The Multiclass ROC Front method for cost-sensitive classification

Simon Bernard[a], Clément Chatelain[b], Sébastien Adam[1a], Robert Sabourin[c]

[a] *Université de Rouen, LITIS (EA 4108), BP 12 - 76801 Saint-Étienne du Rouvray, France*
[b] *INSA de Rouen, LITIS (EA 4108), BP 12 - 76801 Saint-Étienne du Rouvray, France*
[c] *Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle, École de Technologie Supérieure, Université du Québec, Montreal, Canada*

Abstract

This paper addresses the problem of learning a multiclass classification system that can suits to any environment. By that we mean that particular (imbalanced) misclassification costs are taken into account by the classifier for predictions. However, these costs are not well known during the learning phase in most cases, or may evolve afterwards. There is a need in that case to learn a classifier that can potentially suits to any of these costs in prediction phase. The learning method proposed in this work, named the Multiclass ROC Front (MROCF) method, respond to this issue by exploiting ROC-based tools through a multiobjective optimization process. While this type of ROC-based multiobjective optimization approach has been successfully used for two-class problems, it has never been proposed in real-world multiclass classification problems. Experiments led on several real-world datasets show that the MROCF method offers a major improvement over a cost-insensitive classifier and is competitive with the state-of-the-art cost-sensitive optimization method on all but one of the 20 datasets.

Keywords: Multiclass Classification, Cost-Sensitive Classification, ROC optimization, Multi-Objective Optimization, SVM classifier.

1. Introduction

Many real-world classification problems are naturally *cost-sensitive*, such as medical diagnosis or fraud detection for example. In these cases, misclassification costs are defined, that can be very different from each others and/or that can significantly evolve in the future. This context of imbalanced costs can stem from the application itself, or from the imbalanced nature of datasets ([1]).

Consider a K-class classification task with classes in $\{\lambda_1, \lambda_2, \ldots, \lambda_K\}$. During prediction, $K \times (K-1)$ different types of prediction errors can occur. They are generally summarized in the so-called confusion matrix.

[1]Corresponding author. Tel.: +33 2 3295 5210; fax: +33 2 3295 5022. E-mail address: Sebastien.Adam@litislab.eu (S. Adam)

In a cost-sensitive framework, a misclassification cost is associated to each kind of error. Thus, the overall loss function to be minimized on a given dataset $\mathcal{D}$ to be classified is computed as follows:

$$\mathcal{L}_{\mathcal{D}}(\mathcal{C}) = \sum_{i=1}^{K} p(\lambda_i) \left(\sum_{j=1, i \neq j}^{K} \varepsilon_{ij} c_{ij} \right) - \sum_{i=1}^{K} p(\lambda_i) \varepsilon_{ii} c_{ii} \tag{1}$$

Where $p(\lambda_i)$ is the prior probability of class λ_i, $\mathcal{E} = \{\varepsilon_{ij}\}$ are the confusion rates between class λ_i and class λ_j, *i.e.* the proportion of instances belonging to λ_i wrongly classified with λ_j, and $\mathcal{C} = \{c_{ij}\}$ is the matrix of misclassication costs. Here, $p(\lambda_i)$ and $\mathcal{E}$ are estimated on the dataset $\mathcal{D}$, while the misclassification cost matrix comes intrinsically from the real world problem. The second term in this equation, representing the profits, is generally ignored since the typical cost-sensitive framework consider the diagonal elements of $\mathcal{C}$ to be set to zero (*i.e.* there is no cost associated to good predictions).

One can distinguish two configurations for tackling cost-sensitive problems, depending on the point the misclassification costs are known. If they are known at the learning stage, standard learning methods can handle imbalanced misclassification costs by customizing the learning algorithms, *e.g.* by modifying the objective function in the optimization process ([2, 3]), or by under-sampling the dataset ([4]).

However, most of the real-world learning problems do not naturally come with these costs, and making the assumption that they are precisely known during the learning phase is often unrealistic.

If they are unknown at the learning stage, the problem is more difficult and the system has to foresee this uncertainty. We call this problem "Cost-Sensitive Classification" in opposition to the "Cost-Sensitive Learning" problem mentioned above that considers that the cost are known for the learning. In a CSC framework, the objective is to build a machine that once learned should adapt itself to particular misclassification costs (*i.e.* the environment) at the decision stage. In such a context, the ideal learning machine should minimize the loss function whatever the cost matrix, by minimizing the following criterion on a learning dataset $\mathcal{D}$:

$$\mathcal{J} = \int_{\mathcal{C}} \mathcal{L}_{\mathcal{D}}(\mathcal{C}) d\mathcal{C} \tag{2}$$

where $\mathcal{L}_{\mathcal{D}}$ is the loss defined in equation 1.

Two strategies can be used to adapt a system to the environment : (i) learning a single classifier that performs well whatever the costs (typically using AUC-based approaches [5, 6, 7, 8]), and choosing the decision parameters that suit a particular environment (called the single-model approach); (ii) learning a pool of classifiers, each one being specialized to a particular environment, and selecting the one that suits the decision conditions (called the multi-model approach).

In [9], the second strategy was adopted within the context of two class classification problems. The proposed multi-model approach, called ROC front, relies on a multi-objective optimization strategy that considers both kinds of classification errors (false positive and false negative) as objectives, in order to build

a Pareto-optimal pool of classifiers. During the decision stage, the best classifier can be selected according to the environment using iso-performance lines. This approach has shown better performance than state-of-art single-model approaches ([9, 10]). However, its straightforward generalization to multiclass problems raises computational issues since the number of objectives grows quadratically with the number of classes, making the optimization stage very time consuming. As far as we know, no solution to this issue has ever been proposed yet.

Thus, the contribution of this paper is a multi-model method for multiclass cost-sensitive classification. The approach is based on a multi-model strategy coupled with a pairwise decomposition of the problem in order to break the computational barriers. In prediction, when costs are given, the adequate classifiers are selected and combined to make the resulting model suit the best possible to these costs. A deep analysis is presented to validate the method, through a proof of concept on a toy problem and a rigourous experimental procotol. In particular, it includes a comparison with cost-insensitive multiclass classifiers (Naive Bayes classifiers and multiclass SVM) and the state-of-the-art single-model cost-sensitive method presented in [11].

The rest of this paper is organized as follows. Section 2 is dedicated to the related works and the motivation of this work. Section 3 details the principles and the functioning of the proposed method. Section 4 presents the experimental protocol led for evaluating the method and for comparing it to state-of-the-art methods, and analyzes the obtained results.

2. Related works and motivation

In this section, we review methods that have been proposed to respond to cost-sensitive classification problems (*i.e.* for which the misclassification costs are not known for training but only for prediction). We distinguish single model approaches that learn a single classifier expected to suit to every misclassification costs scenario, from multi-model methods that learn a pool of classifiers, each one being specialized in a specific misclassification costs scenario.

In both cases, an adaptation process is required in the prediction phase when $\mathcal{C}$ is known. For the first approach it usually consists in optimizing some decision parameters, while for the second approach it consists in selecting the right classifier according to the misclassification costs.

In the next subsections, we detail the main principles used for both approaches, for two-class and multi-class classification contexts.

2.1. Single-model approaches

2.1.1. Two-class context

A natural way to design a cost-sensitive classification system without knowing the misclassification cost during training is to focus on the optimization of the decision function in the prediction phase. When

using a classifier that supplies output scores, one has to set a decision function to predict a class from these scores. In two-class classification tasks, it is typically done by applying a threshold on the outcome for the positive class, below which the negative class is predicted. This threshold is a decision parameter that can be optimized on a validation dataset to better fit the misclassification costs once they are known for predictions. A typical example of such an approach is given in [12]. However, the main disadvantage of these methods is that the classifier used upstream is not trained in a cost-sensitive manner but rather via error-based criterions. As explained in [1], this can lead to an important "gap between the cost-sensitive evaluation and the objective of training". In other words, it seems that using a regular (cost-insensitive) classifier and focusing only on the decision parameters to make it suits to misclassification costs is not as efficient as using specific cost-sensitive methods.

A popular way to bridge this gap is to use a ROC-based criterion for the training of a classifier. The ROC space is a powerful tool for tackling cost-sensitive issues, since it allows to represent a classifier via its ability to recognize both classes: the first class through the True Positive Rate (TPR) and the second class through the False Positive Rate (FPR) (cf. Figure 1). The ROC space can be used for finding the best

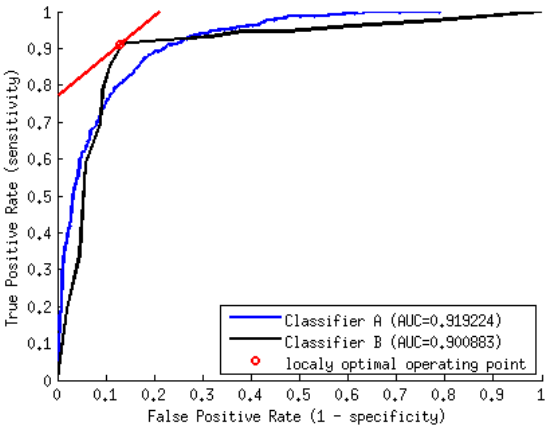


Figure 1: Illustration of two ROC curves with their respective AUC values. The red line is an iso-performance line with a slope equal to 1.0, that is to say for $c_{12} = c_{21}$

decision threshold for a given set of costs, using *iso-performance lines* ([13]) defined by a slope given by:

$$slope = \frac{p(\lambda_1).c_{12}}{p(\lambda_2).c_{21}} \tag{3}$$

The best threshold is given by the operating point for which the iso-performance line is tangent to the ROC curve, as illustrated in Figure 1.

The rationale behind using a ROC-based criterion for learning is to optimize the performance over all possible trade-off "FPR/TPR", that is to say over all possible misclassification costs scenario. Thus, using such an approach implies to modify the learning algorithm to make it optimize a ROC-based measure instead

of a traditional error-based criterion. The AUC measure is usually used for that purpose and some popular learning methods have been modified so they optimize AUC for learning, e.g. Decision Trees ([5]), Boosting ([6]) or SVM ([7, 8]).

However, this approach is sub-optimal since this kind of scalar criterion does not guarantee the resulting classifier to be optimal for all possible misclassification costs. The reason is that such a performance indicator is a resume of the ROC curve taken as a whole and do not consider the curve from a local point of view. Figure 1 illustrates this statement with an example of a ROC curve (the black curve) that exhibits a lower AUC than another ROC curve (the blue curve), but that is locally better (the red point). In that case, the classifier A would be prefered regarding the AUC measure, whereas it would be clearly outperformed by the classifier B in some specific situations of misclassifications costs.

2.1.2. Multiclass context

The two-class context makes possible to efficiently use the ROC space for either learning a classifier on ROC-based criterion or for finding the optimal decision threshold for a given set of costs. The multiclass context is more tricky since a multiclass classifier is typically represented in the ROC space as a point with $K \times (K-1)$ coordinates, each one of them corresponding to one confusion rate: $\xi_{ij}, i, j \in \{1, 2, \ldots, K\}, i \neq j$. As a consequence, the ROC space here is a $K \times (K-1)$ dimensional space. While a two-class classifier generally outputs a single score that can be thresholded to take the decision, a multiclass classifier generally outputs a vector with K different outcomes $h(\lambda_i|\mathbf{x})$, each one being a confidence or a probability that x belongs to the class λ_i :

$$h(\mathbf{x}) = [h(\lambda_1|\mathbf{x}), h(\lambda_2|\mathbf{x}), \ldots, h(\lambda_K|\mathbf{x})]$$

Thus, the prediction can be obtained by applying weights to the $h(\lambda_i|\mathbf{x})$, and by predicting the largest outcomes: for a vector of weights $\mathbf{w} = [w_1, w_2, \ldots, w_K]$, a prediction is obtained with

$$\underset{i=1,\ldots,K}{\arg\max}\ w_i \times h(\lambda_i|\mathbf{x})$$

These weights are often called the operating weights and are usually considered as decision parameters. Evaluating a classifier for all possible $\mathbf{w}$ that produce different outputs is theoretically needed for computing the whole ROC hypersurface. However, as pointed in [14, 15, 11], generating all these possible combinations of weights w_i becomes quickly untractable, even for a reasonable number of classes K.

A way to deal with this complexity is the pairwise decomposition of the multiclass problem. This strategy has been used both for Volume Under the Surface (VUS) estimation in [16, 17, 18] and for decision parameters optimization in [19, 20, 21]. A typical pairwise approach is proposed in [20] where the authors perform a pairwise operating weights optimization and reduce the computational costs by discarding some binary sub-problems. These sub-problems are chosen according to their estimated influence on the result of the overall optimization problem. Figure 2 illustrates the process proposed in [20].

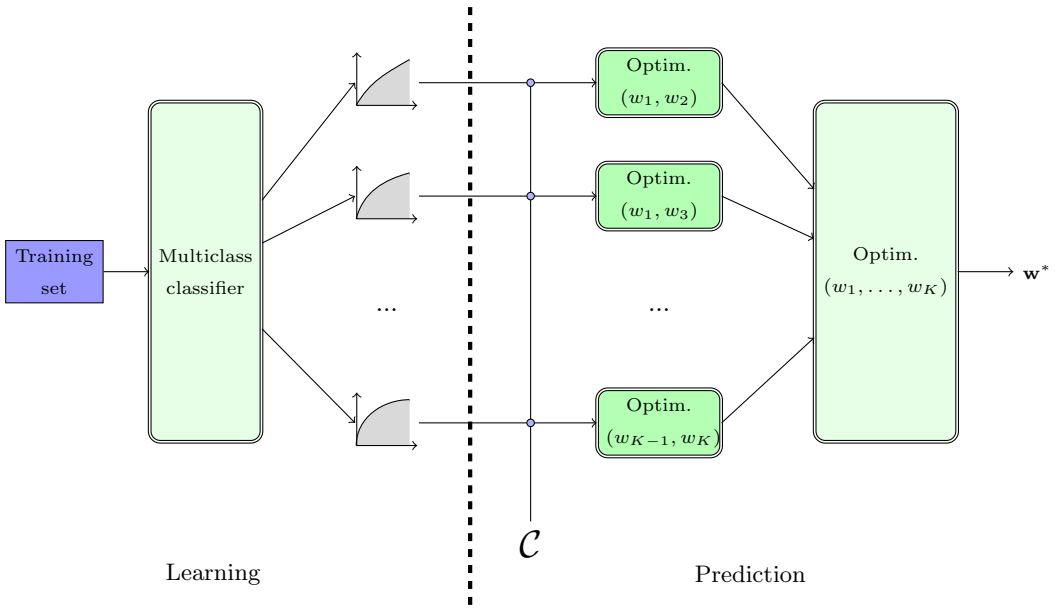


Figure 2: Illustration of a pairwise decomposition approach for cost-sensitive decision parameter optimization in multiclass context ([20])

In [11], the same authors propose a method called ROC skeleton for decision parameter optimization. The key idea is to generate an approximation of the ROC curve/hypersurface by deriving all the possible operating points from the data samples outputs. In two-class classification, these outputs can be ordered by decreasing scores for the positive class, and the enumeration of all possible operating points only required N calculations, N being the number of samples. For multiclass cases, this enumeration becomes untractable as K increases ([15, 11]). The ROC skeleton method relies on the generation of reference operating points from which P new random operatings points are computed. As far as we know, providing that P is sufficiently large, this procedure is the most efficient and tractable multiclass ROC estimation method, and allows to perform decision parameter optimization for cost-sensitive classification tasks.

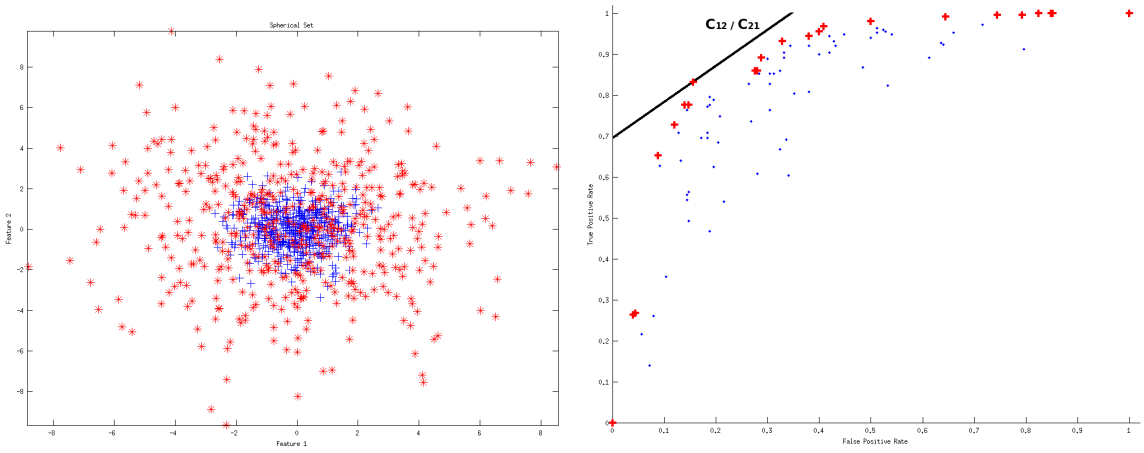
To sum up, the single-model approaches are the most popular for cost-sensitive classification nowadays, mainly because they can be used with any classifier used upstream. With such methods, cost-sensitive efforts are performed at the prediction phase to adapt the model to the environment. A totally different approach for building adaptive systems is to produce a set of models that can handle each possible environment. This strategy is known as multi-model approaches.

2.2. Multi-model approaches

2.2.1. Two-class context

The key idea behind using a multi-model strategy for cost-sensitive classification is to train a pool of classifiers, each one being specialized in a particular misclassification costs scenario, and then to select the one that best suits the costs once they are defined. The motivation behind this strategy is (i) to lighten as much as possible the computational costs of the prediction phase by anticipating multiple environments, and (ii) to overcome the sub-optimality of ROC-based scalar criteria like the AUC measures (cf. Figure 1).

This strategy requires learning algorithms that can be specialized to suit diverse environments. Using such classifiers, a straightforward multi-model approach is to generate a grid of costs pairs, and to train a classifier on each of these environments, leading to a pool of classifiers. Figure 3 gives an illustration of this strategy on a 2D toy problem. The data of the problem are plotted on figure 3(b). A pool of classifiers (here decision trees) has been trained using a simple undersampling method ([4]). Figure 3(b) represents each classifier performance in the ROC space, showing their suitability to different environments. As one can see on this figure, a subset of the resulting classifiers provide worst performance than others from both FP and TP rates point of view. In a multiobjective optimization framework, these classifiers are said to be Pareto-dominated. Removing these dominated classifiers from the pool results in a set called the ROC-front in [9] by analogy with the Pareto-front. Naturally, this set has to be identified using a validation dataset.



(a) the 'Spherical' toy problem: positive instances are in blue and negative instances in red

(b) ROC plot of 100 cost-sensitive classifiers. The blue points are the dominated classifiers and the red points are the non-dominated classifiers. The black line is an iso-performance line.

Figure 3: Illustration of the multi-model approach with decision tree classifiers trained on a synthetic 2D toy problem.

In the simple "overproduce-and-choose" strategy described above, a grid search algorithm is applied to build the ROC-front. An alternative strategy to reach this goal consists in using an optimization algorithm

that guides the search based on the performance. To the best of our knowledge, Kupinski et al. were the first to propose such an approach ([22]). Their method leans on a multiobjective genetic algorithm called Niched Pareto Genetic Algorithm ([23]) for optimizing linear discriminant and neural network classifiers based on both TP and FP ratios, and for applications in computer-aided diagnosis.

Chatelain et al. have proposed in [9] the 'ROC Front' method to form a pool of SVM classifiers using the well-known multiobjective evolutionary algorithm NSGA-II ([24]). In this principle, each SVM is represented by its hyperparameter values that allows to make the resulting model fit to the misclassification costs.

Khreich et al. have proposed in [25] a method named Iterative Boolean Combination (IBC) that iteratively explores the solutions space by combining several classifiers with boolean operators. The same approach have been then applied in [10] on an ensemble of classifiers obtained with a multiobjective genetic algorithm upstream. A first ROC front is found with the genetic algorithm and its classifiers are then combined to further improve this front. This boolean combination principle have also been recently used for face recognition in video surveillance in [26]. The proposed method is a skew-sensitive method that adapts an ensemble of classifiers to unknown or evolutive imbalanced class distributions by exploiting the Precision-Recall Operating Characteristic (P-ROC) space, an alternative to the ROC space visualization of performance ([26]).

All these methods allow to obtain a pool of classifiers spread all along a ROC Front. Once the costs are defined for predictions, the choice of the classifier is quite simple and fast: the orthogonal distance from each of the classifiers to one of the iso-performance lines defined by the costs (*cf.* Figure 3(b)) are computed and the classifier that minimizes this distance is retained.

2.2.2. Multi-class context

To the best of our knowledge, the paper [14] of Everson and Fieldsen describes the only work in the literature that has adopted a multi-model strategy in the multiclass case. They discuss the multiobjective optimization paradigm in the generic K-class case and propose a multiobjective evolutionary algorithm for estimating the Pareto hypersurface. The principle is to jointly optimize the $K(K-1)$ objectives (i.e. the $K(K-1)$ confusion rates), and to retain the best solutions (classifiers). They illustrate the principles of their method on a 3-class synthetic dataset. For cases with a greater number of classes, such a method faces a computational complexity issue since the number of objectives becomes too high to be optimized (one for each possible error between two classes). To the best of our knowledge, there does not exist any work that have applied such a method on real-world multiclass problem with more than 3 classes. For this reason, single-model approaches are still largely prefered for tackling multiclass cost-sensitive classification tasks.

3. The Multiclass ROC Front (MROCF) approach

The previous section has explained the advantages of multi model approaches over single model approaches. However, the generalization of the former to multiclass problems raises computationnal issues. The pairwise decomposition principle used in [20] could be used to overcome this complexity issue and could allow to perform two class ROC-based learning with multiobjective optimization techniques. In this work, we propose to combine the multi-model framework with the pairwise decomposition principle mentioned above. Using such a decomposition, a smart ROC analysis of pairwise classification results allows to concentrate the optimization efforts on the most difficult sub-problems. Some pairs of classes may even be discarded from the optimization process, as it is done in [20]. Therefore, the computational costs are reduced to much less than $\frac{K\times(K-1)}{2}$ times the cost of a binary sub-problem. This strategy is illustrated in [20] on a handwritten character recognition problem where the number of binary problems is reduced from 325 to 22 pairwise sub-problems. The difficult two-class sub-problems being identified, they can be optimized in the ROC space using one of the two-class multi-model approaches described in the previous section. In this work, a set of ROC fronts are computed using the multiobjective optimization strategy described in [9]. The adaptation to the misclassification costs is then achieved through the selection of classifier in each ROC front. These pairwise classifiers are then recombined using a traditional approach to produce a multiclass cost-sensitive approach called Multiclass ROC Front (MROCF).

In the following, we describe both the learning and the prediction phases. All along this section, a proof of concept is given on a synthetic toy dataset proposed in the PRTools matlab toolbox ([27])[2]. The dataset is made of 8 classes of 2D points, as shown in Figure 4. This dataset, called *Synth8* in the rest of the paper, is useful to analyze the behaviour of the MROCF method as four pairs of classes present strong overlap. As a consequence, when decomposing the multiclass dataset in $\frac{8\times(8-1)}{2} = 28$ pairwise subsets, one already knows that 4 of them are much more difficult than the 24 others. In that way, we can focus on these particular sub-problems to see if the MROCF method behaves as expected regarding the misclassification costs.

3.1. Learning phase (without the misclassification costs)

We now detail the MROCF learning phase. Figure 5 gives a simplified illustration of this stage with the four main steps of the learning procedure:

1. Pairwise decomposition of the training set resulting in $\frac{K(K-1)}{2}$ new training subsets.
2. Learning of $\frac{K(K-1)}{2}$ default classifiers, each one on a binary subset.
3. Pairwise ROC analysis using predictions on a validation dataset in order to evaluate the intrinsic confusion in each pairwise sub-problems.

[2]the matlab function used is the *gendatm* function with the same number of instances in each of the 8 classes.

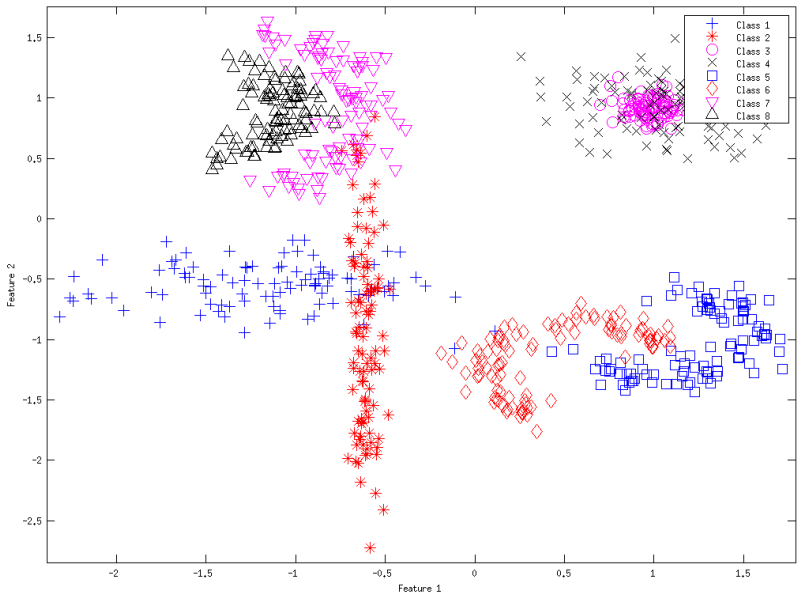


Figure 4: Synthetic dataset made of 8 classes of 2D points ([27])

4. ROCF optimization on pairwise sub-problems w.r.t. these intrinsic confusions.

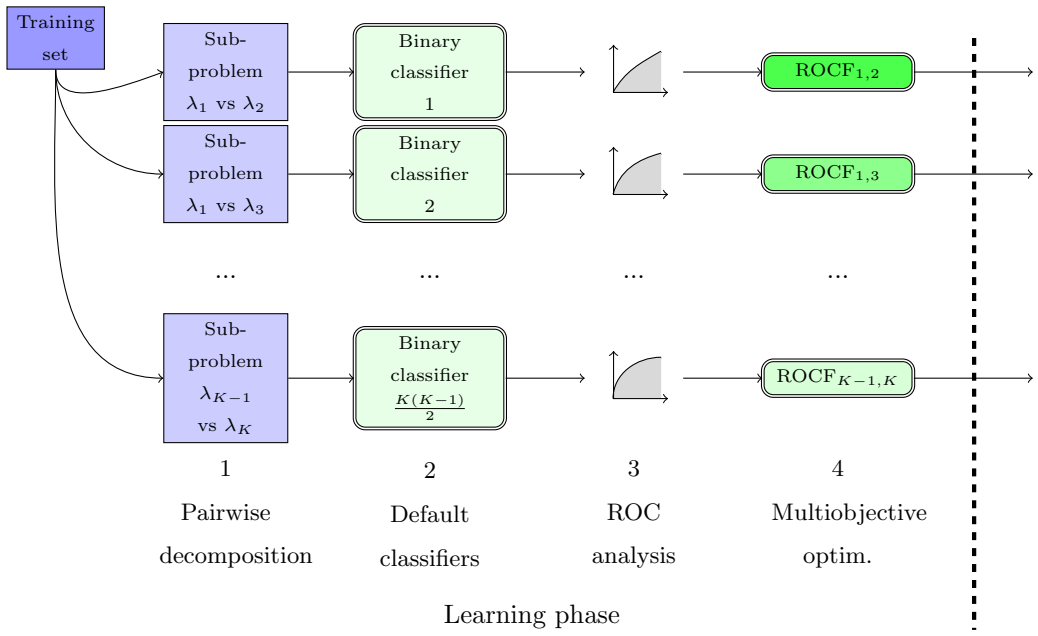


Figure 5: Illustration of the learning stage of the MROCF optimization procedure

The whole learning procedure is given in Algorithm 1. The first step of this process, the pairwise decomposition, has already been discussed in section 2, so we will not cover it again in this section. The

Algorithm 1 *MROCF learning*

Require: T a training set

Require: V a validation set

Require: K the number of classes

Require: $\mathcal{L}$ a learning function for the component classifiers

Require: Θ_{ref} a hyperparameter vector for learning the default classifiers with $\mathcal{L}$

Ensure: $(H_{ij})_{i,j\in[1..K]}$ a family of $\frac{K(K-1)}{2}$ ensembles of classifiers (ROC Front)

Ensure: $(h_{ij}^{ref})_{i,j\in[1..K]}$ an ensemble of $\frac{K(K-1)}{2}$ default classifiers

Ensure: $(PSP_k)_{k\in[1..\frac{K(K-1)}{2}]}$ a set of pairwise sub-problems sorted according to increasing AUC

1: $(H_{ij})_{i,j\in[1..K]} \leftarrow \emptyset, \forall\ i,j \in [1..K]$

{*Step 1 - pairwise decomposition:*}

2: $(t_{ij})_{i,j\in[1..K]} \leftarrow$ pairwiseDecomposition(T,K)

3: $(v_{ij})_{i,j\in[1..K]} \leftarrow$ pairwiseDecomposition(V,K)

4: **for** i from 1 to $K-1$ **do**

5: **for all** j from $i+1$ to K **do**

6: $h_{ij}^{ref} \leftarrow \mathcal{L}(t_{ij}, \Theta_{ref})$ {*Step 2 - default classifiers*}

7: add h_{ij}^{ref} to H_{ij}

8: $AUC_{ij} \leftarrow$ ROCAnalysis(h_{ij}^{ref},v_{ij}) {*Step 3 - ROC analysis*}

9: **end for**

10: **end for**

{*Step 4 - multiobjective optimization:*}

11: $(PSP_k)_{k\in[1..\frac{K(K-1)}{2}]} \leftarrow$ sort($(AUC_{ij})_{i,j\in[1..K]}$)

12: $k \leftarrow 1$

13: **while** stoppingCriterion(AUC_{ij}) **do**

14: $(i,j) \leftarrow PSP_k$

15: $H_{ij} \leftarrow ROCF(\mathcal{L}, t_{ij}, v_{ij})$

16: $k \leftarrow k+1$

17: **end while**

three other steps will be detailed in the following.

3.1.1. Learning default classifiers

In the second step of the MROCF learning procedure, a *default* classifier is learnt from each pairwise sub-problem, each one independantly from the others. It is trained in a traditional cost-insensitive manner, *i.e.* using a regular classification error based criterion and/or with default hyperparameter settings. It will

produce a baseline from which the ROC optimization is performed. The actual purpose of these default classifiers is not necessarily to perform well, but rather capture the overall confusion in each binary sub-problem. These confusion estimations are then used to guide the optimization phase as detailed in the next section.

As a consequence, this step is not considered to require a significant amount of computational resources. Figure 6 shows the decision frontiers of four Support Vector Machine (SVM) that can be used as default classifiers, which in this case operate with default hyperparameter values.

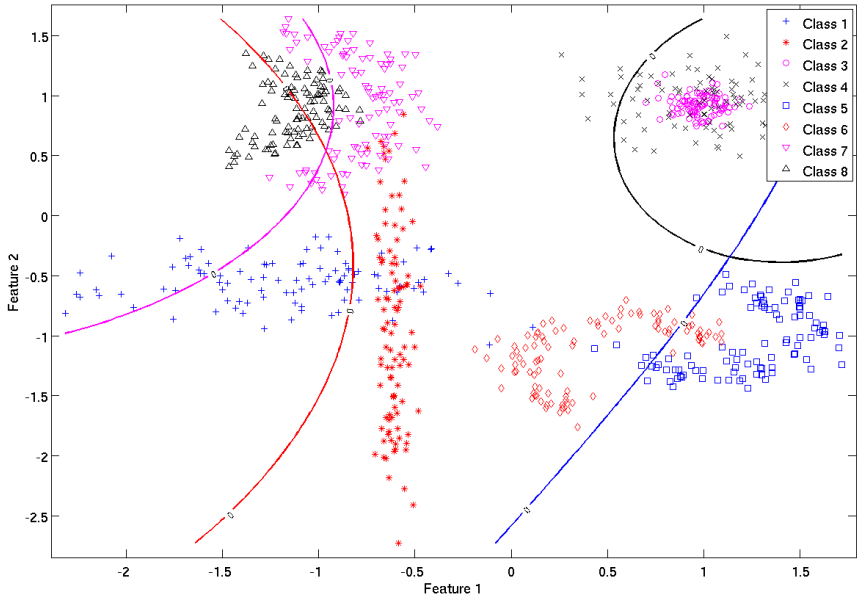


Figure 6: Examples of decision frontiers obtained with 4 default classifiers on the 4 most difficult pairwise sub-problems in the Synth8 dataset, *i.e.* "class 1 vs class 2" (red frontier),"class 3 vs class 4" (black frontier), "class 5 vs class 6" (blue frontier) and "class 7 vs class 8" (pink frontier). These decision frontiers have been obtained with SVM classifiers (with Radial Basis Function), with hyperparamerter values $\gamma = 0.1$, $C_+ = 1.0$ and $C_- = 1.0$.

3.1.2. Pairwise ROC analysis

The third step of the MROCF procedure is a classical two-class ROC analysis performed on each sub-problem. The purpose of this stage is to evaluate the global performance rather than focusing on the local weaknesses in the ROC space. Thus, the drawbacks of the AUC measure pointed out in Section 2 are not an obstacle here: high AUC values indicate that the 2 classes are quite easily discriminated whereas low AUC values indicate that there is a room for improvements in the ROC space, and that the subproblem should benefit from a ROC optimization.

The key idea behind these pairwise ROC analyses is to allow a simplification of the ROC optimization performed afterwards. As explained previously, several binary sub-problems can be discarded from this

Classes	3-4	1-2	7-8	5-6	2-7	2-8	...
AUC	0.50 ± 0.04	0.88 ± 0.04	0.90 ± 0.01	0.93 ± 0.01	0.99 ± 0.01	0.99 ± 0.0	1.0 ± 0.0

Table 1: AUC results on the *Synth8* dataset, obtained by averaging 5 replications of the experiment. These values are presented in the ascending order. The first six pairs of classes have always been ranked in the exact same order for all the runs, while the 22 other pairs have always gave the same AUC of 1.0. This is an expected behavior regarding the nature of the *Synth8* dataset (*cf.* Figure 4).

optimization step. Two procedures can be used for that purpose:

1. The first one is to use a threshold on the AUC values and to simply ignore binary sub-problems for which the AUC is above this threshold. In prediction phase, the default classifier learnt at the previous step can be used for the concerned sub-problem. However, setting this threshold is not an obvious task since AUC values are very different from a problem to another. In [20], it is empirically set for each problem without explaining the choices.
2. The second procedure is to sort the sub-problems according to their increasing AUC values and to perform the ROC optimization following that order. Providing that the learning stage is performed using a limited amount of computational resources, one could process the sub-problems one after the other and stop this step when the computing time is over.

In our experiments (Section 4), the second strategy is retained as it avoids to set a threshold. All the sub-problems are ROC-optimized, in order to evaluate the whole evolution of performance regarding the number of sub-problems processed. As an illustration, Table 1 gives the AUC values obtained on the *Synth8* dataset during these experiments. These results have been averaged over 5 replications of the experiment, and sorted in the ascending AUC order.

3.1.3. ROC-based optimization

As pointed out in Section 2.2.1, there is a few methods that can be used here to perform ROC-based multiobjective optimization. The aim is to obtain an ensemble of classifiers to form the ROC front, based on both FPR and TPR values. For the MROCF method, we choose to use the multiobjective evolutionary algorithm NSGA-II ([24]) as in [9]. The reason is that NSGA-II embeds an efficient mechanism that guarantees the selected classifiers to be spread all along the ROC Front. In that way, it ensures that the largest possible number of cost-sensitive scenario will be handled. We refer the reader to [28] for more details about how multiobjective evolutionary algorithms ensure this repartition in the objective space.

In MROCF, the NSGA-II individuals are classifiers. There is therefore a need to define a representation of these classifiers, according to their cost-sensitive capacities. For example, one can use the *resampling* approach used at the beginning of Section 2.2.1 to make each classifier suit to given misclassification costs. To do it so, each classifier could be represented by a pair of weights, each one used to under-sample a class

([4]). Note however that this method requires each class of the problem to be represented by a large number of instances, since some of them may be strongly under-sampled according to the misclassification costs. This is even more crucial since NSGA-II will need a fraction of these instances to be used to evaluate the objectives as explained below. This makes the under-sampling approach often inadequate. That is one of the reasons why SVM classifiers have been used in this article, since it embed cost-sensitive hyperparameters (more details are given in section 4.1) that can be used as individual representations in NSGA-II.

Once this representation defined, NSGA-II needs the definition of the objectives. In MROCF, objectives are the estimation of the FPR and TPR values. To reduce overfitting in the model selection, classifiers are evaluated through a cross-validation procedure applied on the initial training set. That is to say, 5 estimations of the FPR/TPR values are measured on 5 validation subsets for each individuals (*i.e.* sets of SVM hyperparameters values in our experiments). The mean values of these 5 FPR/TPR pairs are used as the objectives in NSGA-II.

Finally, NSGA-II evolves a population of N classifiers over M generations, depending on the resources allocated to the given sub-problem. The final populations obtained with NSGA-II is made up with N classifiers that are all retained to form the final ROC Front.

Figure 7 illustrates the ROC front for the *Synth8* dataset. It shows the decision frontiers for the '1 vs 2' sub-problem for the two classifiers at the extremity of the ROC front. The first one perfectly classifies the samples from class 1 while the second perfectly classifies the samples from class 2.

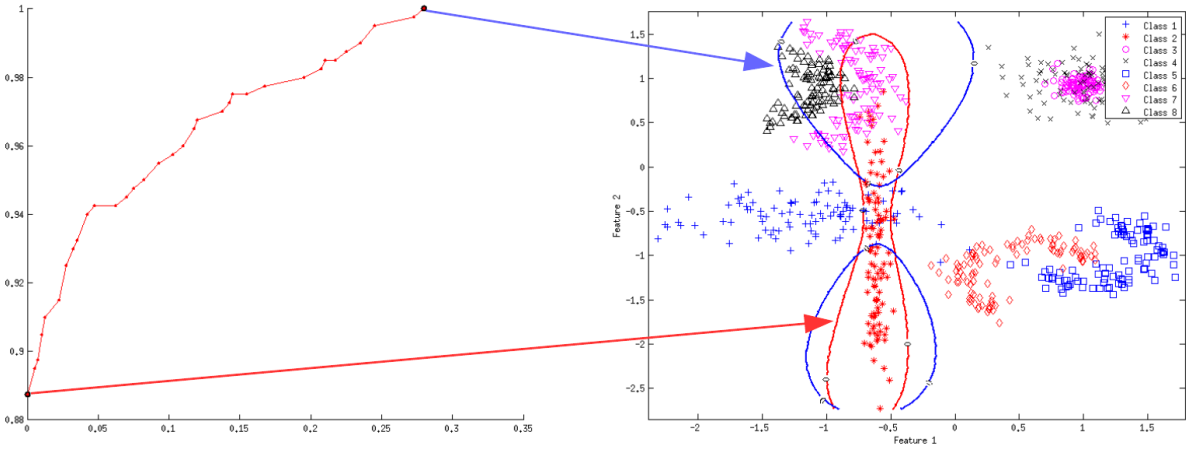


Figure 7: Visualization of two decision frontiers taken from a ROC Front learnt on the 'class 1 versus class 2' sub-problem. The blue frontier (k) is learnt to strictly avoid misclassification error on class 1 and the red frontier (b) to strictly avoid error on class 2.

3.2. Prediction phase (once the misclassification costs are known)

When the misclassification costs are known, the goal is to choose the best classifier for each pairwise sub-problem. For a two-class classification problem, this can be done via the iso-performance lines as explained

in section 2.1.1. For multiclass classification, the choice is more complicated because of the multiclass recombination process that requires to select the best combination of pairwise classifiers among the $N^{\frac{K(K-1)}{2}}$ possibilities, N being the number of classifiers in each ROC Front. Obviously, it becomes rapidly untractable for increasing values of N and K.

In the MROCF method, two tricks allow to break this complexity. First, a greedy algorithm is used for sequentially choosing the classifiers of each ROC front, according to their decreasing AUC. All the ROC Front classifiers are evaluated through their multiclass performance when combined with other pairwise classifiers. For sub-problems that have already been processed, the retained classifier is taken into account in the combination, whereas for remaining sub-problems (with a greater AUC), the default classifier is used for combination. The complexity thus falls from $N^{\frac{K(K-1)}{2}}$ to $N \times \frac{K(K-1)}{2}$. Furthermore, the sub-problems which have been discarded because of their AUC equal to 1.0 do not need any choice. Therefore, only K_0 among $\frac{K(K-1)}{2}$ sub-problems require an evaluation, leading to a complexity of $N \times K_0$.

Algorithm 2 gives all the details on how to make the multiclass classifier suits to a given misclassification costs matrix, without discarding any pairwise sub-problems.

4. Experiments

This section details the experiments led to validate the proposed multiclass ROC Front method. Before analyzing and discussing the results, all the details on the experiment settings and protocol are given in the next two subsections.

4.1. Experimental settings

4.1.1. MROCF Component classifiers

For these experiments, the ROC Front component classifiers are SVM classifiers with Radial Basis Function (RBF) kernel. The main reason of this choice is that SVM with RBF is one of the most powerful and accurate among state-of-the-art methods. A secondary reason is that it presents hyperparameters that intrinsically allow to control the way each class of instances is taken into account during the learning process. More precisely, these classifiers are controled by 3 hyperparameters, noted C_+, C_- and γ. The C_+ and C_- values are the penalty parameters associated with the positive and negative class of a two-class dataset, and allow to control their respective influence on the learning: the lower C_+ compared to C_-, the more prediction errors on the positive class will be allowed by the resulting model to ensure a high accuracy on the negative class, and reciprocally. As for γ, it can be seen intuitively as a parameter for defining how far the influence of a single training example reaches; with low values meaning "far" and high values meaning "close".

For these reasons, SVM with RBF are particularly suitable for the cost-sensitive issue. Note however that the MROCF method can be used with any classifiers that can suit imbalanced misclassification costs.

Algorithm 2 *MROCF prediction*

Require: $\mathcal{C}$ a misclassification costs matrix

Require: T a test set

Require: V a validation set

Require: K the number of classes

Require: $(H_{ij})_{i,j\in[1..K]}$ a family of $\frac{K(K-1)}{2}$ ROC Fronts

Require: $(h_{ij}^{ref})_{i,j\in[1..K]}$ an ensemble of $\frac{K(K-1)}{2}$ pairwise default classifiers

Require: $(PSP_k)_{k\in[1..\frac{K(K-1)}{2}]}$ a set of pairwise sub-problems sorted according to increasing AUC values

Ensure: $\hat{Y}$ a vector of $|T|$ predictions

$H_{multi} \leftarrow (h_{ij}^{ref})_{i,j\in[1..K]}$
$P \leftarrow H_{multi}(T)$
for k from 1 to $\frac{K(K-1)}{2}$ **do**
  $(i,j) \leftarrow PSP_k$
  $H_{eval} \leftarrow H_{multi}$
  $\mathcal{J}_0 \leftarrow \text{loss}(H_{eval}(V),\mathcal{C})$
  for all classifiers $h \in H_{ij}$ **do**
    replace h_{ij}^{ref} by h in H_{eval}
    $\mathcal{J}(h) \leftarrow \text{loss}(H_{eval}(V),\mathcal{C})$
  end for
  $h_{ij} \leftarrow \arg\min_h \mathcal{J}(h)$ {*Selection of the classifier that minimizes* $\mathcal{J}$}
  replace h_{ij}^{ref} by h_{ij} in H_{multi}
  $P(i,j) \leftarrow h_{ij}(T)$ {*Pairwise output scores for* T}
end for
$Scores \leftarrow \text{pairwiseToMulticlass}(P)$
for all k from 1 to $|T|$ **do**
  $\hat{Y}(k) \leftarrow \arg\max_i Scores(k,\lambda_i)$
end for
return $\hat{Y}$

4.1.2. Learning MROCF and comparing with state-of-the-art methods

For a comparison purpose, a first classical multiclass SVM classifier is learnt on each multiclass dataset, using a 'one-vs-one' approach. A grid search method is traditionally used for optimizing the hyperparameter of such SVM classifiers. However, as explained in [29], a random search approach that randomly selects the parameter settings at each try exhibits better results in average. Thus, a 8000 trials random search has been run on each dataset, that is to say as much trials as in the MROCF procedure (see below). The retained

hyperparameter vectors are the ones that have given the best error rates in validation, through a 5-fold cross-validation procedure (leading to $8000 \times 5 = 40000$ learnings).

Secondly, the cost-sensitive single-model method ROC skeleton ([11], cf. Section 2.1.2) is applied on the outputs of these multiclass SVM classifiers. We recall that this method is a decision parameters optimization method that takes as inputs the output scores of a regular multiclass classifier and the misclassification costs matrix. This method has been applied through its implementation in the perClass Matlab toolbox ([30]).

In addition to these multiclass SVM classifiers, our MROCF method is compared to two types of Naive Bayes (NB) classifiers: a traditional multiclass cost-insensitive NB classifier, and its cost-sensitive version, optimized with the ROC Skeleton method as explained above. The multiclass NB classifier has been run through the PRTools matlab toolbox ([27]), with a cross-validation procedure for optimizing the number of bins used during learning.

As for the MROCF method, the default SVM classifiers have been trained with the following hyperparameter values: $\gamma = 1/M$ (M being the number of features), $C_+ = 1$, and $C_- = 1$. It is obviously sub-optimal and one could argue that it could produce classifiers with low accuracies. However, since the role of these default classifiers is to perform pairwise AUC analysis and to propose room for improvements in the ROC space, their accuracies are not critical here. Then, the optimization algorithm detailed in Section 3 has been applied by making NSGA-II evolving 200 generations of 40 individuals each (which equals to 8000 trials, *i.e.* $8000 \times 5 = 40000$ multiclass SVM classifiers on each dataset). Each of these individuals corresponds to a SVM classifier represented through its 3 hyperparameter values (γ, C_+ and C_-) along with a decision threshold value noted t (potentially different from 0.0). In our experiments, no further effort has been spent to tune the NSGA-II hyperparameters, despite the potential benefit in convergence speed that can be brought by a fine tuning. As suggested in [24], they have been set at the following values : $pc = 0.9, pm = 0.1, \nu c = 10, \nu m = 20$.

4.2. Experimental protocol

4.2.1. Datasets

Table 2 describes the multiclass datasets used in this experiment. Except for the *Synth8* and MNIST datasets, all these datasets have been selected from the UCI repository ([31]). The MNIST dataset is a well-known handwritten digits dataset from which 21 features has been extracted following a simple greyscale multi-resolution pyramid technique as in [32]. These datasets have been chosen for this experiment as they present various numbers of classes and various class distributions, what allows to study different cases in terms of complexity and computational costs.

For evaluating the MROCF method, all the datasets are preprocessed as explained below.

Datasets	Classes	Instances	Features	$\frac{K(K-1)}{2}$
Abalone *	7	3294	8	21
Artificial-char	10	3406	7	45
First-order-theorem	6	3059	51	15
MFeat-fourier	10	2000	76	45
MFeat-morphological	10	2000	6	45
MFeat-zernike	10	2000	47	45
MNIST	10	10000	21	45
Page-blocks	5	5473	10	10
Pendigits	10	10992	16	45
Satimage	6	2859	36	15
Segment	7	2310	19	21
Solar-Flare	6	1389	12	15
Synth8	8	4000	2	28
Thyroid-dis	5	2800	26	10
Thyroid-sick	5	2800	26	10
Turkiye-student	13	5820	32	78
Vehicle	4	946	18	6
Yeast *	4	1299	8	6
Wine-red *	4	1571	11	6
Wine-white *	5	4873	11	10

Table 2: Datasets description

First, the multiclass datasets are split in training/test subsets with a traditional stratified 5-folds cross-validation procedure. It produces 5 different partitions in training and test subsets, noted Tr_k and Ts_k, for $k = 1, \ldots, 5$. Second, since our framework is pairwise, we need to decompose the multiclass training sets in $\frac{K(K-1)}{2}$ binary subsets. We denote these binary subsets by Tr_k^{ij}, with $i = 1, \ldots, K-1$ and $j = i+1, \ldots, K$. Third, for ensuring NSGA-II to compute reliable evaluations of FPR and TPR values, another 5-fold cross-validation procedure is performed on all the Tr_k^{ij}. These 5 pairs of training and validation subsets are noted Tr_{kl}^{ij} and Tv_{kl}^{ij}, with $l = 1, \ldots, 5$.

With this experimental protocol, it is required that every class of the datasets is represented by a minimum number of 50 instances. The reason is that the double 5-fold cross-validation preprocess divides by 10 the number of instances of each class available in the Tv_{kl}^{ij}. For that reason, some datasets have been reduced by

removing the classes represented by less than 50 instances (these datasets are noted with asterisk in Table 2). In that way, after applying the two 5-fold cross-validation procedures, at least 5 instances of each class is available in our validation sets.

4.2.2. Evaluation procedure

The learning phase outputs $\frac{K(K-1)}{2}$ default SVM classifiers and $\frac{K(K-1)}{2}$ ROC Fronts each of which made up of 40 SVM classifiers. Evaluation, then, is made thanks to the pairwise predictions on Ts_k. Precisely, it consists for each sub-problem in selecting the right SVM classifiers regarding the given misclassification costs, as explained in the subsection 3.2. This is done one sub-problem at a time, in the decreasing AUC order. Each time one of these sub-problems is processed, the corresponding predictions on Ts_k obtained with the default SVM are replaced by the ones obtained with the selected SVM, at the condition that it has allowed to improve the validation cost-sensitive performance. Then, the multiclass predictions are computed as for a traditional multiclass SVM classifier ([33]).

In such a way, for a given misclassification costs matrix, $\frac{K(K-1)}{2}$ multiclass confusion matrices are obtained on Ts_k, each one being updated from its predecessor by processing one more binary sub-problem. Performance have been measured on each of these confusion matrices. For reliability concerns, 30 misclassification costs matrices have been randomly generated and an evaluation has been performed on each of them. Results presented in section 4.3 are averaged on these 30 matrices.

4.2.3. Evaluation measures

In the next subsections, the results are presented as values of three different measures:

1. the Error rates
2. the Loss values from Equation 1
3. the Mean Subjective Utility (MSU) score from [34]

This latter measures is a rescaled variant of the loss function given by equation 1. The loss allows to measure the performance in a given misclassification costs scenario but it is dependent on the number of classes and on the costs. As a consequence, it is impossible to compare loss values obtained with different datasets and/or with different values of $\mathcal{C}$. Moreover, as this measure has no known bound, raw loss values are not independantly interpretable. The MSU score has been proposed as a rescaling of the loss measure in order to overcome this issue, resulting in a performance measure that always scales between 0 and 1 (higher is better, contrary to the loss). It is given by the following equations:

$$MSU = \sum_{i=1}^{C} \left(\sum_{j=1, i \neq j}^{C} \xi_{ij} \mathcal{C}'_{ij} \right) \tag{4}$$

with

$$\mathcal{C}' = -\beta_1 \mathcal{C} + \beta_2$$

and β_1 and β_2 verifying

$$\begin{cases} -\beta_1 \left(\sum_{i=1}^{C} p(\lambda_i) \mathcal{C}_{ii} \right) + \beta_2 = 1 \\ -\beta_1 \left(\frac{1}{C} \sum_{i=1}^{C} p(\lambda_i) \sum_{j=1, j \neq i}^{C} \mathcal{C}_{ij} \right) + \beta_2 = 0 \end{cases}$$

The MSU measure is thus more interpretable than the loss value and it facilitates the comparisons from one dataset to another and from one cost-sensitive scenario to another. This can be seen in the next section by looking at the Tables 3 and 4. While the loss values given in Table 3 are not straightforwardly interpretable, the MSU values given in Table 4 all belong to $[0, 1]$, with MSU values close to 1.0 meaning that the cost-sensitive performance are near the optimality.

4.3. Results and discussion

First of all, Table 3 gathers the final Loss values and Table 4 gathers the final MSU results obtained in this experiment. The first two columns give the results for the cost-insensitive classifiers, namely SVM and Naive Bayes (NB) classifiers. The third and fourth columns give the results for the ROC Skeleton methods. And the final column gives the result of the MROCF method after all the binary subproblems have been processed in the ROC-based optimization step.

In addition to these measurements, the comparison between the 5 methods have been assessed with a statistical test. Following the recommendations given in [35], we have used the non-parametric Friedman test along with the Nemenyi post-hoc test. This procedure is based on the ranks of the algorithms on each datasets. That is the reason why, these ranks have been reported in Table 4 (between brackets in each cell). The critical difference diagram given in Figure 8 summarizes the result of this statistical comparison procedure ([35]).

The first observation made from Table 4 is that both the MROCF and the Skeleton methods allow to improve the MSU results in comparison with cost-insensitive methods, for all the dataset but one, namely *Pendigits* (explanations about this pathological case are given below). This illustrates that traditional cost-insensitive classifiers, as accurate as they may be, fail to tackle cost-sensitive classification problems.

The second interesting result is that the MROCF approach is competitive with the Skeleton method, and gives the best mean rank over all the dataset among the five methods, and particularly among the three cost-sensitive methods. Note however that, if the difference between $MROCF$ and NB_{Skel} is significant, it is not the case between $MROCF$ and SVM_{skel}, according to the Nemenyi test, as shown on Figure 8. Nevertheless, the MROCF method, that we recall performs all the cost-sensitive optimization efforts during training, is better ranked on 14 of the 20 datasets than the SVM_{skel} method, which needs to perform the optimization step in prediction. This highlights from our point of view that the single-model optimization

	Cost-insensitive		Cost-sensitive		
Datasets	NB	SVM	NB_{Skel}	SVM_{Skel}	$MROCF_{full}$
Abalone	0.115 ± 0.002	0.122 ± 0.002	0.094 ± 0.000	0.096 ± 0.002	$\mathbf{0.091} \pm 0.011$
Artificial-c.	0.062 ± 0.002	$\mathbf{0.034} \pm 0.001$	0.058 ± 0.001	$\mathbf{0.034} \pm 0.002$	$\mathbf{0.034} \pm 0.004$
First-o.	0.123 ± 0.005	0.127 ± 0.007	0.109 ± 0.002	$\mathbf{0.107} \pm 0.003$	$\mathbf{0.107} \pm 0.009$
MFeat-f.	0.027 ± 0.002	0.021 ± 0.004	0.025 ± 0.000	0.019 ± 0.005	$\mathbf{0.017} \pm 0.004$
MFeat-m.	0.035 ± 0.001	0.037 ± 0.007	0.032 ± 0.001	$\mathbf{0.028} \pm 0.005$	$\mathbf{0.028} \pm 0.004$
MFeat-z.	0.033 ± 0.002	0.026 ± 0.006	0.030 ± 0.002	0.019 ± 0.004	$\mathbf{0.018} \pm 0.004$
MNIST	0.037 ± 0.001	0.020 ± 0.002	0.036 ± 0.001	$\mathbf{0.018} \pm 0.002$	$\mathbf{0.018} \pm 0.002$
Page-bl.	0.095 ± 0.012	0.092 ± 0.023	0.056 ± 0.004	$\mathbf{0.039} \pm 0.012$	0.064 ± 0.009
Pendigits	0.015 ± 0.001	$\mathbf{0.001} \pm 0.000$	0.016 ± 0.000	$\mathbf{0.001} \pm 0.000$	0.005 ± 0.002
Satimage	0.036 ± 0.004	0.020 ± 0.002	0.033 ± 0.003	$\mathbf{0.018} \pm 0.002$	0.022 ± 0.006
Segment	0.016 ± 0.002	$\mathbf{0.006} \pm 0.002$	0.016 ± 0.002	$\mathbf{0.006} \pm 0.001$	$\mathbf{0.006} \pm 0.002$
Solar-f.	0.088 ± 0.004	0.089 ± 0.006	$\mathbf{0.068} \pm 0.004$	0.078 ± 0.004	0.073 ± 0.012
Synth8	0.015 ± 0.002	0.019 ± 0.005	0.016 ± 0.002	$\mathbf{0.011} \pm 0.001$	$\mathbf{0.011} \pm 0.004$
Thyroid-d.	0.156 ± 0.007	0.142 ± 0.010	$\mathbf{0.097} \pm 0.007$	0.106 ± 0.013	0.100 ± 0.016
Thyroid-s.	0.155 ± 0.008	0.159 ± 0.010	0.110 ± 0.016	0.119 ± 0.006	$\mathbf{0.096} \pm 0.016$
Turkiye-s.	0.067 ± 0.002	0.060 ± 0.002	0.054 ± 0.001	0.046 ± 0.001	$\mathbf{0.044} \pm 0.006$
Vehicle	0.122 ± 0.010	0.081 ± 0.024	0.076 ± 0.036	0.056 ± 0.026	$\mathbf{0.047} \pm 0.021$
Yeast	0.130 ± 0.006	0.116 ± 0.010	0.111 ± 0.003	$\mathbf{0.101} \pm 0.005$	0.107 ± 0.019
Wine-red	0.187 ± 0.005	0.170 ± 0.012	0.152 ± 0.010	$\mathbf{0.135} \pm 0.014$	0.158 ± 0.029
Wine-white	0.162 ± 0.006	0.142 ± 0.003	0.122 ± 0.005	$\mathbf{0.108} \pm 0.004$	0.111 ± 0.021

Table 3: Summary of the Loss results. The values are given in bold.

approach does not fully reach the potential of the proposed multi-model approach for which the classifiers are learnt with cost-sensitive objectives.

The reason why the method fails to improve the MSU results for the *Pendigits* datasets is that the default classifiers do not offer a sufficient room for improvements. One can see that both SVM and NB classifiers already allow to obtain good MSU results although they are not cost-sensitive methods. This shows that the *Pendigits* datasets is very few impacted by imbalanced misclassification costs since only very few prediction errors are made. In this case, the MROCF has actually decreased the performance compared to the default SVM classifiers used altogether before the ROC-based optimization step. To give better insights of this phenomenon, details on the results obtained on this dataset are given in the following. Table 5 gives the AUC evaluations on the 5 first binary sub-problems. These 5 values are very close to the maximum AUC value, *i.e.* 1.0, and all the other sub-problems exhibit higher values (exactly 1.0 for 31 of the 45 sub-

	Cost-insensitive		Cost-sensitive		
Datasets	NB	SVM	NB_{Skel}	SVM_{Skel}	$MROCF_{full}$
Abalone	0.216 ± 0.013 (4)	0.163 ± 0.018 (5)	0.354 ± 0.003 (2)	0.338 ± 0.014 (3)	$\mathbf{0.374} \pm 0.069$ (1)
Artificial-c.	0.379 ± 0.019 (5)	0.656 ± 0.015 (3)	0.420 ± 0.006 (4)	0.657 ± 0.019 (2)	$\mathbf{0.660} \pm 0.044$ (1)
First-o.	0.263 ± 0.029 (4)	0.239 ± 0.041 (5)	0.346 ± 0.013 (3)	0.357 ± 0.019 (2)	$\mathbf{0.359} \pm 0.053$ (1)
MFeat-f.	0.728 ± 0.019 (5)	0.793 ± 0.047 (3)	0.752 ± 0.003 (4)	0.812 ± 0.050 (2)	$\mathbf{0.827} \pm 0.042$ (1)
MFeat-m.	0.645 ± 0.015 (5)	0.643 ± 0.059 (4)	0.679 ± 0.007 (3)	0.705 ± 0.047 (2)	$\mathbf{0.710} \pm 0.048$ (1)
MFeat-z.	0.673 ± 0.021 (5)	0.779 ± 0.059 (3)	0.700 ± 0.025 (4)	0.805 ± 0.045 (2)	$\mathbf{0.817} \pm 0.042$ (1)
MNIST	0.627 ± 0.008 (5)	0.799 ± 0.021 (3)	0.638 ± 0.006 (4)	0.814 ± 0.018 (2)	$\mathbf{0.821} \pm 0.021$ (1)
Page-bl.	0.523 ± 0.059 (5)	0.542 ± 0.115 (4)	0.719 ± 0.021 (2)	$\mathbf{0.807} \pm 0.060$ (1)	0.678 ± 0.045 (3)
Pendigits	0.849 ± 0.008 (4)	$\mathbf{0.994} \pm 0.001$ (1)	0.841 ± 0.001 (5)	0.992 ± 0.002 (2)	0.948 ± 0.021 (3)
Satimage	0.787 ± 0.024 (5)	0.883 ± 0.014 (2)	0.803 ± 0.019 (4)	$\mathbf{0.890} \pm 0.010$ (1)	0.869 ± 0.035 (3)
Segment	0.886 ± 0.017 (5)	0.960 ± 0.016 (3)	0.888 ± 0.017 (4)	0.961 ± 0.006 (2)	$\mathbf{0.962} \pm 0.015$ (1)
Solar-f.	0.474 ± 0.026 (4)	0.467 ± 0.039 (5)	$\mathbf{0.591} \pm 0.024$ (1)	0.534 ± 0.026 (3)	0.564 ± 0.070 (2)
Synth8	0.881 ± 0.016 (3)	0.849 ± 0.040 (5)	0.874 ± 0.016 (4)	0.908 ± 0.008 (2)	$\mathbf{0.911} \pm 0.028$ (1)
Thyroid-d.	0.219 ± 0.036 (5)	0.289 ± 0.052 (4)	$\mathbf{0.517} \pm 0.037$ (1)	0.470 ± 0.066 (3)	0.502 ± 0.082 (2)
Thyroid-s.	0.227 ± 0.041 (4)	0.207 ± 0.048 (5)	0.451 ± 0.081 (2)	0.403 ± 0.029 (3)	$\mathbf{0.522} \pm 0.082$ (1)
Turkiye-s.	0.134 ± 0.028 (5)	0.214 ± 0.027 (4)	0.299 ± 0.016 (3)	0.408 ± 0.017 (2)	$\mathbf{0.422} \pm 0.079$ (1)
Vehicle	0.513 ± 0.041 (5)	0.674 ± 0.094 (4)	0.701 ± 0.142 (3)	0.776 ± 0.104 (2)	$\mathbf{0.814} \pm 0.081$ (1)
Yeast	0.482 ± 0.026 (5)	0.535 ± 0.038 (4)	0.556 ± 0.012 (3)	$\mathbf{0.594} \pm 0.019$ (1)	0.574 ± 0.078 (2)
Wine-red	0.253 ± 0.019 (5)	0.318 ± 0.049 (4)	0.414 ± 0.029 (2)	$\mathbf{0.462} \pm 0.054$ (1)	0.381 ± 0.121 (3)
Wine-white	0.191 ± 0.030 (5)	0.288 ± 0.015 (4)	0.392 ± 0.026 (3)	$\mathbf{0.460} \pm 0.021$ (1)	0.450 ± 0.104 (2)
Mean rank	4,65	3.75	3.05	1,95	1,6

Table 4: Summary of the MSU results. The highest values are highlighted in bold and ranks of the 5 methods are reported in each cell between brackets.

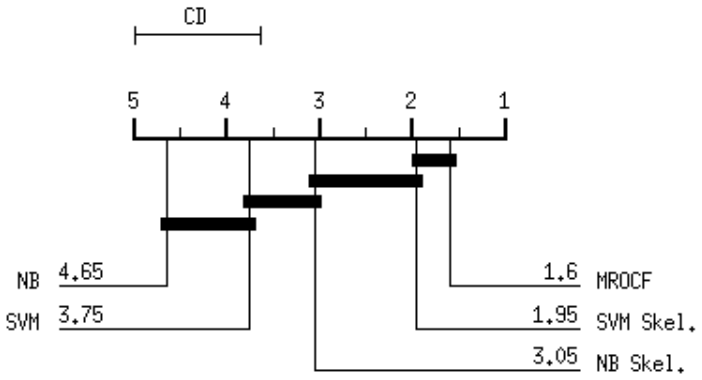


Figure 8: Comparison of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at p = 0.05) are connected. The Critical Difference ("CD") is given in the upper part of the Figure.

Classes	2-4	2-3	4-10	9-10	4-8	...
AUC	0.9976 ± 0.0024	0.9986 ± 0.0016	0.9988 ± 0.0025	0.9989 ± 0.0024	0.9993 ± 0.0016	...

Table 5: AUC results on the *Pendigits* dataset, for the 5 first binary sub-problems when sorted according to their increasing AUC evaluation.

problems). Tables 6, 7 and 8 gives the three first confusion matrices, obtained with the default classifiers and with the first and the second sub-problems processed with the MROCF optimization procedure (*i.e.* for classes '2-vs-4' and '2-vs-3'). The corresponding MSU values are given in the caption of each Table. At the first step, the MSU value has been improved mainly because the SVM classifier selected in the ROC Front has allowed to better recognize the instances belonging to the class '2' (see the bold values in Table 7 that indicate the differences with the Table 6). At the second step on the contrary, the MSU value has been decreased because the selected SVM classifier has introduced more confusion on this same class '2' (see the bold values in Table 8 that indicate the differences with the Table 7). Since, only few prediction errors are made on the whole multiclass problem, the additionnal errors introduced here have a significant impact on the cost-sensitive results.

However, these results given on the *Pendigits* datasets are not very meaningful regarding the performance of the MROCF method, since we recall that the optimization step of the MROCF procedure is not intended to be performed for binary sub-problems for which the ROC analysis has given high AUC values. In the case of the *Pendigits* dataset, all the AUC values are greater than 0.99 as shown in Table 5. They can all be considered sufficiently high to conclude that no cost-sensitive optimization process would allow to improve the performance. It is worth noting, moreover, that the ROC skeleton method has also decreased the MSU results in comparison with the multiclass SVM.

	1	2	3	4	5	6	7	8	9	10
1	221	0	0	0	3	0	0	0	3	1
2	0	217	7	2	0	2	0	1	0	0
3	0	4	225	0	0	0	0	0	0	0
4	0	1	0	210	0	0	0	0	0	0
5	0	0	0	0	225	0	0	3	0	0
6	0	1	0	3	0	206	0	0	1	0
7	0	1	0	0	0	1	210	0	0	0
8	0	4	0	0	0	0	0	222	0	2
9	1	0	0	0	0	0	0	1	209	0
10	0	0	0	0	0	0	0	1	0	210

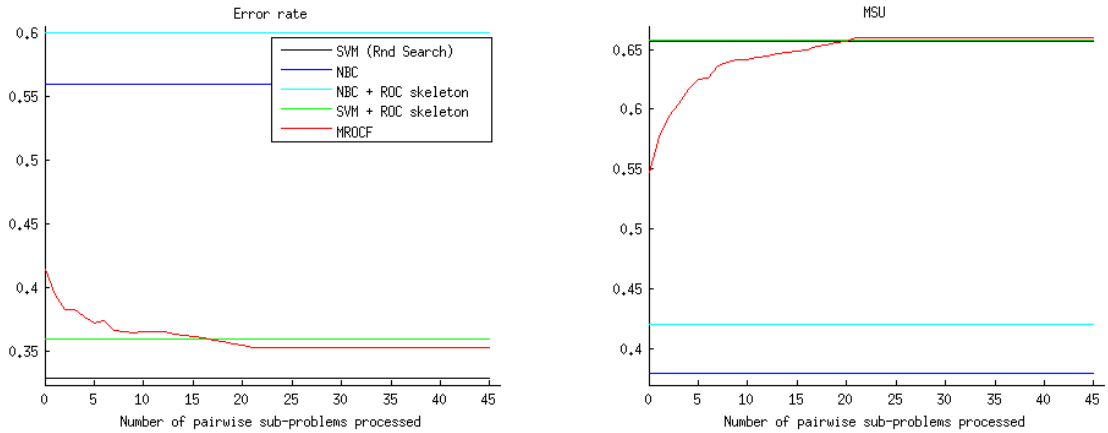
Table 6: Confusion matrix obtained with the default classifiers of MROCF (*i.e.* without any of the binary sub-problems optimized yet). The corresponding MSU is **0.9775**

	1	2	3	4	5	6	7	8	9	10
1	221	0	0	0	3	0	0	0	3	1
2	0	**219**	**8**	**1**	0	**0**	0	1	0	0
3	0	**3**	**226**	0	0	0	0	0	0	0
4	0	1	0	210	0	0	0	0	0	0
5	0	0	0	0	225	0	0	3	0	0
6	0	1	0	3	0	206	0	0	1	0
7	0	**0**	0	0	0	1	**211**	0	0	0
8	0	**3**	0	0	0	0	0	**223**	0	2
9	1	0	0	0	0	0	0	1	209	0
10	0	0	0	0	0	0	0	1	0	210

Table 7: Confusion matrix obtained with the MROCF procedure after the first sub-problem (class 2 vs class 4) have been processed. The corresponding MSU is **0.9797**

	1	2	3	4	5	6	7	8	9	10
1	221	0	0	0	3	0	0	0	3	1
2	0	**216**	**7**	1	0	**4**	0	1	0	0
3	0	**1**	**227**	0	0	0	0	**1**	0	0
4	0	1	0	210	0	0	0	0	0	0
5	0	0	0	0	225	0	0	3	0	0
6	0	1	0	3	0	206	0	0	1	0
7	0	0	0	0	0	1	211	0	0	0
8	0	**2**	0	0	0	0	0	**224**	0	2
9	1	**1**	0	0	0	0	0	1	**208**	0
10	0	0	0	0	0	0	0	1	0	210

Table 8: Confusion matrix obtained with the MROCF procedure after the second sub-problem (class 2 vs class 3) have been processed. The corresponding MSU is **0.9784**

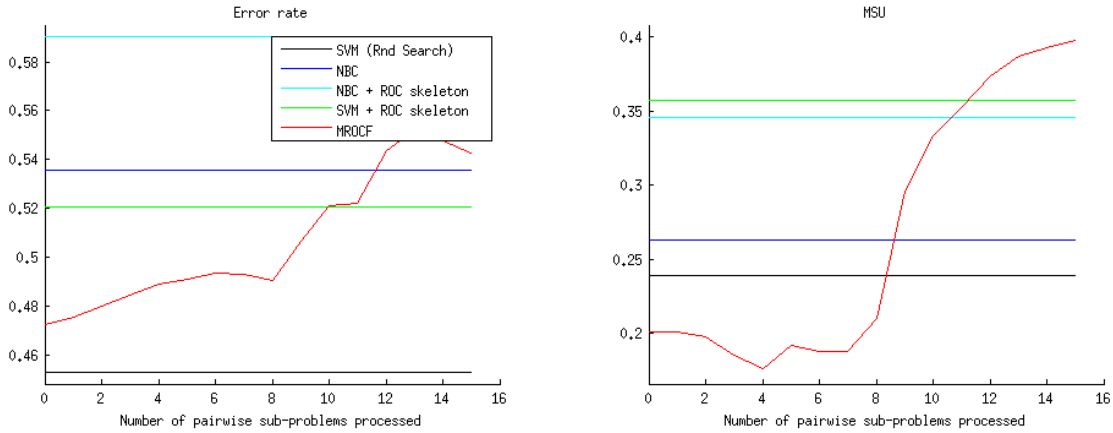


(a) Error rate w.r.t. the number of sub-problems processed (b) MSU values w.r.t. the number of sub-problems processed

Figure 9: MSU and Error rates results for the *Artificial-char* dataset

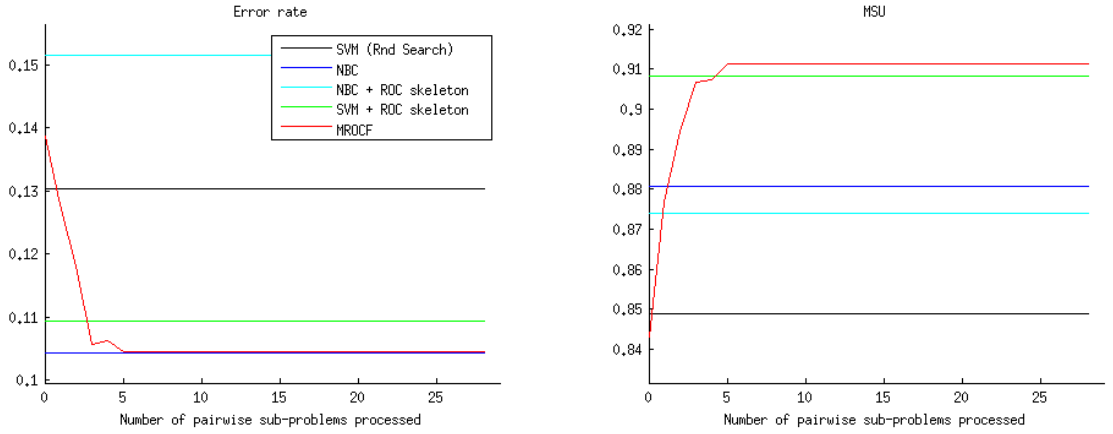
We now give in Figures 9, 10, 11, 12 and 13, five illustrations of the evolution of performances during the whole learning phase. For clarity concerns, we only give these illustrations for 5 of the 20 datasets here. These figures show the evolution of the Error rates and the MSU values with respect to the number of pairwise sub-problems processed inside the MROCF learning phase. It allows to highlight three interesting results with the proposed method:

1. these curves show that processing the sub-problems following the ascending AUC order results in an asymptotical improvement of performance. By this way, the best pairwise improvements are most of the time obtained at first, which makes possible to prematurely stop the optimization process without a significant loss of performance. However, the evolution pattern of these curves can be very different in terms of convergence speed. Therefore, as expected, it seems difficult to deduce from these results how to set a stopping criterion.
2. these curves also highlight the differences between the traditional error rates evaluation and the cost-sensitive evaluation. For the *First-order-theorem* dataset and the *Wine-white* dataset for example, one can observe that better error rates have been obtained with the SVM classifier than with the MROCF method, whereas it is the exact opposite with the MSU results.
3. One can also see that for some datasets, namely *First-order-theorem*, *Turkiye-student* and *Wine-white* here, the MROCF sequential process increases the error rate, *i.e.* introduces more prediction errors, while improving the MSU results at the same time. This illustrates the way the MROCF method manages to focus on the most important classes according to the misclassification costs, at the expense of more errors on the other classes.



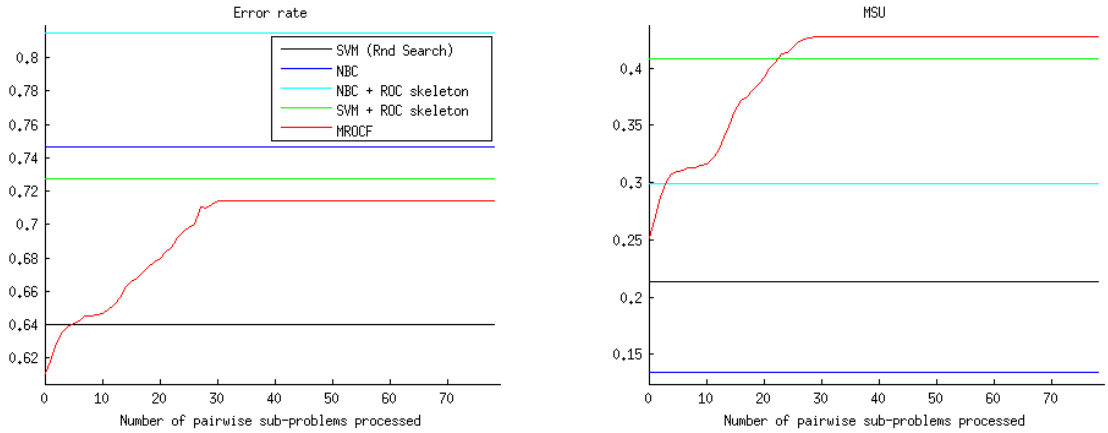
(a) Error rate w.r.t. the number of sub-problems processed (b) MSU values w.r.t. the number of sub-problems processed

Figure 10: MSU and Error rates results for the *First-order-theorem* dataset



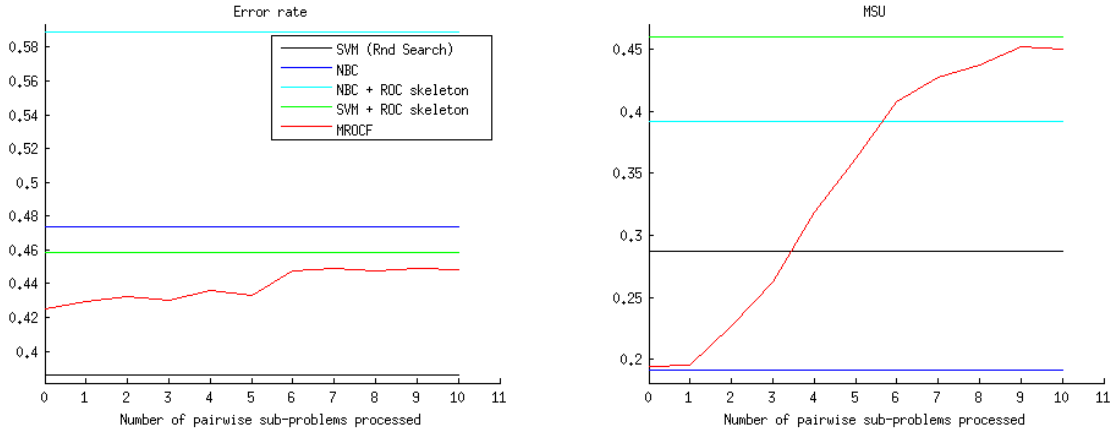
(a) Error rate w.r.t. the number of sub-problems processed (b) MSU values w.r.t. the number of sub-problems processed

Figure 11: MSU and Error rates results for the *Synth8* dataset



(a) Error rate w.r.t. the number of sub-problems processed (b) MSU values w.r.t. the number of sub-problems processed

Figure 12: MSU and Error rates results for the *Turkiye-student* dataset



(a) Error rate w.r.t. the number of sub-problems processed (b) MSU values w.r.t. the number of sub-problems processed

Figure 13: MSU and Error rates results for the *Wine-white* dataset

5. Conclusion

In this work we tackle the problem of learning a multiclass classification system that can suits to any environment, *i.e.* to any data distribution and/or any (imbalanced) misclassification costs. Nowadays, this problem has not been extensively studied in the literature, and existing approaches are either restricted to a small number of classes due to computational issues, either adressed through a heavy optimization procedure during the prediction phase. Our proposed approach called MROCF relies on two main ideas. The first idea is the multi-model principle that trains a pool of classifiers, each one being specialized to a particular environment. The second idea is to circumvent the computational issues of a brut force optimization by a decomposition of the multiclass problem into pairwise subproblems, coupled with a sparse selection of the difficult suproblems.

The experiments led on several real-world datasets, and assessed with a statistical test of significance, show that our method is competitive with existing ones with respect to cost-sensitive criteria. Furthermore, when compared with other multiclass cost-sensitive methods, our approach minimizes the computational efforts made during the prediction phase. Indeed, the multi-model framework enables to foresee a wide range of misclassification cost scenario, which are optimized during the training phase. This anticipation of the misclassification costs scenario is at the price of a more expensive training stage, but allows to quickly adapt the system to a new environment.

Our future works will focus on a smart pairwise subproblem filtering in order to lighten the whole training process (Step 3 of the learning algorithm). The challenge in this step is to select the subset of pairwise subproblems to optimize that maximizes the multiclass performance for any environment, while minimizing the computational resources required.

6. Acknowledgements

This work was partly supported by grants from the ANR-11-JS02-010 Lemon

References

[1] P. Cao, D. Zhao, O. R. Zaane, An optimized cost-sensitive svm for imbalanced data learning., in: J. Pei, V. S. Tseng, L. Cao, H. Motoda, G. Xu (Eds.), PAKDD (2), Vol. 7819 of Lecture Notes in Computer Science, Springer, 2013, pp. 280–292.

[2] J. Xu, Y. Cao, H. Li, Y. Huang, Cost-sensitive learning of svm for ranking, in: J. Frnkranz, T. Scheffer, M. Spiliopoulou (Eds.), Machine Learning: ECML 2006, Vol. 4212 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 833–840.

[3] S. Lomax, S. Vadera, A survey of cost-sensitive decision tree induction algorithms, ACM Comput. Surv. 45 (2) (2013) 16:1–16:35.

[4] C. Drummond, R. C. Holte, C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling, in: ICML Workshop on Learning from Imbalanced Datasets II, 2003, pp. 1–8.

[5] C. Ferri, P. Flach, J. Hernandez-Orallo, Learning Decision Trees Using the Area Under the ROC Curve, in: Proceedings of the 19th International Conference on Machine Learning, 2002, pp. 139–146.

[6] Y. Freund, R. Iyer, R. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, Journal of Machine Learning Research 4 (2003) 933–969.

[7] A. Rakotomamonjy, Optimizing Area Under Roc Curve with SVMs, in: Workshop on ROC Analysis in Artificial Intelligence, 2004, pp. 469–478.

[8] U. Brefeld, T. Scheffer, AUC Maximizing Support Vector Learning, in: In Proc. ICML workshop on ROC Analysis in Machine Learning, 2005.

[9] C. Chatelain, S. Adam, Y. Lecourtier, L. Heutte, T. Paquet, A multi-model selection framework for unknown and/or evolutive misclassification cost problems, Pattern Recognition 43 (3) (2010) 815–823.

[10] J.-C. Levesque, A. Durand, C. Gagne, R. Sabourin, Multi-objective evolutionary optimization for generating ensembles of classifiers in the ROC space, in: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference, 2012, pp. 879–886.

[11] T. Landgrebe, P. Paclik, The ROC skeleton for multiclass ROC estimation, Pattern Recognition Letters 31 (9) (2010) 949–958.

[12] T. Fawcett, An introduction to roc analysis, Pattern Recognition Letters 27 (8) (2006) 861–874.

[13] T. Fawcett, ROC Graphs: Notes and Practical Considerations for Researchers, Tech. rep., HP Laboratories (2004).

[14] R. Everson, J. Fieldsend, Multi-class ROC analysis from a multi-objective optimisation perspective, Pattern Recognition Letters (2006) 918–927.

[15] C. Bourke, K. Deng, S.D.Scott, R. Schapire, N. Vinodchandran, On reoptimizing multi-class classifiers, Machine Learning 71 (2008) 219–242.

[16] D. Hand, R. Till, A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems, Machine Learning 45 (2) (2001) 171–186.

[17] C. Ferri, J. Hernández-orallo, M. Salido, Volume Under the ROC Surface for Multi-class Problems. Exact Computation and Evaluation of Approximations, in: Proc. of 14th European Conference on Machine Learning, 2003, pp. 108–120.

[18] T. Landgrebe, R. P. W. Duin, A simplified extension of the Area under the ROC to the multiclass domain, in: 17th annual Symposium of the Pattern Recognition Association of South Africa, 2006, pp. 241–245.

[19] N. Lachiche, P. Flach, Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves, in: Proc. 20th International Conference on Machine Learning (ICML'03), 2003, pp. 416–423.

[20] T. Landgrebe, R. Duin, Approximating the multiclass ROC by pairwise analysis, Pattern Recognition Letters 28 (13) (2007) 1747–1758.

[21] J. Fieldsend, R. Everson, Multiobjective Supervised Learning, in: Multiobjective Problem Solving from Nature, Natural Computing Series, Springer, 2008, pp. 155–176.

[22] M. Kupinski, M. Anastasio, Optimization and FROC analysis of rule-based detection schemes using a multiobjective approach, IEEE Trans. Med. Imaging 18 (8) (1999) 675–685.

[23] J. Horn, N. Nafpliotis, D. Goldberg, A niched pareto genetic algorithm for multiobjective optimization, in: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, 1994, pp. 82–87 vol.1.

[24] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast elitist multi-objective genetic algorithm: Nsga-ii, IEEE Transactions on Evolutionary Computation 6 (2002) 182–197.

[25] W. Khreich, E. Granger, A. Miri, R. Sabourin, Iterative boolean combination of classifiers in the roc space: An application to anomaly detection with hmms, Pattern Recognition Accepted - September 5, 2009 (2009) 36 pages.

[26] P. V. Radtke, E. Granger, R. Sabourin, D. O. Gorodnichy, Skew-sensitive boolean combination for adaptive ensembles an application to face recognition in video surveillance, Information Fusion 20 (0) (2014) 31 – 48.

[27] R. P. W. Duin, P. Juszczak, D. de Ridder, P. Paclik, E. Pekalska, D. M. Tax, PRTools, a Matlab toolbox for pattern recognition (2004).
URL `http://www.prtools.org`

[28] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, 2001.

[29] J. Bergstra, Y. Bengio, Random Search for Hyper-Parameter Optimization., Journal of Machine Learning Research 13 (2012) 281–305.

[30] Perclass 4.3 : Matlab toolbox for pattern recognition (2014).
URL http://perclass.com/index.php/html/

[31] K. Bache, M. Lichman, UCI Machine Learning Repository (2013).
URL http://archive.ics.uci.edu/ml

[32] S. Bernard, S. Adam, L. Heutte, Using random forests for handwritten digit recognition, in: Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on, Vol. 2, 2007, pp. 1043–1047.

[33] T.-F. Wu, C.-J. Lin, R. C. Weng, Probability estimates for multi-class classification by pairwise coupling, Journal of Machine Learning Research 5 (2004) 975–1005.

[34] R. A. McDonald, The mean subjective utility score, a novel metric for cost-sensitive classifier evaluation, Pattern Recognition Letters 27 (13) (2006) 1472 – 1477.

[35] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.