi_\phi(y_{ic}h_{tc}(\boldsymbol{x})) \\
&= \frac{1}{m}\left\{ \sum_{i:e(t)\sim_{\mathcal{S},k} i} D_{\tilde{\psi}_\phi}(0 \| \nabla^{-1}_{\tilde{\psi}_\phi}(-y_{ic}h_{(t+1)c}(\boldsymbol{x}))) \right. \\
&\qquad \left. - \sum_{i:e(t)\sim_{\mathcal{S},k} i} D_{\tilde{\psi}_\phi}(0 \| \nabla^{-1}_{\tilde{\psi}_\phi}(-y_{ic}h_{tc}(\boldsymbol{x}))) \right\} \ ,
\end{aligned} \tag{C.24}$$

where $\tilde{\psi}_\phi(x) \doteq (\psi_\phi)^\star(-x)$ and (C.24) comes from Lemma 1 in [98]. We have $\nabla_{\psi_\phi}(x) = -\nabla^{-1}_\phi(-x)/(\phi(0)-\phi(1/2))$, implying $\nabla^{-1}_{\tilde{\psi}_\phi}(x) = \nabla_{(\psi_\phi)^\star}(x) = -\nabla_\phi((\phi(1/2)-\phi(0))x)$. Thus:

$$\tilde{\psi}_\phi(x) \quad = \quad -\frac{\phi\left((\phi(0)-\phi(1/2))x\right)}{\phi(0)-\phi(1/2)} \ . \tag{C.25}$$

Furthermore,

$$\begin{aligned}
&\nabla^{-1}_{\tilde{\psi}_\phi}(-y_{ic}h_{(t+1)c}(\boldsymbol{x}_i)) \\
&= \quad \frac{1}{\phi(0)-\phi(1/2)}\nabla^{-1}_\phi(-\delta_{e(t)}y_{ic}y_{e(t)c} - y_{ic}h_{tc}(\boldsymbol{x}_i)) \\
&\qquad = w_{(t+1)i} \ ,
\end{aligned} \tag{C.26}$$

and $\nabla^{-1}_{\tilde{\psi}_\phi}(-y_{ic}h_{tc}(\boldsymbol{x})) = w_{ti}$ as well, so that, using (C.25) and (C.26), we can simplify (C.24) as follows:

$$\begin{aligned}
&\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{(t+1)c}, c) - \varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{tc}, c)\\
&= \frac{1}{m}\left\{\sum_{i:e(t)\sim_{\mathcal{S},k}i} D_{\tilde{\psi}_\phi}(0\|w_{(t+1)i}) - \sum_{i:e(t)\sim_{\mathcal{S},k}i} D_{\tilde{\psi}_\phi}(0\|w_{ti})\right\}\\
&= -\frac{1}{m}\sum_{i:e(t)\sim_{\mathcal{S},k}i}\left\{\tilde{\psi}_\phi(w_{(t+1)i}) - \tilde{\psi}_\phi(w_{ti})\right.\\
&\qquad \left.-w_{(t+1)i}\nabla_{\tilde{\psi}_\phi}(w_{(t+1)i}) + w_{ti}\nabla_{\tilde{\psi}_\phi}(w_{ti})\right\}\\
&= -\frac{1}{m}\sum_{i:e(t)\sim_{\mathcal{S},k}i}\left\{\tilde{\psi}_\phi(w_{(t+1)i}) - \tilde{\psi}_\phi(w_{ti})\right.\\
&\qquad \left.+w_{(t+1)i}\left[\delta_{e(t)}y_{ic}y_{e(t)c} - \nabla_{\tilde{\psi}_\phi}(w_{ti})\right] + w_{ti}\nabla_{\tilde{\psi}_\phi}(w_{ti})\right\}\\
&= -\frac{1}{m}\sum_{i:e(t)\sim_{\mathcal{S},k}i} D_{\tilde{\psi}_\phi}(w_{(t+1)i}\|w_{ti})\\
&\qquad -\frac{\delta_{e(t)}}{m}\sum_{i:e(t)\sim_{\mathcal{S},k}i} w_{(t+1)i}y_{ic}y_{e(t)c}\ .
\end{aligned} \tag{C.27}$$

We lowerbound the divergence term, starting by an important property for $\psi_\phi$. We say that a differentiable function $\psi$ is $\omega$ *strongly smooth* [78] iff there exists some $\omega > 0$ such that $D_\psi(x'\|x) \leq \frac{\omega}{2}(x'-x)^2, \forall x, x'$.

**Lemma 2** *For any permissible $\phi$, $\psi_\phi$ is $\mathrm{H}^*_{\psi_\phi}$ strongly smooth, where $\mathrm{H}^*_{\psi_\phi}$ is defined in (C.8).*

**Proof:** Taylor-Lagrange remainder brings that there exists some $x'' \in (x, x')$ such that

$$D_{\psi_\phi}(x'\|x) = \frac{1}{2}(x-x')^2\mathrm{H}_{\psi_\phi}(x'') \leq \frac{1}{2}(x-x')^2\mathrm{H}^*_{\psi_\phi}$$

(we used (C.8)). This proves Lemma 2. □ It comes from [78] that $(\psi_\phi)^\star$ is $(\mathrm{H}^*_{\psi_\phi})^{-1}$ strongly convex; so,

$$\tilde{\psi}_\phi(w) - \frac{1}{2\mathrm{H}^*_{\psi_\phi}}w^2 \text{ is convex.} \tag{C.28}$$

Any convex function $\varphi$ satisfies $\varphi(w') \geq \varphi(w) + \nabla_\varphi(w)(w'-w), \forall w, w'$. We apply this inequality taking as $\varphi$ the function in (C.28), $w = w_{ti}$ and $w' = w_{(t+1)i}$. We sum for each $i$

such that $e(t) \sim_{\mathcal{S},k} i$:

$$\begin{aligned}&\sum_{i:e(t)\sim_{\mathcal{S},k}i} D_{\tilde{\psi}_\phi}\left(w_{(t+1)i}\|w_{ti}\right)\\ &\quad\geq \frac{1}{2\mathrm{H}^*_{\psi_\phi}}\sum_{i:e(t)\sim_{\mathcal{S},k}i}\left(w_{(t+1)i}-w_{ti}\right)^2 \ . \end{aligned}\tag{C.29}$$

Finally, Cauchy-Schwartz inequality yields:

$$\begin{aligned}&\sum_{i:e(t)\sim_{\mathcal{S},k}i}\left(y_{ic}y_{e(t)c}\right)^2\sum_{i:e(t)\sim_{\mathcal{S},k}i}\left(w_{(t+1)i}-w_{ti}\right)^2\\ &\quad\geq \left(\sum_{i:e(t)\sim_{\mathcal{S},k}i} y_{ic}y_{e(t)c}\left(w_{(t+1)i}-w_{ti}\right)\right)^2 \ . \end{aligned}\tag{C.30}$$

Fix for short $u \doteq \sum_{i:e(t)\sim_{\mathcal{S},k}i} w_{(t+1)i}y_{ic}y_{e(t)c}$. Plugging altogether (C.27), (C.29) and (C.30), we obtain the following upperbound for $\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{(t+1)c},c)-\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{tc},c)$:

$$\begin{aligned}&\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{(t+1)c},c)-\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{tc},c)\\ &\quad\leq -\frac{\left(u-\sum_{i:e(t)\sim_{\mathcal{S},k}i} w_{ti}y_{ic}y_{e(t)c}\right)^2}{2H^*_{\psi_\phi}m\sum_{i:e(t)\sim_{\mathcal{S},k}i}\left(y_{ic}y_{e(t)c}\right)^2}-\frac{\delta_{e(t)}}{m}u\\ &\quad= \frac{\Delta_t(u)}{m} \ . \end{aligned}\tag{C.31}$$

$\Delta_t(u)$ takes its maximum value for $u = u^* \doteq \eta(c,e(t)) - H^*_{\psi_\phi}n_{e(t)}\delta_{e(t)}$, for which we have: $\Delta_t(u^*) = (1/2)H^*_{\psi_\phi}n_{e(t)}\delta_{e(t)}\left(\delta_{e(t)} - (2\eta(c,e(t))/(H^*_{\psi_\phi}n_{e(t)}))\right)$. We pick $\delta_{e(t)}$ as in (C.10), *i.e.*, $\delta_{e(t)} = 2(1-\varepsilon)\eta(c,e(t))(H^*_{\psi_\phi}n_{e(t)})^{-1}$, for $\varepsilon\in(0,1)$. This yields:

$$\Delta_t(u) \leq \Delta_t(u^*) = -2\varepsilon(1-\varepsilon)\frac{\eta^2(c,e(t))}{H^*_{\psi_\phi}n_{e(t)}} \ . \tag{C.32}$$

We now show that the **WLA** implies a strictly positive lowerbound on the absolute value of edge $\eta(c,e(t))$. Letting I[.] be the indicator function, we have $\hat{p}_{\boldsymbol{w}_t}[y_{e(t)c}\neq y_{ic}|e(t)\sim_{\mathcal{S},k} i] = (\sum_{i:e(t)\sim_{\mathcal{S},k}i} w_{ti}\mathrm{I}\left[y_{ic}y_{e(t)c}=-1\right])/(\sum_{i:e(t)\sim_{\mathcal{S},k}i} w_{ti})$, and since $\mathrm{I}\left[y_{ic}y_{e(t)c}=-1\right] = 1 - y_{ic}y_{e(t)c}$, we obtain after simplification:

$$\hat{p}_{\boldsymbol{w}_t}[y_{e(t)c}\neq y_{ic}|e(t)\sim_{\mathcal{S},k} i] = \frac{1}{2}-\frac{\eta(c,e(t))}{2\sum_{i:e(t)\sim_{\mathcal{S},k}i} w_{ti}} \ .$$

Using statement (ii) in the **WLA**, this equality brings $|\eta(c, e(t))| \geq 2\vartheta \sum_{i:e(t)\sim_{\mathcal{S},k} i} w_{ti}$. Using statement (i) in the **WLA**, we finally obtain:

$$|\eta(c, e(t))| \ \geq \ 2\frac{\vartheta \varrho n_{e(j)}}{\phi(0) - \phi(1/2)} \ . \tag{C.33}$$

Plugging (C.33) into (C.32), and the resulting inequality into (C.31), we obtain:

$$\begin{aligned} & \varepsilon_{\mathcal{S}}^{\psi_\phi}(h_{(t+1)c}, c) - \varepsilon_{\mathcal{S}}^{\psi_\phi}(h_{tc}, c) \\ & \leq \ -8\varepsilon(1-\varepsilon)\frac{n_{e(t)}\vartheta^2\varrho^2}{mH^*_{\psi_\phi}(\phi(0) - \phi(1/2))^2} \ . \end{aligned} \tag{C.34}$$

At the initialization, all leveraging coefficients $\boldsymbol{\alpha}_j$ equal the null vector, and so the corresponding surrogate risk equals $\psi_\phi(0)$. To guarantee that $\varepsilon_{\mathcal{S}}^{\psi_\phi}(h_{Tc}, c) \leq (\varepsilon_{\mathcal{S}}^{\psi_\phi})^*_c + \tau$ under the **WLA**, for some $\tau \in [0, \psi_\phi(0)]$, it is thus sufficient to have:

$$\sum_{t=1}^{T} n_{e(t)} \geq \frac{(\psi_\phi(0) - (\varepsilon_{\mathcal{S}}^{\psi_\phi})^*_c - \tau)H^*_{\psi_\phi}(\phi(0) - \phi(1/2))^2}{8\varepsilon(1-\varepsilon)\vartheta^2\varrho^2} \times m \ .$$

This inequality leads to the statement of the Theorem, provided we remark the three following facts. The first one is proven in the following Lemma.

**Lemma 3** *We have* $(\varepsilon_{\mathcal{S}}^{\psi_\phi})^*_c = (\phi(0) - \phi_c(\mathcal{S}))/(\phi(0) - \phi(1/2))$, *where* $\phi_c(\mathcal{S})$ *is defined in (C.19).*

**Proof:** From Lemma 1, we have $\nabla_\phi^{-1}(x) = 1 - \nabla_\phi^{-1}(-x)$, with which we obtain after few derivations: $\arg\min_h \varepsilon_{\mathcal{S}_{\boldsymbol{x}}}^{\psi_\phi}(h, c) = \nabla_\phi(\hat{p}_{\mathcal{S}}[y_c = 1|\boldsymbol{x}])$, where $\mathcal{S}_{\boldsymbol{x}}$ is the subset of $\mathcal{S}$ whose observations match $\boldsymbol{x}$. Then, we compute $\varepsilon_{\mathcal{S}}^{\psi_\phi}(h, c)$ with this value for $h$, which, after simplification using Legendre conjugates, brings $\mathbf{E}_{\mathcal{S}_{\boldsymbol{x}}}[\psi_\phi(y_{ic}h(\boldsymbol{x}))] = (\phi(0) - \phi(\hat{p}_{\mathcal{S}}[y_c = 1|\boldsymbol{x}]))/(\phi(0) - \phi(1/2))$. Finally, we average this over all distinct observations in $\mathcal{S}$ to obtain Lemma 3. □ The last two facts that lead to the statement of the Theorem are simpler: we indeed have $\sum_{t=1}^{T} n_{e(t)} = \ell(T, c)$, and $\psi_\phi(0) = (\phi(0) - \phi(\nabla_\phi^{-1}(0)))/(\phi(0) - \phi(1/2)) = 1$. This concludes the proof of Theorem 4. □

## C.4.3. GNNB boosts the empirical risk

The following bound holds on the empirical risk.

**Corollary 1** *Assume the* ***WLA*** *holds, and let* $\tau \in [0, 1]$. *Suppose we run* GNNB *so that,* $\forall c$, $\ell(T, c)$ *meets:*

$$\ell(T, c) \ \geq \ \frac{\Delta'_\phi(\mathcal{S}, \tau, c)(\phi(0) - \phi(1/2))H^*_{\psi_\phi}}{8\varepsilon(1-\varepsilon)\vartheta^2\varrho^2} \times m \ . \tag{C.35}$$

*Then the leveraged* $k$*-* $NN_k$ $\mathcal{H}$ *learned by* GNNB *satisfies:*

$$\varepsilon_{\mathcal{S}}^{\mathrm{H}}(\mathcal{H}) \ \leq \ (\varepsilon_{\mathcal{S}}^{\mathrm{H}})^* + \tau \ . \tag{C.36}$$

**Proof:** Following [12], let us define $H(\epsilon) \doteq \inf_{\delta \in \mathbb{R}}\{\epsilon\psi_\phi(\delta) + (1-\epsilon)\psi_\phi(-\delta)\}$, $\psi_{\text{BJM}}(\epsilon') \doteq \psi_\phi(0) - H\left((1+\epsilon')/2\right)$, with $\epsilon \in [0,1]$ and $\epsilon' \in [-1,1]$. We have:

$$\begin{aligned} H(\epsilon) &= \inf_{\delta\in\mathbb{R}}\left\{\frac{\epsilon\phi^\star(-\delta) + (1-\epsilon)\phi^\star(\delta) + \phi(0)}{\phi(0) - \phi(1/2)}\right\} \\ &= \inf_{\delta\in\mathbb{R}}\left\{\frac{\phi^\star(\delta) - \epsilon\delta + \phi(0)}{\phi(0) - \phi(1/2)}\right\} \quad (C.37) \\ &= \frac{\phi^\star(\nabla_\phi(\epsilon)) - \epsilon\nabla_\phi(\epsilon) + \phi(0)}{\phi(0)-\phi(1/2)} \\ &= \frac{-\phi^{\star\star}(\epsilon) + \phi(0)}{\phi(0)-\phi(1/2)} = \frac{\phi(0) - \phi(\epsilon)}{\phi(0)-\phi(1/2)} \ . \quad (C.38) \end{aligned}$$

Here, (C.37) follows from Lemma 1, and (C.38) follows from the fact that $\phi$ is convex and lower semicontinuous. We thus have:

$$\psi_{\text{BJM}}(\epsilon') = \frac{\phi((1-\epsilon')/2) - \phi(1/2)}{\phi(0) - \phi(1/2)} \ . \quad (C.39)$$

It is proven in [12], Theorem 1, that:

$$\psi_{\text{BJM}}\left(\varepsilon^{0/1}_{\mathcal{S}}(h_c, c) - (\varepsilon^{0/1}_{\mathcal{S}})^*_c\right) \leq \varepsilon^{\psi_\phi}_{\mathcal{S}}(h_c, c) - (\varepsilon^{\psi_\phi}_{\mathcal{S}})^*_c \ .$$

The argument of $\psi_{\text{BJM}}$ is in $[0,1]$. On this interval, $\psi_{\text{BJM}}$ admits an inverse because $\phi$ admits an inverse on $[0, 1/2]$. To ensure $\varepsilon^{0/1}_{\mathcal{S}}(h_c,c) \leq (\varepsilon^{0/1}_{\mathcal{S}})^*_c + \tau'$, it is thus equivalent to ensure $\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_c,c) - (\varepsilon^{\psi_\phi}_{\mathcal{S}})^*_c \leq \psi_{\text{BJM}}(\tau')$. There remains to combine (C.39) and (C.22) to obtain the statement of Corollary 1. □

### C.4.4. GNNB is universally consistent

We analyze GNNB in the setting where WIC yields the leveraging of a subset of $m' < m$ examples out of the $m$ available in $\mathcal{S}$. This setting is interesting because it covers the optimization of GNNB in which we repeatedly leverage the most promising example, for example from the standpoint of $|\delta_j|$. We call GNNB$^*$ this variation of GNNB. We assume that $\mathcal{S}$ is sampled i.i.d. according to some fixed density. The following (weak) universal consistency result on GNNB is not surprising, as $NN_k$ approaches were the first to be proven consistent [129], and there have been since a wealth of weak and strong related universal consistency results [38]. The result also applies to UNN [102].

**Lemma 4** *Provided $T, k \rightarrow \infty$, $k/m' \rightarrow 0$ and $m'/m \rightarrow 0$, GNNB$^*$ is (weak) universally consistent: its expected Hamming risk converges to the Hamming risk of Bayes rule.*

**Proof sketch:** The proof gathers several blocks, the first of which is the fact that the empirical minimization of surrogate BCL $\psi_\phi$ in an $NN_k$ approach amounts to a maximum likelihood fitting of class posteriors [98] (Lemma 4). Indeed, after dropping temporarily the class index $c$ to focus first on a single class, the corresponding empirical risk $\varepsilon^{0/1}_{\mathcal{S}}(h) \doteq \sum_{\mathcal{S}} \hat{p}[(\boldsymbol{x}, y)]\psi_\phi(yh(\boldsymbol{x}))$ (C.13) satisfies (see Theorem 2):

$$\varepsilon^{0/1}_{\mathcal{S}}(h) \;\; \propto \;\; \sum_{l=1}^{v} \hat{p}[\mathcal{V}_l] \underbrace{\sum_{\mathcal{S}\cap\mathcal{V}_l} \frac{\hat{p}[(\boldsymbol{x}, y)]}{\hat{p}[\mathcal{V}_l]} D_\phi(p(y)\|\hat{p}_l)}_{\varepsilon_l} \;\; . \tag{C.40}$$

Here, $v$ is the number of Voronoi cells $\mathcal{V}_l, l = 1, 2, ..., v$; $\hat{p}[\mathcal{V}_l] \doteq \sum_{\mathcal{S}\cap\mathcal{V}_l} \hat{p}[(\boldsymbol{x}, y)]$ and $\hat{p}_l$ is $\hat{p}_{\phi,h}(\boldsymbol{x})$ for cell $\mathcal{V}_l$. Second, the right-population minimizer of any Bregman divergence is always the arithmetic average [8] (Proposition 1): at the minimum of each $\varepsilon_l$ in (C.40), $\hat{p}_l = \hat{p}^*_l \doteq \sum_{\mathcal{S}\cap\mathcal{V}_l} \hat{p}[(\boldsymbol{x}, y)]p(y)/\hat{p}[\mathcal{V}_l] = \hat{p}[y = +1|\mathcal{V}_l]$. Third, in a metric space, the number of distinct Voronoi cells is linear in $m'$ (but exponential in the dimension of $\mathcal{O}$ [38], Corollary 11.1), and as $m' \to \infty$ and provided $k/m' \to 0$, the distance between each point and all its $k$- $NN_k$ vanishes [38] (Lemma 5.1). So, as $k/m' \to 0$, $m'/m \to 0$ and provided the class posteriors are uniformly continuous, the expectation of $\hat{p}_l$ converges to the true cell posterior $p^*_l$. Last, Corollary 6.2 in [38] makes that a sufficient condition for the (weak) universal consistency of GNNB with respect to class $c$, and by extension to all classes for Hamming risk. $\square$

## C.5. Discussion

We chose not to normalize permissible functions, *i.e.* typically ensuring $\phi(1/2) = 1$ and $\phi(0) = 0$, because normalization would reduce the number of BCL that can be generated. For example, out of the two in rows B and C in Table C.1, the classical form of the logistic loss in B would disappear. Bounds in (C.22) and (C.35) advocate for a simple implementation of WIC: since the number of examples *leveraged* equals, on average, $\ell(T, c)/k$, we should put emphasis on leveraging examples with large inverse neighborhoods.

Our results call for several technical comparisons between GNNB, UNN and mathematical greedy algorithms [11]. Let us define:

$$\pi^* \;\; \doteq \;\; (\phi(0) - \phi(1/2))^2 H^*_{\psi_\phi}/2 \;\; , \tag{C.41}$$

and let us respectively define $\pi(\varepsilon)$ and $\pi'(\varepsilon)$ the terms factoring $m(\vartheta^2\varrho^2)^{-1}$ in (C.22) and (C.35). Because $\Delta_\phi(\mathcal{S}, \tau, c) \leq \Delta'_\phi(\mathcal{S}, \tau, c) \leq \phi(0) - \phi(1/2)$, it comes $\pi(1/2) \leq \pi'(1/2) \leq \pi^*$. Table C.1 provides examples of values for $\pi^*$ for different choices of $\phi$: they are small, in $[1/8, 1/16]$. Hence, when $\varepsilon = 1/2$, a sufficient number of weight updates to ensure (C.23) and (C.36) is $\ell^*(T, c) = (m/8) \times (\vartheta^2\varrho^2)^{-1}$. This happens to be a very reasonable constraint, given that the range of $\delta_j$ is of minute order compared to that in UNN, where $\delta_j$ can take on infinite values.

There is more: let $\ell_{\text{GNNB}}(T, c)$ and $\ell_{\text{UNN}}(T, c)$ denote the number of weight updates ensuring (C.36) (and thus ensuring (C.23) as well), respectively for GNNB and UNN. Inspecting Theorem 2.3 in [102] reveals that we have $\ell_{\text{GNNB}}(T, c) = \Theta\left(\ell_{\text{UNN}}(T, c)/(\varepsilon(1 - \varepsilon))\right)$. Hence, convergence rates of GNNB compete with those known for UNN.

| category | name | $m$ | $C$ | ref. |
|---|---|---|---|---|
| small | Liver | 345 | 2 | [7] |
| | Ionosphere | 351 | 2 | [7] |
| | Pima | 768 | 2 | [7] |
| | Scene | 2407 | 6 | [7] |
| | Satellite | 6435 | 6 | [7] |
| | Segment | 2310 | 7 | [7] |
| | Cardio | 2126 | 10 | [7] |
| | OptDig | 5620 | 10 | [7] |
| | Letter | 2561 | 26 | [7] |
| large | Caltech | 30607 | 256 | [61] |
| | SUN | 108656 | 397 | [149] |

Table C.2.: Domains used in our experiments, ordered in increasing number of classes, and then examples.

Mathematical greedy algorithms [11] have a very wide scope, and they can be specialized to statistical learning with a high-level scheme which is close to the iterative scheme of boosting algorithms. Situating GNNB with respect to them is thus interesting and reveals quite a favorable picture, from the computational and convergence rate standpoints. These greedy algorithms are indeed computationally expensive, requiring at each iteration a local optimization of the classifier that GNNB does not require. Regarding convergence rates, the bound most relevant to our setting can be stated as follows, omitting unnecessary technical details and assumptions [11] (Theorem 3.1 and its proof): after $t$ iterations, the squared risk of the greedy output is no more than $\tau(t) = \beta\left((\kappa/t) + (t\ln(m)/m)\right)$, for some $\kappa, \beta$ that meet in general $\kappa \gg m$, and $\beta > 10^4$. This bound takes its minimum for some $t^*$ which is $\gg m$ in general. Even for this large $t^*$, the corresponding upperbound on the squared risk, $\tau(t^*) = 2\beta\sqrt{\kappa\ln(m)/m}$, is significantly weaker than the guarantees of Theorem 4 and Corollary 1. Obviously however, our bounds rely on the **WLA**.

## C.6. Experiments

### C.6.1. Domains and metrics

Experiments have been performed on a dozen domains summarized in Table C.2. We have split the domains in small and large domains. Large domains have a significantly larger number of examples and classes. We refer the reader to the UCI machine learning repository for the related domains. We give a brief description of the "large" domains. The Caltech [61] domain is a collection of 30607 images of 256 object classes. We adopt the Fisher vectors [103] encoding in order to describe these images as features vector. Fisher Vector are computed over densely extracted SIFT descriptors and local color features, both projected with PCA in a sub space of dimension 64. Fisher Vectors are extracted using a vocabulary of 16 Gaussian and normalized separately for both channels and then combined by concatenating the two features vectors. This yields a 4K dimensional features vector. The SUN [102, 149] domain is a collection of 108656 images divided into 397 scenes categories. The number of images varies

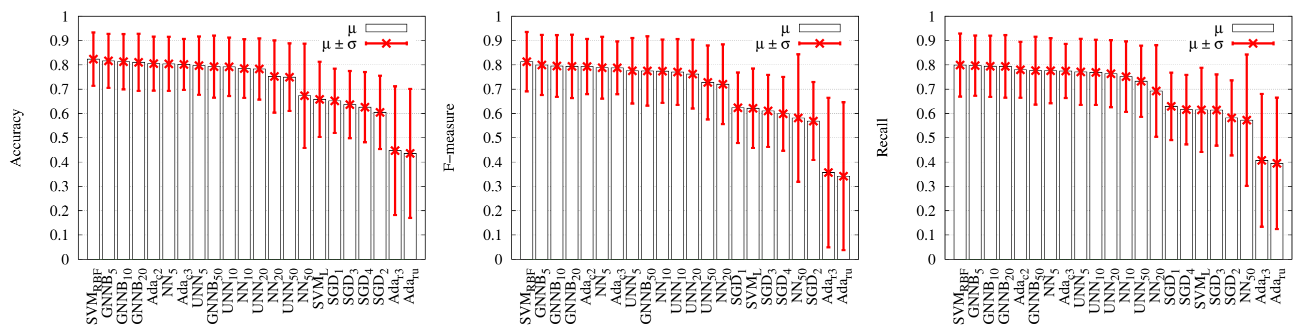

Figure C.3.: Average ($\mu$) $\pm$ Std dev. ($\sigma$) for accuracy (left), $F\,measure$ (center) and recall (right) over the small domains for all algorithms. In each plot, algorithms are ordered from left to right in decreasing average of the metric.

across categories, but there are at least 100 images per category. Each observation is represented as feature vector computed in the same way as for Caltech. Experiments are performed on a classical five-fold cross-validation basis, except for the large domains Caltech and SUN for which we have adopted the standardized approaches to use 30 (for Caltech) and 50 (for SUN) random images from each class to train classifiers and the remaining for testing.

We consider three types of metrics: the accuracy, which is one minus the Hamming risk (C.12, C.13) and which is directly optimized by GNNB (Corollary 1), the recall and the F-measure.

## C.6.2. Algorithms

To make an extensive analysis of the performances of GNNB, we have evaluated on small domains twenty-two (22) algorithms, on each of the three metrics. The version of GNNB used is GNNB(log) (Row B in Table C.1) with values of $k = 5, 10, 20, 50$. Contenders of GNNB can be put in five categories: ordinary nearest neighbors, universal nearest neighbors (UNN), stochastic gradient descent algorithms, (Ada)boosting algorithms and support vector machines.

Ordinary nearest neighbors, NN, and UNN(log) were tested with $k = 5, 10, 20, 50$. UNN performs for this choice of BCL approximations to the optimal boosted updates [102]. We used the simplest, non optimized WIC in UNN and GNNB, which returns index $t \mod m$.

We considered Stochastic Gradient Descent (SGD) [103, 16, 120], with four varying number of iterations. In the first, referred to as SGD$_1$, the number of iterations is equal to that of GNNB and UNN. In the second, SGD$_2$, number of iterations for SGD is fixed to be the "equivalent" to that of UNN and GNNB. Indeed, each iteration of SGD contributes to classify all examples in the training sample, while each iteration of UNN or GNNB contributes to classify $\theta(k)$ examples only. Thus, we need $\theta(m/k)$ iterations on UNN or GNNB for the classification of all examples to be eventually impacted. So, if $T$ is the total number of boosting iterations in UNN and GNNB, then we perform $T \times k/m$ iterations of SGD. The two last runs of SGD, hereafter noted SGD$_3$ and SGD$_4$, consider a larger number of iterations, two times the size of the training set in SGD$_3$ and three times in SGD$_4$. With those runs, we wanted to capture "limit" performances of SGD.

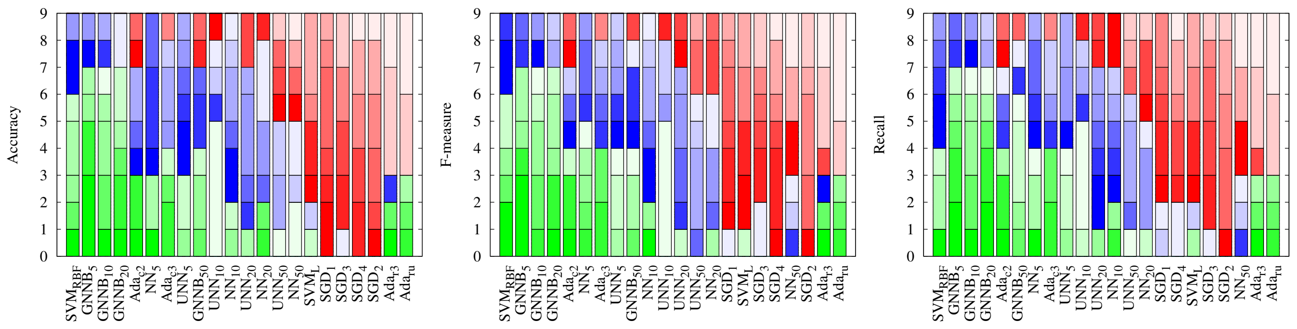

Figure C.4.: Ranking results: colors indicate the number of times an algorithm ranked among the top-tier (green), second-tier (blue) and third-tier (red, for the worst 8 algorithms) among all algorithms, over all small domains. For each color, the lighter the tone, the worse the rank. For example, dark green is rank 1, the lightest green is rank 7 and dark blue is rank 8. Algorithms are ordered from left to right in decreasing average of the metric at hand.

We also considered ADABOOST [115], with four different flavors. In ADABOOST$_{c2}$ (resp. ADABOOST$_{c3}$), the weak learner is C4.5 [109] with depth-2 (resp. depth-3) trees. C4.5 is a powerful weak learner: it repeatedly minimizes the expected $-\phi$ in row B of Table C.1. Again, the weak learner (WIC) used in GNNB and UNN is deliberately not optimized at all. For this reason, we have also tested ADABOOST with a non-optimized weak learner, which returns random trees. In ADABOOST$_{r3}$, these trees have depth 3, and in ADABOOST$_{ru}$, these trees have unbounded depth. In all four flavors of ADABOOST, the number of boosting rounds equals that of GNNB and UNN.

We have also considered two flavors of support vector machines, the first of which is affordable on small domains (but out of reach on our largest domains), non-linear SVM with radial basis function kernel in which the regularization parameter and the bandwidth are further optimized by a five-fold cross-validation on the training sample. We refer to them as SVM$_{RBF}$. The second flavor is linear SVM, SVM$_l$.

On large domains, we have tested GNNB against the contenders that scored top in the small domains or were easily scalable to large domains: $NN_k$, UNN, SGD. We have also tried SVM$_{LLC}$, that is, linear SVM with locality-constrained linear coding LLC [138].

## C.6.3. Results on small domains

### Results on average metrics

Figure C.3 presents the average results obtained for the 22 algorithms on the 3 metrics. Over all metrics, one can notice that the algorithms cluster in 3 groups. The first is the group of the best performing algorithms, with non-linear and mostly optimized large margin algorithms: SVM$_{RBF}$, GNNB (all $k$s), UNN (all $k$s), ADABOOST+C4.5, and NN with $k = 5, 10, 20$. The second group performs not as well as the first, with mostly linear classification algorithms: SVM$_l$, all SGD algorithms and NN with $k = 50$. The last group perform the worst of all, containing randomized large margin classification: ADABOOST with random trees.

Several observations can be made. First, the performances of all nearest neighbor methods

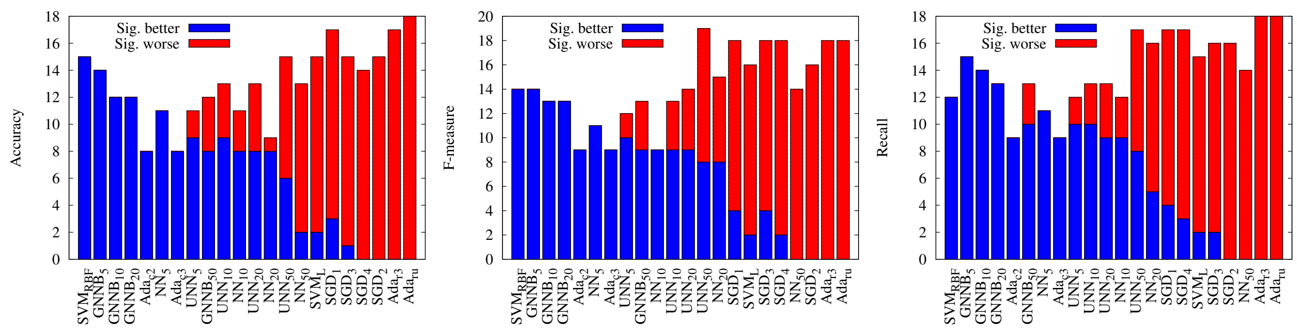

Figure C.5.: Ranking results contd: number of times each algorithm performed significantly better than the others (blue) or worse (red) according to Student paired $t$-test ($p = .1$). Algorithms order follow Figs C.3 and C.4 (see text).

($\text{GNNB}, \text{UNN}, \text{NN}$) decrease with $k$, in the range of values selected. Second, boosting nearest neighbors ($\text{GNNB}, \text{UNN}$) dampens the degradation of performances. Third, GNNB is the best of all kinds of nearest neighbor methods, from the standpoint of all metrics.

In fact, GNNB performs on par with SVM$_{\text{RBF}}$, for a wide range of $k$ $(5, 10, 20)$. The comparison with ADABOOST$_{\text{r3}}$ and ADABOOST$_{\text{ru}}$ is clear and final, as regardless of $k$ and for all metric, GNNB is better by more than .2 points on average; finally, GNNB performs also slightly better than ADABOOST+C4.5 (for $k = 5, 10, 20$). These are good news, first because GNNB is not optimized as ADABOOST+C4.5 is (for example from the standpoint of the weak learner), and second because GNNB is the lightest machinery among all, and so the easiest to scale to large domains.

## Ranking results

To drill down into these general results, we have also computed the global ranking results of each algorithm, recording the number of times each ranked first, second, third and so on, on the 9 domains. These results (Figure C.4), yield the following observations.

First, there is a subgroup in the group of the best performing algorithms according to the average metrics, which is the best according to ranking: SVM$_{\text{RBF}}$ and GNNB ($k = 5, 10, 20$). In this group, it appears that GNNB tends to be ranked higher than SVM$_{\text{RBF}}$, for a wide range of $k$ $(5, 10, 20)$, and this is particularly visible for F-measure and recall. From the recall standpoint, GNNB is almost always in top-tier results, while SVM$_{\text{RBF}}$ is more often in the second-tier.

Second, SGD performs poorly from the ranking standpoint, as all flavors mostly score among the third-tier results. We also observe that SGD performances are not monotonous with the number of iterations, as SGD$_1$ performs the best of all, both from the average and ranking standpoints. Linear classification methods tend to perform poorly, as displayed by SVM$_l$'s ranking results, very similar to those of stochastic gradient descent. If we compare ranking results with those of ADABOOST+random trees, which performs the worst of all from the expected metrics standpoint, then the ranking results display that SGD is more often in the third-tier of all algorithms.

Finally, ADABOOST with random trees sometimes scores very well among algorithms. Its

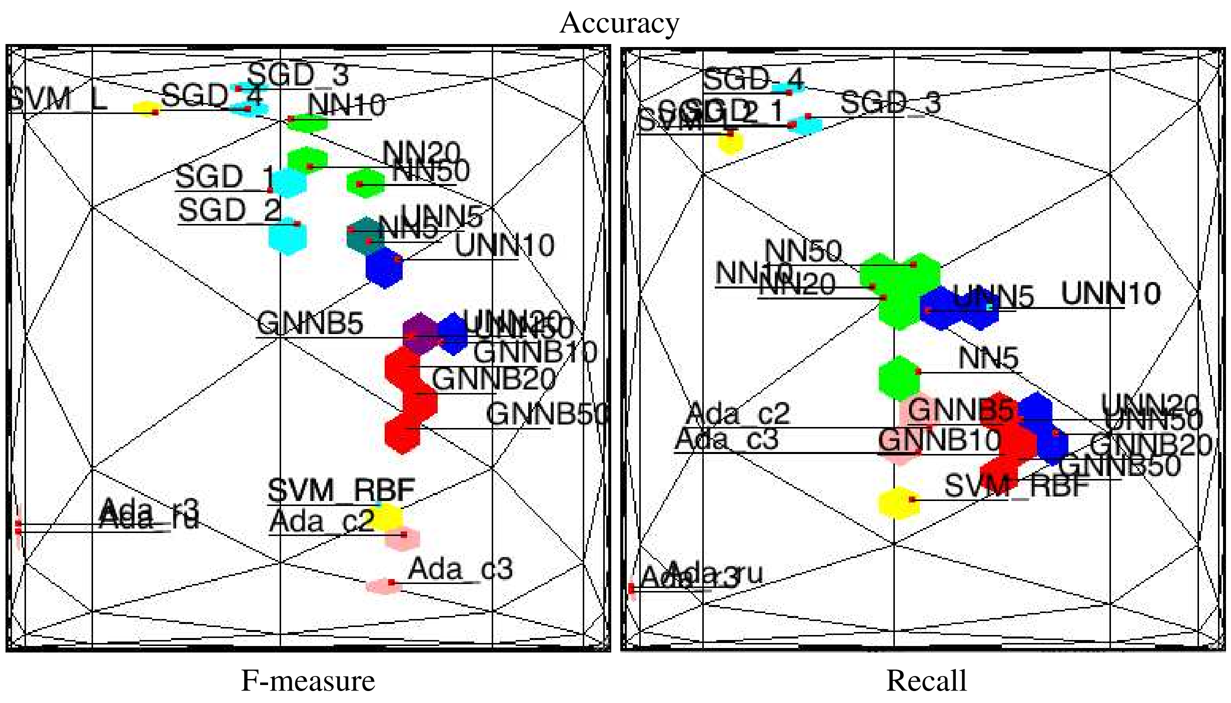

Figure C.6.: Manifold classification patterns for the accuracy (up, commented), $F measure$ (bottom left) and Recall (bottom right), for all 22 algorithms (see text); colors in hexagons cluster types of algorithms: red = GNNB, yellow = SVM, pink = ADABOOST, cyan = SGD, blue = UNN, green = $NN_k$ (see text and [100] for details).

ranking patterns indicate that the poor average results are essentially due to some domains for which replacing the randomized weak learner by an optimized one would make the classifier jump from the worst performances to at least second-tier performances.

We validated these ranking results with Student paired $t$-test comparison for each algorithm against all others (462 comparisons), recording those for which we can reject the null hypothesis (per-domain difference has zero expectation) for level $p = .1$, and then clustering the "significant" differences as to whether they are in favor, or in disfavor, of the algorithm at hand. Figure C.5 summarizes the results obtained, for all three metrics. They allow to cluster algorithms in three: those that are never significantly outperformed (GNNB for $k = 5, 10, 20$, $\text{SVM}_{\text{RBF}}$, ADABOOST+C4.5, NN for $k = 5$), those that never significantly outperform ($\text{SGD}_2$, NN for $k = 50$, ADABOOST+random trees), and the rest of the algorithms. They confirm that, on a wide range of values of $k$, GNNB performs on par with or better than optimized large margin non-linear algorithms ($\text{SVM}_{\text{RBF}}$, ADABOOST+C4.5).

## Classification patterns

The algorithms we have tested on small domains are representative of major families of supervised classification algorithms, ranging from linear to non-linear, induced to non-induced, including large margin classification methods, stochastic algorithms, and so on. To get a *qualitative* picture of the performances of GNNB, we have learned a manifold on the algorithms'

| top-1 accuracy ($\times 100$), Caltech | | | |
|---|---|---|---|
| | splits $n$ | | |
| iteration $t$ | 8 | 16 | 32 |
| 1 | 19.21 | 20.11 | 21.14 |
| 10 | 28.47 | 30.08 | 31.45 |
| 100 | 30.02 | 30.89 | 31.66 |
| 1000 | 30.38 | 31.52 | 32.65 |

| top-1 accuracy ($\times 100$), SUN | | | |
|---|---|---|---|
| | splits $n$ | | |
| iteration $t$ | 8 | 16 | 32 |
| 1 | 23.63 | 26.40 | 29.46 |
| 10 | 23.85 | 26.63 | 29.62 |
| 100 | 25.64 | 27.98 | 30.54 |
| 1000 | 25.64 | 27.97 | 30.56 |

| top-5 accuracy ($\times 100$), SUN | | | |
|---|---|---|---|
| | splits $n$ | | |
| iteration $t$ | 8 | 16 | 32 |
| 1 | 49.36 | 52.67 | 55.59 |
| 10 | 49.52 | 52.72 | 55.62 |
| 100 | 50.34 | 53.17 | 55.91 |
| 1000 | 50.33 | 53.19 | 55.92 |

Table C.3.: Performance of our divide-and-conquer approach on large domains for GNNB(log), using top-1 and top-5 accuracies.

results, one for each of the three metrics, as follows.

To get rid of the quantitative differences, we have normalized results to zero mean and unit standard deviation in each domain. Then, a manifold was learned using a standard procedure, with the normalized cosine similarity measure, and computing the second and third leading eigenvector of the Markov chain from the associated similarity matrix [101].

The corresponding manifolds are displayed in Figure C.6, using a focus+context display [100] in which the focus area is the center of the square. Plots also display in the background the mapping of a regular equilateral triangular tiling of the plane. The main observation from the plots, which cannot be observed in the average metrics and ranking experiments, is that the recall plot is much different from the accuracy and F-measure plots, that are very similar. The recall plot clusters the algorithms in three categories: linear classification (top-left, SGD, SVM$_l$), randomized boosting (ADABOOST+random trees, bottom left), and the rest of the algorithms (center). The accuracy and F-measure plot make a clear distinction between non-linear large margin "optimized" (down-right), non-linear large margin "random" (down-left) and linear (up). Looking at nearest neighbor algorithms as $k$ increases reveals that boosted nearest neighbor algorithms (UNN, GNNB) tend to behave more and more like large margin classification algorithms as $k$ increases, while vanilla $NN_k$ tends to behave more and more like linear classification algorithms as $k$ increases. This observation for $NN_k$ is consistent with the simple example that sampling two spherical Gaussians with identical variance (one for each class) makes a non-linear frontier for $k, m \ll +\infty$, which tends to a linear one as both parameters tend to $+\infty$.

## Training times

We have computed the training times for GNNB (all $k$s), SVM$_{\text{RBF}}$ and ADABOOST+C4.5 (depth-3 trees), that belong to the top-5 or top-6 algorithms in terms of average metric performances. We have computed the ratio between training times for each domain and each value of $k$, for SVM$_{\text{RBF}}$ to GNNB, and ADABOOST+C4.5 to GNNB. As already displayed for UNN [102], the ratios are clearly in favor of GNNB. We obtained a synthetic and accurate picture of these advantages by regressing the ratio against $1/k$, that is, computing the regression coefficients $a, b$ for $\rho = (a/k) + b$. Here, $\rho$ is *e.g.* the ratio for the SVM$_{\text{RBF}}$ training time to GNNB training time, averaged over all domains, and then computed for each $k$. The results, that we give with

| | NN | GNNB | $\text{SGD}_1$ | $\text{SVM}_{\text{LLC}}$ | | |
|---|---|---|---|---|---|---|
| $f$ | 4K | 4K | 4K | 4K | 4×4K | 5×4K |
| Acc. | 25.50 | 36.40 | 36.00 | 27.99 | 35.18 | 36.76 |
| F-m. | 20.97 | 29.24 | 30.87 | 24.00 | 31.67 | 33.33 |
| Rec. | 17.13 | 31.47 | 31.35 | 22.00 | 28.66 | 30.41 |

Table C.4.: Results on Caltech (accuracy, F-measure and recall are $\times 100$). $f$ is the number of features, and $k = 200$ for $NN_k$, GNNB.

the coefficient of determination $r^2$, are (t.t. = training time):

$$\frac{\text{t.t.}(\text{SVM}_{\text{RBF}})}{\text{t.t.}(\text{GNNB})} \approx \frac{851}{k} + 49 \quad (r^2 = 0.96) \ ,$$
$$\frac{\text{t.t.}(\text{ADABOOST+C4.5})}{\text{t.t.}(\text{GNNB})} \approx \frac{9547}{k} + 398 \quad (r^2 = 0.97) \ .$$

These regressions mean that, regardless of the value of $k$, $\text{SVM}_{\text{RBF}}$'s training time is at least roughly 50 times that of GNNB, while ADABOOST+C4.5's training time is at least roughly 400 times that of GNNB. These ratios are in good agreement with those observed in favor of UNN against $\text{SVM}_{\text{RBF}}$ and ADABOOST+stumps [102].

### Summary for small domains

The results obtained on small domains bring the following general observations. First, GNNB scores among the top algorithms and performs on par with, or better than, optimized machineries like non-linear SVM or ADABOOST+trees, and it beats these latter approaches, from the training times standpoint, by factors that range from tens to thousands of times. These good performances go hand in hand with the desirable property that results are stable against reasonable variations of $k$, which is not the case for UNN.

## C.6.4. Results on large domains

We have used the instantiation of SGD that performed the best on small domains, $\text{SGD}_1$, and the number of iterations of GNNB and $\text{SGD}_1$ is 6000. We split the analysis between the comparison of GNNB vs UNN, and GNNB vs the rest of the algorithms.

### A divide-and-conquer optimization of GNNB

It is well known that NN classifiers suffer of the curse of dimensionality [38], so that the accuracy can decrease when increasing the size of descriptors. This may also affect GNNB, in particular on large domains like SUN and Caltech. Fisher vectors employ powerful descriptors but they generate a space with about 4K dimension for 32 gaussians, which could impair GNNB performance. Our approach relies on a property of classification-calibrated losses that one can get simple posteriors estimators from the classifier's output, based on the matching

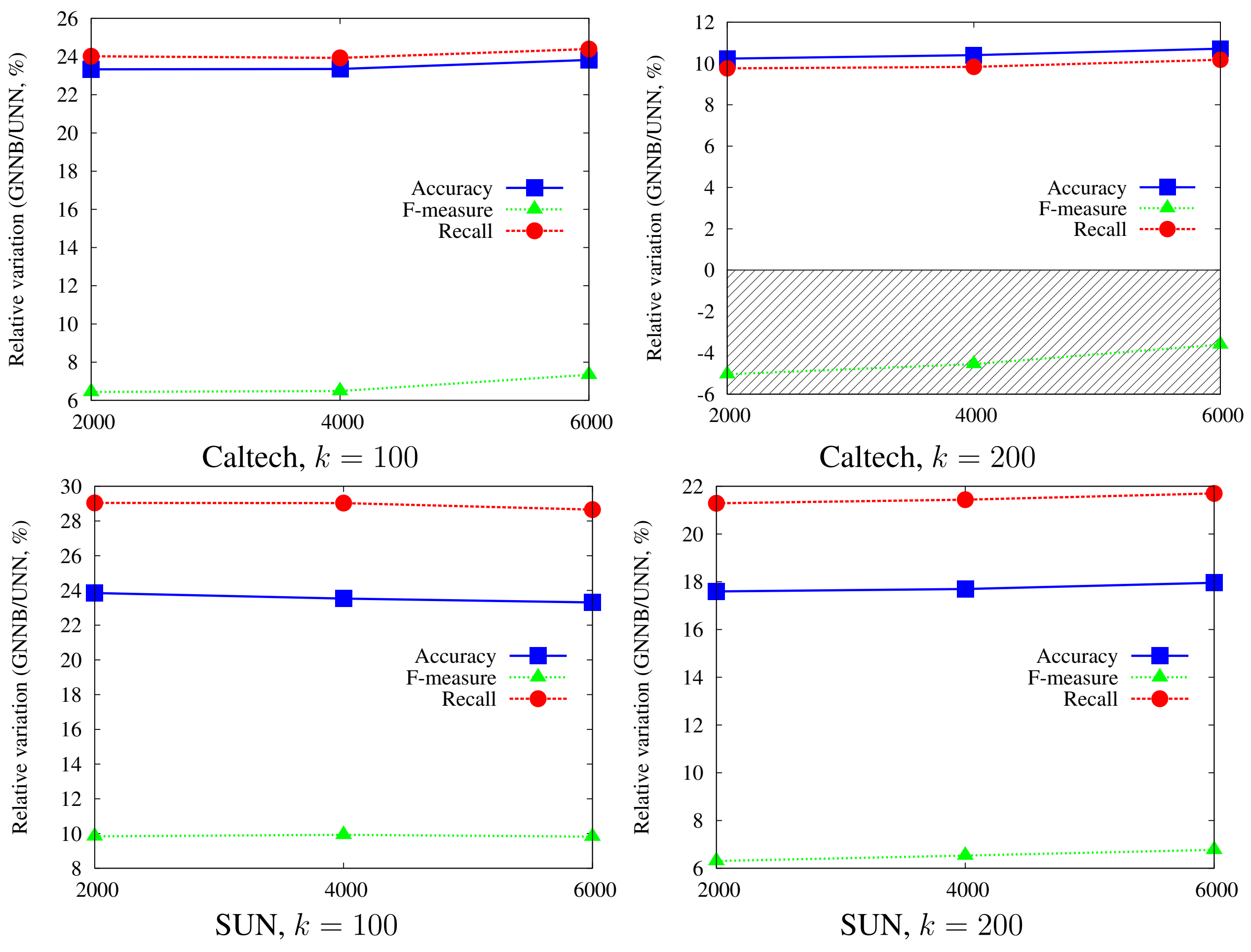

Figure C.7.: Relative variation (in %) of GNNB over UNN, expressed as a function of the number of boosting rounds $t$. Positive values mean better results for GNNB; a dashed rectangle indicates the zone of negative values.

posterior $\hat{p}_{\phi,h}$ in (C.5) (see [33, 102], and the right plot Figure C.1) . The method we propose consists in (i) splitting the set of descriptors, (ii) compute posteriors over each of these sets, and finally (iii) average the posteriors over all splits. The set of Fisher descriptors is split in a regular set of $n \in \{8, 16, 32\}$ sub-descriptors; each set is normalized in $L_1$ or $L_2$ norm. Finally, posteriors are combined linearly, with an arithmetic average.

Table C.3 presents the results obtained on our large domains. Results in Table C.3 show that increasing $n$, the number of splits, always improves the performances of GNNB, in a range between 1% and 6%, the largest improvements being obtained for the largest domain (SUN). We have also checked that increasing the number of iterations still keeps this pattern, which is thus robust to both variations in $n$ and the total number of boosting iterations $t$. We have witnessed in some cases differences that become much more important with the increase in $t$. For example, after 7650 iterations on Caltech, GNNB's top-1 accuracy becomes respectively 31.91%, 33.79% or 36.13% for $n = 8, 16$ and $n = 32$.

In the following results, GNNB is ran with $n = 32$ splits. To remain fair with UNN, we have also carried out the same $n = 32$ splitting strategy, and checked that it improves the performances of UNN as well.

### Results on Caltech

The two left plots of Figure C.7 display the results of GNNB vs UNN on Caltech. We have chosen to put emphasis on the relative variations of GNNB wrt UNN, to get a clean quantitative picture of the improvements. Those plots display that GNNB outperforms UNN, and this phenomenon is dampened as $k$ increases. For $k = 100$, the improvement of GNNB on accuracy and recall exceed $+20\%$, and it is reduced to $+10\%$ for $k = 200$. Table C.4 compares GNNB to $NN_k$, SGD$_1$ and LLC encoding for linear SVM using the same codebook as [138]. LLC produces a very large number of descriptors compared to the 4K Fisher vectors used in the other approaches, and a significant part of the improvement due to encoding comes in fact from this very large description space [21]. In order to make fair comparisons with the other techniques that rely on 4K descriptors, we have extracted the two first layers of descriptors of LLC, of size 4K and 4×4K, to analyze SVM$_{\text{LLC}}$ over 4K descriptors, 4×4K descriptors and 4K+4×4K = 5×4K descriptors.

The accuracy results show that GNNB tops $NN_k$ and SGD$_1$, and beats SVM$_{\text{LLC}}$ until 16K descriptors. It is only when SVM$_{\text{LLC}}$ uses five times the number of descriptors of GNNB that it beats GNNB. In fact, when using the same description size as the other algorithms, LLC encoding is beaten from the standpoint of all metrics by GNNB and SGD$_1$. SGD$_1$ performs well from the standpoint of the F-measure, and performs on par with GNNB from the recall standpoint.

### Results on SUN

The comparison between GNNB and UNN (Figure C.7, right plots) displays the same patterns as for Caltech: as $k$ increases, the improvements of GNNB wrt UNN are dampened, yet they are now always in favor of GNNB, and the improvements are more significant.

Table C.5 compares the performances of GNNB, $NN_k$ and SGD$_1$. This time, SGD$_1$ beats GNNB from the standpoint of all metrics. This observation has to be taken with a pinch of salt, as the experimental setting for large domains disfavors GNNB. Indeed, GNNB, like UNN and $NN_k$, is a local classifier, and for such kinds of methods, the experimental setting amounts to producing random edited nearest neighbors [38] by filtering out most ($\approx$80$\%$) of the dataset, with consequences that are likely to be harmful as (i) drastic random editing increases significantly the distances between nearest neighbors and impairs estimators quality and (ii) the weak learner WIC used so far makes no selection among examples selected. On the other hand, random subsampling may have minor effects on linear separators, and thus on SGD: for example, when a linear separator exists with minimal margin $\gamma$, sampling $\tilde{\Omega}(\gamma^{-2})$ examples (tilde hides dependences in other parameters) at random still guarantees with high probability the existence of a linear separator with $\Omega(\gamma)$ margin and small true risk [15].

To get a more reliable picture of the performances reachable by GNNB on our largest domain, we have thus considered a naive optimization of the weak index chooser WIC in GNNB, and tested it in an experimental setting computationally affordable for GNNB and less in disfavor than the former one. The new WIC in GNNB returns the index of the example with the

| | NN | GNNB | $\text{SGD}_1$ |
|---|---|---|---|
| Acc. | 20.92 | 30.16 | 32.20 |
| F-m. | 17.39 | 27.02 | 30.96 |
| Rec. | 23.39 | 34.32 | 35.53 |

Table C.5.: Results on SUN (conventions follow Table C.4).

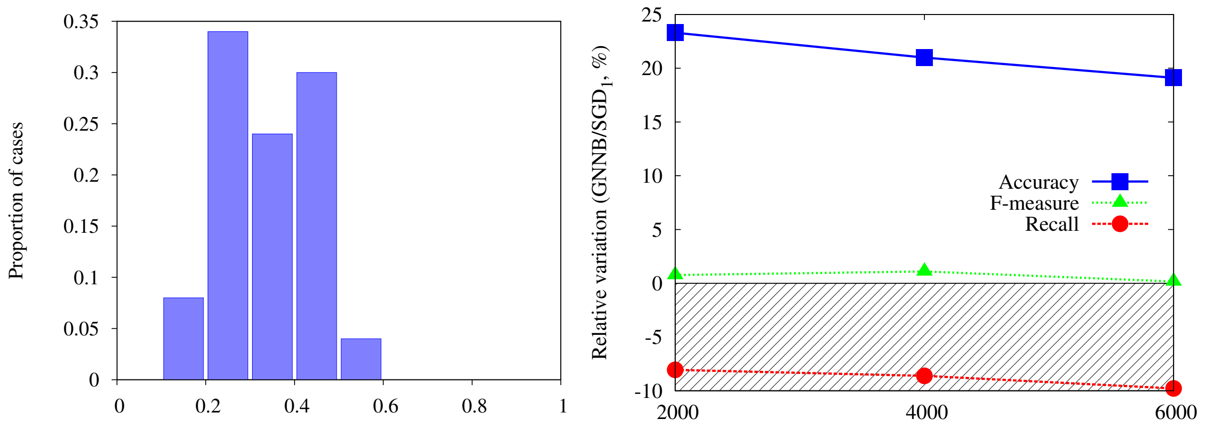

Figure C.8.: Left: frequency of cases among classes for the proportion of examples used per class by GNNB; Right: GNNB$^*$ ($k = 200$) vs $\text{SGD}_1$ (conventions follow Fig. C.7).

largest current $|\delta_j|$. This version of GNNB, GNNB$^*$, is shown to be universally consistent in Subsection C.4.4. To alleviate the negative effects of the experimental setting, we performed a holdout estimation of GNNB$^*$'s performances by training/testing on a random half/half partition of the database, for 6000 iterations. This computationally intensive setting was not applicable to $\text{SGD}_1$, but fortunately we could check that the number of examples actually used by GNNB$^*$ (*i.e.* leveraged or reweighted) was comparable to that used by $\text{SGD}_1$, so that both algorithms had at least approximately the same amount of information for learning. This is shown in Figure C.8 (left plot): we have recorded for each class the percentage of examples actually used in training by GNNB$^*$, and plotted the corresponding estimated density. The expectation of this density is roughly $40\%$. Thus, $40\%$ of the $50\%$ of each class was used in average by GNNB$^*$, *i.e.* $\approx$54 examples, to be compared to the 50 used by $\text{SGD}_1$.

The right plot in Figure C.8 summarizes the improvements of GNNB$^*$ with respect to $\text{SGD}_1$. One sees this time that even when the recall of GNNB$^*$ is smaller than that of $\text{SGD}_1$, the accuracy is now comparatively significantly higher. While optimizing WIC in GNNB was not the purpose of this work, this simple experiment displays that (i) there is significant room for further improvement of GNNB while staying in the boosting/consistency regimes, and (ii) these improvements are affordable in a large scale learning setting.

## C.7. Conclusion

We proposed a simple Newton-Raphson leveraging scheme for nearest neighbors to optimize any even, twice differentiable proper scoring rule, with guaranteed convergence rates under the boosting framework that compete with those known for non-gentle approaches [102]. To the best of our knowledge, those convergence rates in the boosting framework are knew for gentle boosting approaches. Experiments display that GNNB significantly outperforms UNN, converging faster to better solutions. On small domains, GNNB performs on par with or better than powerful non-linear large margin learners like non-linear SVM and Adaboost+C4.5. Large domains, on which these latter approaches are ruled out for computational costs, display that GNNB provides a lightweight competitive alternative to stochastic gradient descent. A byproducts of our experiments shows that manifold learning may be useful to assess global qualitative comparisons of algorithms. As learning algorithms are rapidly becoming more numerous and complex, this may be interesting for large-scale benchmarking, and might help in the design of new algorithms.

## C.8. Acknowledgments

The authors would like to thank the reviewers for insightful comments. Codes used (GNNB and manifold learning) available upon request to R. N. and M. B. .