ROC-based cost-sensitive classification with a reject option

Clément Dubos, Simon Bernard, Sébastien Adam, Robert Sabourin

ROC-based cost-sensitive classification with a reject option

Clément Dubos*†,Simon Bernard*,Sébastien Adam*,Robert Sabourin†
*Université de Rouen, LITIS (EA 4108), BP 12 - 76801 Saint-Étienne du Rouvray, France
clement.dubos@etu.univ-rouen.fr, simon.bernard@univ-rouen.fr, sebastien.adam@univ-rouen.fr
†Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle,
École de Technologie Supérieure, Université du Québec, Montreal, Canada
robert.sabourin@etsmtl.ca

***Abstract*—In many real-world classification tasks, such as medical diagnosis, it is crucial to take into account misclassification costs for designing an accurate classification system. Nevertheless, begin able to reject a sample is also often needed in order to avoid a very risky prediction error. In that case, a cost-sensitive classifier must embed a rejection mechanism, that takes into account the rejection costs as well as the misclassification costs. In binary classification, the ROC space has shown to be very powerful for designing cost-sensitive classifiers, but it has been poorly exploited for designing classifiers able to reject. The purpose of this work is to extend a ROC-based ensemble method recently proposed, called the ROC Front method, with a cost-sensitive rejection mechanism. This approach compares favorably to the state-of-the-art ROC-based rejection rule recently proposed for binary cost-sensitive classification. It is also more robust as it allows to design an accurate classifier for all cost-sensitive situations contrary to the state-of-the-art method that fails in many cases, as for example with small datasets.**

I. Introduction

Many real-world classification problems naturally exhibit imbalanced misclassification costs. Medical diagnosis is a typical example for which predicting a healthy condition for a patient who actually suffers from a serious pathology is obviously more dangerous than the opposite situation. For such cases, a wide variety of classifiers exist that take into account some predefined costs, associated to each of the possible classification errors. However, even by focusing on specific types of error, at the expense of the others, it may be difficult to completely avoid very costly prediction errors. In the previous example of medical diagnosis, one single diagnosis error can imply very serious consequences. In such a situation, it is desirable to be able not to predict any of the healthy/pathological classes, instead of taking the risk to predict 'healthy' instead of 'pathological'. Consequently, in addition to cost-sensitivity, classifiers must be able to reject a sample when the risk of being wrong is critical.

In the cost-sensitive binary classification framework, the *Receiver Operating Characteristic* (ROC) space has shown to be very powerful for dealing with imbalanced misclassification costs ([1]). The reason is that it allows to describe the performance of binary classifiers at different operating points, i.e. considering different cost ratios. However, it has been poorly exploited for incorporating a reject option in the cost-sensitive framework. The main reason is that the ROC space does not naturally allows to consider the reject option as a third possible outcome of binary classifiers.

Consider a binary classification task, where a classifier predicts either the *positive* (P) or the *negative* (N) class. In a cost-sensitive framework, the misclassification costs are defined in a matrix, as shown in Table Ia. When adding the ability to reject, a third column has to be defined for the cost of rejecting samples from each of the two classes, as shown in Table Ib. The traditional 2D ROC space allows the representation of a classifier ability to handle C_{FP} and C_{FN}. Nevertheless, if one wants to transpose the ROC space in the Table Ib case, he will have to add 2 more dimensions, corresponding to C_{RP} and C_{RN}.

TABLE I: Cost matrices definition. C_{FN} denotes the cost of predicting *negative* instead of *positive*, C_{FP} the cost of predicting *positive* instead of *negative*, C_{TP} and C_{TN} are the profits associated to correct predictions, and C_{RP} (resp. C_{RN}) denotes the cost of rejecting a *positive* (resp. *negative*) sample.

(a) Traditional Cost Matrix

	$\hat{P}$	$\hat{N}$
P	C_{TP}	C_{FN}
N	C_{FP}	C_{TN}

(b) Cost Matrix with a reject option

	$\hat{P}$	$\hat{N}$	$\hat{R}$
P	C_{TP}	C_{FN}	C_{RP}
N	C_{FP}	C_{TN}	C_{RN}

As far as we know, very few works have proposed a reject-based extension of the ROC space ([2], [3]). In both [2] and [3], a third dimension, related to the rejection performance, is added to the regular 2D ROC space. In the first work ([2]), this third dimension represents the ability of an operating point to reject a sample predicted with a low confidence. However, it does not allow to take into account cost matrices like the one in Table Ib, since two more dimensions are needed to separately consider C_{RP} and C_{RN}, instead of just one.

In the second work ([3]), the third dimension represents the rates of samples predicted as *positive* whereas they are expected to be rejected. This extended ROC space is used for outlier detection, where samples are classified either as *positive* or *negative* whereas they may actually belong to a third unseen class. This case would correspond to adding a new row in the Table Ia, instead of a new column as it is the case in our context.

Tortorella proposes in [4] another type of ROC-based rejection mechanism. The traditional 2D ROC space is used to design a reject rule that leans on two decision thresholds applied on the binary classifier outcomes. It has been shown in [5] that this method is theoretically equivalent to the optimal Chow's reject rule proposed in [6], for binary classification. However this method is ineffective in two types of situation: (i) when the Equation 8 is not true (cf. Section II) and (ii) when the dataset used for the ROC analysis is too small to correctly estimate the ROC curve. In that case, the two decision thresholds are most of the times equals and irrelevant.

In a recent paper ([7]), we have shown that, for cost-sensitive classification, it can be more efficient to exploit the ROC space for learning a pool of classifiers, instead of only focusing on the decision thresholds proposed in the ROC curve of a single classifier. The purpose of the present work is to extend the method in [7] with a rejection mechanism that is able to jointly take into account the misclassification and rejection costs as defined in the Table Ib. The proposed method shows a significant improvement over the Tortorella's method, and allows to give an accurate cost-sensitive solution in all cases, in particular when the dataset is small.

The rest of the paper is organized as follows: the next section explains the Tortorella's method for designing a reject rule thanks to a ROC analysis; Section III presents the proposed approach; and Section IV details the experimental comparison between both methods, i.e. Tortorella's reject rule and our approach.

II. ROC-based reject rule (Tortorella's method)

A natural way to make any binary classifier cost-sensitive is to optimize a decision threshold on its outcome, according to a given cost matrix. This allows to train the classifier regardless the costs, and to find the decision threshold that best suits to the misclassification costs afterwards. To do so, the ROC space is often used, since it allows to represent any binary classification system 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). Such a classification system, that gives a prediction $\hat{y}$ for any instance $\mathbf{x}$, will be represented in the ROC space by a 2D point, as illustrated in Figure 1.

Consider now that a classifier gives an output score $h(\mathbf{x})$ on any instance $\mathbf{x}$. For having a prediction $\hat{y}$, a threshold t has to be set on $h(\mathbf{x})$, such that $\hat{y} = P$ if $h(\mathbf{x}) > t$, $\hat{y} = N$ else. A typical way to represent the performance of such a classifier in the ROC space is to consider all the possible thresholds, and to draw the corresponding ROC curve, as illustrated on Figure 2. In that way, it is easy to find the operating point of this curve (and thus the corresponding threshold t) that best suits to given misclassification costs. This is usually achieved using iso-performance lines ([8]) defined by a slope given by

$$m = \frac{p(N)C_{FP}}{p(P)C_{FN}} \tag{1}$$

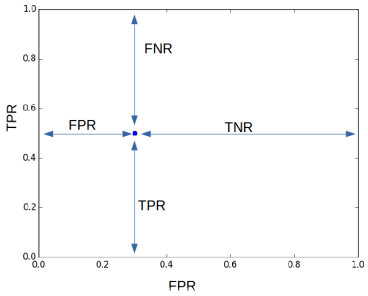

Fig. 1: Representation of a classifier system in the ROC space. FPR, TPR, FNR and TNR respectively denotes the false positive, true positive, false negative and true negatives rates.

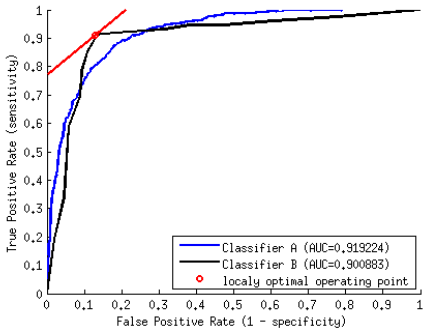

Fig. 2: ROC curves and iso-performance lines

where, $p(P)$ and $p(N)$ are the prior probabilities of the *positive* and the *negative* classes. 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 2.

The idea behind the method proposed in [4], denoted RBR hereafter, is to extend the iso-performance line principle for finding two decision thresholds, t_1 and t_2, in such a way that the final prediction $\hat{y}$ is defined by:

$$\hat{y} = \begin{cases} P & if\ h(\mathbf{x}) > t_1 \\ R & if\ t_2 < h(\mathbf{x}) < t_1 \\ N & else \end{cases} \tag{2}$$

These thresholds t_1 and t_2 can be find using two iso-performance lines, the slopes of which are:

$$m_1 = -\frac{p(N)C'_{TN}}{p(P)C'_{FN}} \quad and \quad m_2 = -\frac{p(N)C'_{FP}}{p(P)C'_{TP}} \tag{3}$$

where C'_{TN}, C'_{FN}, C'_{FP} and C'_{TP} are modified costs, as:

$$C'_{TN} = C_{TN} - C_{RN} \tag{4}$$
$$C'_{FP} = C_{FP} - C_{RN} \tag{5}$$
$$C'_{FN} = C_{FN} - C_{RP} \tag{6}$$
$$C'_{TP} = C_{TP} - C_{RP} \tag{7}$$

The threshold t_1 (resp. t_2) is found with the m_1 (resp. m_2) iso-performance line.

Note that there are cases for which it is not possible to find two consistent thresholds t_1 and t_2. When determining both values, three different situations can be encountered:

1) $t_1 < t_2$
2) $t_1 = t_2$
3) $t_1 > t_2$

In the second and third situations, the prediction function of Equation 2 can not be applied. In that case, Tortorella suggests in [4] not to perform any rejection, and to use instead the regular method with only one threshold. Two types of situations can led to these situations: (i) when Equation 8 is not true and (ii) when the dataset is too small. For further explanations, please see [4].

$$\frac{C'_{TN}}{C'_{FN}} > \frac{C'_{FP}}{C'_{TP}} \tag{8}$$

III. Rejecting with ROC-based cost-sensitive classifiers

In [9], [7], we have proposed an ensemble method for cost-sensitive classification, called the ROC Front method. The rationale behind this method is to replace the traditional decision parameter optimization by a model selection approach, based on an ensemble of cost-sensitive classifiers optimized in the ROC space. This approach has shown to significantly improve the cost-sensitive results over the decision parameter optimization approach. The goal of the method proposed in this paper is to extend this ROC-based model selection with a rejection mechanism.

What is called ROC Front is an ensemble of diverse cost-sensitive classifiers, that proposes a variety of solutions that suit to different cost-sensitive scenarios. When a cost matrix is considered, the most suitable classifier is selected accordingly in this ensemble. For adding a reject option, a second stage is added, composed with the following steps:

1) All pairs of classifiers from the ROC front are generated. Each pair of classifiers $H_{ij} = (h_i, h_j)$ is used as a combining classifier such that the prediction is obtained following the decision function:

$$\hat{y} = \begin{cases} P & if\ h_i(\mathbf{x}) = P\ and\ h_j(\mathbf{x}) = P \\ N & if\ h_i(\mathbf{x}) = N\ and\ h_j(\mathbf{x}) = N \\ R & else \end{cases} \tag{9}$$

2) These H_{ij} classifiers are projected in a 4D ROC space, where the dimensions correspond to the TPR, FPR, RPR and RNR, where RPR and RNR stand for the rejected positive rates and rejected negative rates. In that extended ROC Space, the point P_o of coordinates $(1, 0, 0, 0)$ corresponds to a classifier that always predicts the correct class for any new instance, and that never reject. Any other points of this space is a particular cost-sensitive classifier, represented by its 4 rates (TPR, FPR, RPR, RNR). The closer to P_o, the more accurate this classifier will be. However, as for the ROC Front method, the goal here is to obtain a diverse pool of classifiers, well spread all across the 4D ROC space.

Dataset	Instances	Positive	Negative	features
Pima	768	268	500	8
Heart	270	120	150	13
Australian	690	383	307	14
Diabetes	768	268	500	8
a1a	1605	395	1210	123
a2a	2265	572	1693	123
Sonar	208	97	111	60
Splice	1000	517	483	60
Seismic-bumps	2584	2414	170	19
Spectf	349	254	95	44
Spambase	4601	1813	2788	57
German	1000	300	700	24

TABLE II: Datasets

3) Among all the H_{ij} classifiers, a new 4D ROC Front is built with the non-dominated solutions, in the sense of Pareto-domination as defined in the multiobjective optimization literature (please see [10] for a formal definition of Pareto-domination). In practice, this means that the H_{ij} classifiers that can never be considered to be the best solution for at least one cost-sensitive scenario, are discarded from the 4D ROC Front.

This ROC Front variant, with reject option, is denoted RFR in the following.

Finally, when a given cost matrix is considered, the model selection process is applied in the same way it is done in [9], [7], that is to say by minimizing the loss function defined by:

$$\mathcal{L}(H_{ij}, D) = p(P)\left(\sum_{j=1..3} C_{1j}R_{1j}\right) + p(N)\left(\sum_{j=1..3} C_{2j}R_{2j}\right) \tag{10}$$

where D is a dataset, C_{1j} (resp. C_{2j}) corresponds to the costs in the first (resp. second) row of the Table Ib matrix, and R_{1j} (resp. R_{2j}) are the corresponding rates, estimated on D. For example, C_{11} is C_{TP} in Table Ib, and R_{11} is the true positive rate of H_{ij}, estimated on D.

IV. Experiments

A. Experimental protocol

For evaluating both the RBR and RFR methods, SVM classifiers with a Radial Basis Function (RBF) Kernel have been used as base classifiers. The reason of this choice is that these classifiers are naturally cost-sensitive through their 3 hyperparameters: γ, C^+ and C^- (please see [7] for additional explanations on these hyperparameters). By tuning these values, in particular C^+ and C^-, one can make such an SVM classifier suits to given misclassification costs. As a consequence, the ROC front is made up with an ensemble of SVM classifiers, all trained with different triplets of hyperparameters (γ, C^+, C^-). In the whole experiments, all the SVM has been implemented via the LibSVM software ([11]).

As for the experimental protocol, it has been inspired by the protocol used in [4]. First, 12 datasets have been selected from the UCI repository ([12]), and are described in Table II. For both methods, an independent validation set is required for finding the t_1 and t_2 in RBR, and for evaluating and selecting

	CTP/N	CFN	CRP	CFP	CRN
CM1	U(-10,0)	U(0,50)	1	U(0,50)	1
CM2	U(-10,0)	U(0,50)	1	U(0,100)	1
CM3	U(-10,0)	U(0,100)	1	U(0,50)	1
CM4	U(-10,0)	U(0,50)	U(0,30)	U(0,50)	U(0,30)
CM5	U(-10,0)	U(50,100)	U(0,50)	U(50,100)	U(0,50)
CM6	U(-10,0)	U(50,100)	U(0,50)	U(25,75)	U(0,50)
CM7	U(-10,0)	U(25,75)	U(0,50)	U(50,100)	U(0,50)
CM8	U(-10,0)	U(25,75)	U(0,50)	U(25,75)	U(0,50)

TABLE III: Cost Models. U denotes a uniform random selection in the corresponding interval of values. CM1 to CM4 are excerpt from [4]. In these first 4 CM, C_{RP} is always equal to C_{RN}, while in CM5 to CM8, different values of C_{RP} and C_{RN} can be obtained.

the pairs of classifiers in RFR. For that reason, each datasets have been divided into 3 distinct subsets, with 60% of the instances in the training set D_r, 20% in the validation set D_v, and the remaining 20% in the test set D_s. For reliability concerns, 10 random partitions of each dataset have been considered in the whole experiments.

Second, several cost matrices have been generated. For that purpose, eight types of cost matrices, called cost models (CM), have been considered. Each of these CM represents a typical imbalanced costs scenario. The first four CM are excerpt from the experimental protocol proposed in [4]. The last four CM are additional CM we propose to extend the analysis: none of the CM proposed in [4] correspond to situations where the reject costs are potentially imbalanced. However, it is an important feature in our context, since it may be important to focus on the rejection of one class of instances, rather than the other. Thus, it is desirable not to reject too much instances from the healthy class. As a consequence, the four additional CM propose situations where C_{RP} and C_{RN} are potentially imbalanced. In addition to this, these CM present a wider variety of imbalanced costs situations, with (i) both misclassification costs always higher than reject costs (CM5), (ii) one misclassification always higher than reject costs (CM6 and CM7) and (iii) potentially balanced misclassification costs (CM8). Table III summarizes the definition of these 8 cost models.

Each of these CM have been used to generate 1000 random cost matrices, leading to a total of $8 \times 1000 = 8000$ cost matrices. For each of them, results are averaged over the 10 random partitions described above. Consequently, both methods have been evaluated over $10 \times 8000 = 80000$ runs for each dataset.

Concerning the RBR method, a single SVM classifier has been trained on D_r, for which the hyperparameters have been chosen according to a classical grid search procedure, as proposed in the LibSVM ([11]). Then, the thresholds t_1 and t_2 have been selected with the validation set D_v. And finally, the resulting classifier has been evaluated on D_s, by computing the cost-sensitive loss defined in Equation 10.

As for the RFR method, the ROC Front has been built on D_r, as explained in [9], [7]. The pool of classifier combinations, made up with the pairs of classifiers from this ROC front, is then evaluated on D_v, through a 5-fold cross-validation (CV) procedure. As a result, each combination is represented in the extended 4D ROC space by a (TPR, FPR, RPR, RNR) point, each of these coordinates being averaged over the 5 folds of the CV procedure. Details are given by the Algorithm 1.

This algorithm takes as input a ROC Front, represented by an ensemble of hyperparameter vectors (γ, C_+, C_-), each of which corresponding to one particular SVM classifier. Firstly, the 5-fold cross-validation is prepared, by recording the predictions of all the classifiers from the ROC Front, on all the 5 folds (line 1 to 7). Secondly, all the possible pairs of classifiers from the ROC Front are generated (line 8). Thirdly, for all these pairs of classifiers, new predictions are given for each folds of the CV procedure, applying the decision rule given in Equation 9 (line 9 to 23). These predictions are used to build a mean confusion matrix, averaged over the rates obtained on the 5 folds. The projection of any combining classifier of the RFR method in the extended 4D ROC space, is excerpt from this mean confusion matrix. Finally, the algorithm outputs the "best" combining classifiers in this ROC space, in the sense of Pareto-domination, as explain in Section III (line 24).

When the cost matrix is then considered, the most suitable pairs is selected by minimizing the loss function of Equation 10, also estimated on D_v. Finally, the resulting combining classifier has been evaluated on D_s, as for the RBR method.

B. Results

In order to assess the significance of the results, the Wilcoxon signed ranks test has been applied on each cost matrix evaluation, as in [4] and as recommended in [13] for comparison of two classifiers. Consequently, each CM, applied on each dataset, have led to 1000 significant win/loss results, obtained over the 10 random partitions. These results are gathered in Table IV. Each cell of Table IV contains 3 values: from the top to the bottom, the significant win counts for the RFR method, the significant loss counts and the number of cost matrices for which both methods do not give loss results significantly different. For exhaustiveness, we also report in the last column, the total counts of win/loss/tie counts, over the 8000 matrices.

The first observation made from this table is that RFR and RBR do not give statistically different results in a majority of cases, i.e. 80.9% (71216 of the 88000 comparisons). Then, RFR significantly outperforms RBR for 17.78% of the cases (15642 over 88000), while RBR rarely outperforms RFR, i.e. for 1.3% of the cases only (1142 over 88000). To sum it up, the proposed approach is at least as accurate as the Tortorella's approach, and gives a significant improvement for more than 1 out of 6 cases, and so for a wide variety of imbalanced costs.

A strong advantage of the RFR method is that it never fails to give a suitable cost-sensitive solutions, whatever the priors and the costs. On the contrary, as explained at the end of Section II, it is sometimes impossible for the RBR approach

Files	CM1	CM2	CM3	CM4	CM5	CM6	CM7	CM8	Total
Heart	36	26	26	333	0	0	227	260	908
	11	8	68	1	0	1	0	0	89
	953	966	906	666	1000	999	773	740	7003
Australian	84	51	53	418	1	26	253	262	1148
	0	0	0	0	0	0	0	0	0
	916	949	947	582	999	974	747	738	6852
Diabetes	98	53	63	473	0	104	255	262	1308
	4	5	6	1	6	1	2	1	26
	898	942	931	526	994	895	743	737	6666
German	86	61	48	399	0	6	258	263	1121
	1	0	2	0	5	0	0	0	8
	913	939	950	601	995	994	742	737	6871
Pima	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	1000	1000	1000	1000	1000	1000	1000	1000	8000
Sonar	31	20	28	327	0	1	221	253	881
	6	11	5	4	7	0	12	14	59
	963	969	967	669	993	999	767	733	7060
Spectf	394	184	558	867	628	850	575	930	4986
	0	0	0	0	0	0	0	0	0
	606	816	442	133	372	150	425	70	3014
Spambase	126	130	81	402	7	7	263	263	1279
	0	0	1	0	0	0	0	0	1
	874	870	918	598	993	993	737	737	6720
Splice	65	51	45	361	0	0	254	261	1037
	209	262	212	36	68	72	23	11	893
	726	687	743	603	932	928	723	728	6070
A1a	120	88	68	549	8	206	308	280	1627
	0	0	0	0	0	0	0	0	0
	880	912	932	451	992	794	692	720	6373
A2a	112	67	63	476	0	88	279	262	1347
	4	5	2	5	10	6	1	33	66
	884	928	935	519	990	906	720	705	6587

TABLE IV: Results of the comparison as win/loss/tie counts over the $8 \times 11 = 88000$ cost matrices

Files	CM1	CM2	CM3	CM4	CM5	CM6	CM7	CM8
Heart	22	16	22	25	0	0	0	0
Australian	58	23	38	108	0	25	0	0
Diabetes	75	30	52	163	0	104	1	0
German	52	22	31	86	0	5	5	1
Pima	0	0	0	0	0	0	0	0
Sonar	15	10	23	22	5	1	13	13
Spectf	240	114	155	541	577	776	323	653
Spambase	63	41	37	84	0	7	0	0
Splice	50	32	35	55	3	11	0	0
A1a	96	52	56	238	8	206	54	18
A2a	89	45	53	171	8	94	26	33

TABLE V: Counts of cost matrices for which a 0-reject classifier has been used in replacement of the RBR method

to find consistent thresholds such that $t_1 < t_2$. Every time such a situation has been encountered in our experiments, a 0-reject classifier has been used instead, with only one decision threshold t used, and found by minimizing the loss function. The number of cases for which the RFR method is compared to the 0-reject classifier in replacement of the RBR method are summarized in Table V. These counts confirm the conclusion of Tortorella in [4]: it is more likely with the cost model CM4 than with the cost models CM1 to CM3, that the condition for finding the thresholds t_1 and t_2, expressed by Equation 8, are not met.

Another interesting observation made from this table is that RBR often fails to find a consistent pair of threshold (t_1, t_2) for the *Spectf* dataset. From our point of view, it highlights another drawback of the method: when there is only few validation instances available, which is the case of the *Spectf* dataset due to the low number of negative instances, the ROC curve does not contain a sufficient number of operating points to propose enough different cost-sensitive solutions. This is well illustrated in Figure 3, which represents the ROC curve of the classifier used in the RBR method for the *Spectf* dataset (the blue points). The potentially selected operating points for any cost-sensitive scenarios are the points lying on the red curve, called the convex hull. One can see that only 4 points are available here. In that case, it is likely that the same operating point will be selected for both thresholds. This pathological case is also discussed in [4]. On the opposite, the corresponding ROC Front used in RFR, made up with several cost-sensitive classifiers (the red points), proposes a lot more diverse ensemble of solutions. The reason is that the number of

Algorithm 1 ROC Front with reject option (RFR)

Require: D_v : a Dataset
Require: F : the number of folds for the cross-validation (CV) procedure
Require: $\Theta = \{\theta_i, i = 1..K\}$: a pool of K triplets of hyperparameters, composing the ROC Front, where $\theta_i = (C_i^+, C_i^-, \gamma_i)$
Data $\Omega = \{(\theta_i, \theta_j), i, j = 1..K, i \geq j\}$: a pool of $K(K+1)/2$ pairs of classifiers from Θ
Data T_n : the n^{th} fold of CV procedure applied on D_v, with $n = 1..F$
Data $P_{n,i,j}$: the prediction of the i^{th} classifier, on the j^{th} instance of the n^{th} fold
Data $Conf_{n,i}$: the confusion matrix of the i^{th} classifier, obtained on the n^{th} fold
Function $PartitionForCV(D_v, F)$: outputs F distincts folds following a classical CV procedure.
Function $Train(\theta, T)$: outputs a classifier trained on the dataset T, with the θ hyperparameter values.
Function $Predict(\theta, T)$: outputs the prediction of a given SVM classifier θ for all the instances in T
Function $bestCombiningClassif(Conf, \Omega)$: outputs the subset of Ω that represent the best solutions in the multiobjective optimization sense (Pareto optimality).
Ensure: $\Gamma = \{(\theta_i, \theta_j), \theta_i, \theta_j \in \Omega\}$: the ensemble of pairs of SVM classifiers, composing the RFR ensemble.

$T_{n=1..F} \leftarrow PartitionForCV(D_v, F)$
for i from 1 to K **do**
  for n from 1 to F **do**
    $\vartheta \leftarrow Train(\theta_i, \cup_{s=1,s\neq n}^{F} T_s)$
    $P_{n,i,.} \leftarrow Predict(\vartheta, T_n)$
  end for
end for
$\Omega \leftarrow \{(\theta_i, \theta_j), \theta_i, \theta_j \in \Theta, i, j = 1..K, i \geq j\}$
for i=1..K **do**
  for j=i..K **do**
    for n from 1 to F **do**
      for p from 1 to $|T_n|$ **do**
        if $P_{n,i,p} \neq P_{n,j,p}$ **then**
          $Preject_p \leftarrow Rejection$
        else
          $Preject_p \leftarrow P_{n,j,p}$
        end if
      end for
      $Conf_{n,.} \leftarrow ConfusionMatrix(T_n, Preject)$
    end for
    $\overline{Conf_t} \leftarrow mean(Conf_{n,.})$
  end for
end for
$\Gamma \leftarrow bestCombiningClassif(\overline{Conf.}, \Omega)$

these classifiers does not depend on the number of validation instances available. This explains the particularly good results of RFR on the *Spectf* dataset, as shown in Table IV.

V. Conclusion

In this paper, a ROC-based ensemble method has been proposed for cost-sensitive classification with rejection. This method relies on a pool of cost-sensitive classifiers, called the ROC Front. Pairs of classifiers from the ROC Front are combined to add the ability to reject. The combining classifiers that proposes the best solution for tackling particular cost-sensitive situations, i.e. with given (imbalanced) misclassification and rejection costs, are retained to compose a new ROC Front with reject option.

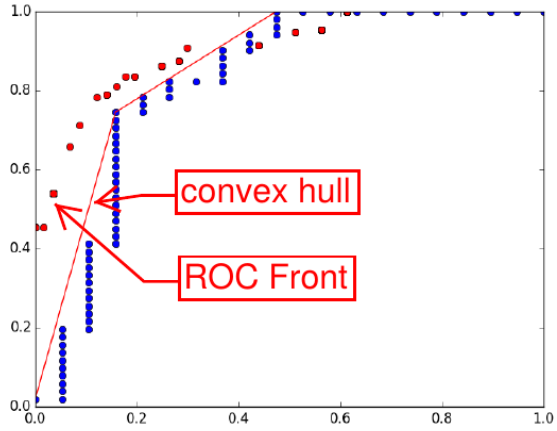

Fig. 3: Illustration of the pathological case for RBR, where the ROC curve does not contain enough operating points for selecting two differents thresholds

The proposed approach compares favorably to the state-of-the-art ROC-based reject rule, with two noticeable improvements: (i) it gives a significant improvement in terms of cost-sensitive performance and (ii) it is always able to propose an efficient solution for any cost-sensitive situation, contrary to the state-of-the-art method that regularly fails to introduce the reject option regarding the costs ratios.

References

[1] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.
[2] S. Alsing, E. Blasch, and K. Bauer, "Three-dimensional receiver operating characteristic (roc) trajectory concepts for the evaluation of target recognition algorithms faced with the unknown target detection problem," pp. 449–458, 1999.
[3] T. Landgrebe, D. Tax, P. Paclk, and R. Duin, "The interaction between classification and reject performance for distance-based reject-option classifiers," *Pattern Recognition Letters*, vol. 27, pp. 908–917, 2006.
[4] F. Tortorella, "A roc-based reject rule for dichotomizers," *Pattern Recognition Letters*, vol. 26, no. 2, pp. 167–180, 2005.
[5] C. Santos-Pereira and A. Pires, "On optimal reject rules and roc curves," *Pattern Recognition Letters*, vol. 26, no. 7, pp. 943–952, 2005.
[6] C. Chow, "On optimum recognition error and reject tradeoff," *IEEE Transactions on Information Theory*, vol. 16, no. 1, pp. 41–46, 1970.
[7] S. Bernard, C. Chatelain, S. Adam, and R. Sabourin, "The multiclass roc front method for cost-sensitive classification," *Pattern Recognition*, vol. 46, pp. 46–60, 2016.
[8] F. Provost and T. Fawcett, "Robust classification for imprecise environments," *Machine Learning*, vol. 42, no. 3, pp. 203–231, 2001.
[9] C. Chatelain, S. Adam, Y. Lecourtier, L. Heutte, and T. Paquet, "A multi-model selection framework for unknown and/or evolutive misclassification cost problems," *Pattern Recognition*, vol. 43, no. 3, pp. 815–823, 2010.
[10] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.
[11] C. Chang and C. Lin, "Libsvm: A library for support vector machines," *ACM Transaction on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.
[12] K. Bache and M. Lichman, "UCI Machine Learning Repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[13] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.