Comparing Multi-objective and Threshold-moving ROC Curve Generation for a Prototype-based Classifier

Ricardo Aler
Computer Science Department
Universidad Carlos III de Madrid, Spain
aler@inf.uc3m.es
Julia Handl
Manchester Business School
University of Manchester, UK
julia.handl@mbs.ac.uk
Joshua D. Knowles
School of Computer Science
University of Manchester, UK
j.knowles@manchester.ac.uk

July 10, 2013

Abstract

Receiver Operating Characteristics (ROC) curves represent the performance of a classifier for all possible operating conditions, i.e., for all preferences regarding the tradeoff between false positives and false negatives. The generation of a ROC curve generally involves the training of a single classifier for a given set of operating conditions, with the subsequent use of threshold-moving to obtain a complete ROC curve. Recent work has shown that the generation of ROC curves may also be formulated as a multi-objective optimization problem in ROC space: the goals to be minimized are the false positive and false negative rates. This technique also produces a single ROC curve, but the curve may derive from operating points for a number of different classifiers. This paper aims to provide an empirical comparison of the performance of both of the above approaches, for the specific case of prototype-based classifiers. Results on synthetic and real domains shows a performance advantage for the multi-objective approach.

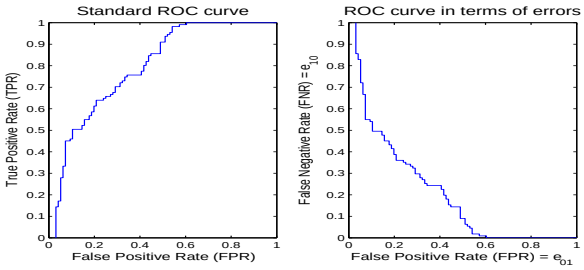

Figure 1: Standard ROC curve (left) and ROC curve in terms of errors FNR and FPR (right)

1 Introduction

Receiver Operating Characteristic (ROC) curves are an important tool in classification problems, especially in applications that involve dynamic or imprecise operating conditions [19], where the class distributions and the misclassification costs of future data may differ from those of the data available at the time of training. More concretely, a ROC curve is a graphical plot of the True Positive Rate (TPR) versus the False Positive Rate (FPR) of a binary classifier (although ROC analysis can be extended to multi-class problems, see [5]).

A ROC curve is typically constructed in two steps. First a classifier capable of ranking or scoring instances (e.g. a neural network) is trained for a particular set of operating conditions, using a standard learning algorithm. For instance, backpropagation optimizes neural network weights by minimizing training error, assuming equal missclassification costs and the class distribution present in the training data. The resulting classifier is then evaluated, which produces a single point in ROC space.

Second, the output threshold of the classifier is varied to obtain a range of different points (TPR, FPR) in ROC space [5]. The resulting ROC curve represents the classifier's performance for different operating conditions, which allows for the selection of an optimal operating point (threshold) for this classifier once the operating conditions become known. Figure 1 displays a standard ROC curve on the left. ROC curves can also be represented in terms of errors (FNR and FPR), as will be shown later in the paper and it is pictured in Figure 1 on the right.

Interestingly, points in ROC space obey the principle of Pareto dominance. This allows the formulation of ROC curve generation as a multi-objective (MO) optimization problem, where the objective space corresponds to the ROC space and the decision space corresponds to the parameters (e.g. neural network weights) and the thresholds of the classifiers employed. A complete formulation of this approach was presented by Everson and Fieldsend, who used PAES for evolving the parameters of neural network-based classifiers [4]. In contrast to the approach of threshold-moving (where a single classifier is obtained), the MO

approach may produce a number of different classifiers (e.g. a set of different weights for the neural networks used in [4]), each one optimized for different operating conditions. This set of classifiers maps to an approximation of the Pareto front (PF) in ROC space, which may be used in the same way as a ROC curve. In short, a threshold-moving ROC curve involves learning the parameters of a classifier and then moving the threshold, while a MO ROC curve basically involves learning a non-dominated collection of parameters.

Given that classifiers are optimized for individual operating conditions, one may expect the MO approach to provide better ROC curves than the curves produced using standard thresholding, in theory. However, there has been only limited empirical evaluation regarding the actual practical performance differences between the two approaches. This paper focuses on the specific case of prototype-based classifiers and carries out an empirical comparison across several data sets.

The structure of this paper is as follows. Section 2 defines key concepts about ROC curves, including the generation methods based on threshold-moving and MO optimization. Section 3 introduces prototype-based classifiers and describes the generation of ROC curves for these classifiers. It also sets out the methodology employed for the comparison of ROC curves. Section 4 presents and discusses the results of the experiments. Finally, Section 5 draws conclusions and discusses future research directions.

2 Background

The aim of this Section is to define key concepts about classifiers, ROC curves and the multi-objective and threshold-moving methods to generate them.

2.1 Basic terminology

We consider binary classification, and two-class ROC analysis. A binary classifier maps instances into two classes $\{C_0, C_1\}$ commonly referred to as the negative and the positive class. Discrete classifiers return a class prediction only, whereas rankers, scorers, and probability estimators output a numerical value that is linked to the degree of membership of instances to specific classes [7]. More concretely, rankers sort all instances from the negative to the positive class. Scorers go beyond this and provide a continuous output that measures the degree of membership of instances to one of the classes. Probability estimators are specific types of scorers that estimate class memberships using probabilities (e.g. $p(C_i|x)$, for every instance x). It is clear that both probability estimators and scorers can be easily transformed into rankers, which in turn can be transformed into discrete classifiers by setting a threshold. Most work in classification and ROC analysis deals with scorers and this will also be the focus of this manuscript.

In binary classification, classifiers are commonly evaluated by estimating the True Positive Rate (TPR) and the False Positive Rate (FPR). TPR is the

number of instances classified as positive divided by the number of instances that are actually positive, whereas FPR is the number of negative instances classified as positive, divided by the number of negative instances. More generally, the performance of a classifier can be measured by the set of probabilities e_{ij} that measure the probably of mistakenly assigning instances in class C_i to class C_j. This formulation is more homogeneous (it only uses errors), can be extended to multi-class problems, and, for binary classifiers, is equivalent to the above, as $TPR = 1 - e_{1,0}$ and $FPR = e_{0,1}$.

For two-class problems, ROC curves (ROCs) are two-dimensional graphs in which TPR is plotted on the Y axis and FPR rate is plotted on the X axis. If the errors e_{ij} are used instead, the result is an inverted (but equivalent) ROC curve, where $e_{1,0}$ is plotted against $e_{0,1}$. For $q > 2$ classes, ROC space has $q * (q - 1)$ coordinates (e_{ij} with $i \neq j$) [4].

2.2 Threshold-moving

A given (trained) discrete classifier can be represented in ROC space as a point $(e_{1,0}, e_{0,1})$. Furthermore, a given (trained) ranker, scorer, or probability estimator can be represented by a ROC curve, which is generated by moving the classifier's threshold t. If $s(x)$ is the score assigned to instance x by the scorer s, all instances with a score of $s(x) > t$ will be classified as positive, and all others as negative. For any choice of the threshold t, a point $(e_{1,0}, e_{0,1})$ of the ROC curve can be generated. For $q > 2$ classes, scorers usually output q scores. In this case, threshold-moving becomes impractical (NP-hard), because each threshold has to be paired with all possible thresholds for the rest of the classes [15]. Therefore, there is potential for heuristic optimization approaches [4].

2.3 Multi-objective approaches for ROC generation

Based on the observation that Pareto dominance holds in ROC space, an "optimal" ROC curve can be defined more generally as the result of the multi-objective optimization of $s_{w,t}$ where s_w is a scorer defined by the set of parameters or weights $w = \{w_1, w_2, \ldots\}$ and $s_{w,t}$ is the discrete classifier that results from applying a particular threshold t to the scorer s_w. For instance, $s_{w,t}$ could be a neural network with the weights w and the output threshold t. The decision space of this multi-objective optimization problem is (w, t) and the objective space is $(e_{0,1}, e_{1,0})$ (e_{ij} in general, with $i \neq j$). In this formulation, a multi-objective optimizer can be used to find an approximation to the real Pareto front, which can be interpreted as a ROC curve. However, in contrast to threshold-moving, the optimal operating points in this ROC curve may correspond to different scorers $s_{w,t}$ (i.e. classifiers that have been discretized at different values of the threshold t, but may also differ in terms of their actual parameters w). This multi-objective formulation was first suggested in [22], tested in [13] using a Niched-Pareto Genetic Algorithm, and generalized to multi-class problems in [4], using the algorithm PAES [12].

2.4 Differences between the two approaches

The multi-objective approach provides a more general formulation for the problem of generating ROC curves. A further motivation of the approach is the observation that it provides classifiers that are optimized for all possible operating conditions [13]. In contrast to this, threshold-moving trains a classifier for a single operating point only. The ROC curve derived from threshold-moving is likely to be optimal for this particular operating point, but possibly not optimal for very different operating conditions.

Despite these conceptual and theoretical considerations, there is currently only limited empirical evidence for an actual practical advantage of the multi-objective approach. Initial work on this topic [13] showed that, for neural networks, the ROC curves obtained using multi-objective optimization improved upon those obtained from standard threshold-moving. However, the study was limited to a single XOR synthetic domain and neural networks with two hidden units only. Everson and Fieldsend [4] focused on the extension of the multi-objective approach to multi-class problems. The resulting method was applied to a number of different domains, but, due to the focus of the work, an empirical comparison to threshold-moving was not provided. Finally, [8] used NSGA-II for the direct evolution of neural networks in ROC space and compared them to a single-objective approach based on threshold-moving. The results on three different 2-class UCI domains showed that anti-overfitting measures worked better for the multi-objective than for the threshold-moving approach. However, if no anti-overfitting measures were used, the ROC curves obtained from both approaches were quite similar. In principle, this work did not exclude the possibility that the threshold-moving approach could have been competitive, if other anti-overfitting measures had been used.

As mentioned in Section 2.2, standard threshold-moving becomes NP-hard for more than two classes. Therefore, two-class domains offer a good setting for comparison, because simple thresholding is still efficient, so it can be checked whether the additional conceptual advantages for the MO approach hold in this simple case.

2.5 Research question

Given the above, the principal aim of this manuscript can be formalized as follows. Let $s_w(x) : X \rightarrow \Re$ be a scorer whose parameters or weights are $w = \{w_1, w_2, \ldots\}$, that maps instances $x \in X$ to a continuous value, indicating the degree of membership to the positive class. Let $s_{w,t}(x) : X \rightarrow \{0, 1\}$ be the discrete classifier obtained from $s_w(x)$ by setting a threshold t, that classifies instances x as positive if $s_w(x) > t$ and negative otherwise.

The multi-objective approach generates a ROC curve by optimizing the two objectives $(e_{0,1}, e_{1,0})$ over the decision space (w, t). As will be explained in Section 3.3, a population-based multi-objective evolutionary algorithm will be used, and the ROC curve generated can potentially contain all members of the population. The threshold-moving approach optimizes some single-objective

(scalar) criterion (such as the mean squared error or the classification accuracy) over the decision space w and, subsequently, generates the ROC curve by moving the threshold t. In this paper, the threshold-moving approach will also be referred to as "single objective" (SO) approach, given that a scalar value is optimized.

The hypothesis to be tested in our experiments is whether the multi-objective approach generates ROC curves that are consistently better than those returned by simple thresholding. If this is not the case, the single objective would be preferable, in principle, due to its smaller computational expense.

3 Methodology

3.1 Prototype-based classifiers

Prototype-based classifiers generally show good classification performance and tend to generate very simple models [9]. They are intuitive since they represent their decisions in terms of the prototypes in feature space and have a direct geometrical representation, which is useful for the visualization of classification boundaries. Finally, the classification procedure used in prototype-based classifiers generalizes to more than two classes.

A prototype-based classifier consists of a set of p prototypes, where p indicates the complexity of the classifier. A prototype is represented by its coordinates in feature space, as well as its class. A prototype-based classifier classifies instances by computing the distance to each of the prototypes and returning the class of the closest prototype. To obtain a classifier that returns a score rather than a class, we can score instances x based on the difference between the distances of the two nearest different-class prototypes p_0 and p_1 [17]. This is done in the following way: let p_0 and p_1 be the closest prototypes to x belonging to class 0 and 1, respectively. The score of instance x is $s_w(x) = d(x, p_1) - d(x, p_0)$, where d is the Euclidean distance. Scores range from $-\infty$ (class 1) to ∞ (class 0). It can be seen threshold $t = 0$ corresponds to the actual prototype-based classifier. Scores can be normalized to [0,1] by means of the sigmoid function. In that case, the default threshold becomes $\frac{1}{2}$.

3.2 Objective function

In order for the comparison between the SO and MO approach to be meaningful, the scalar criterion or loss function to be optimized by the SO approach should guarantee that the optimal classifier is part of the Pareto optimal set of the multi-objective formulation. Otherwise, the MO approach might outperform the SO approach just because the SO criterion does not return solutions that are optimal in the Pareto sense. Certain classification training criteria, such as the mean-squared error, do not meet this condition. A scalar criterion of the form $S_{A,B} = A * e_{0,1} + B * e_{1,0}$ guarantees that s_w will be on the front, for any values of A and B. If A and B were set to the proportion of negative

and positive instances (respectively), then $S_{A,B}$ would provide the classification error (i.e. 1−accuracy). In this paper, the setting $A = B = 1$ has been chosen, so that the SO criterion does not depend on class proportions and gives equal importance to both classes. Also note that the Area under the Curve (AUC) of a discrete classifier with TPR and TNR (true positive and negative rates) is $\frac{(TPR+TNR)}{2}$[1]. Therefore, the minimization of $e_{0,1} + e_{1,0}$ is equivalent to the maximization of the AUC of the corresponding discrete classifier.

3.3 Optimization method

A further experimental choice are the algorithms used for the MO and SO optimization. For the MO case, MSOPS-II will be used [10]. MSOPS-II is one of the best algorithms for multi-objective optimization for many-objective problems. In this article, only two-class problems have been tested, but given that ROC space dimensionality grows as the square of the number of classes, future research is expected to require an optimizer capable of dealing with many objectives. For the SO optimization, Learning Vector Quantization (LVQ) is a well-known family of training algorithms for prototype-based systems [21]. However, LVQ does not guarantee that solutions are Pareto optimal in ROC space. In addition, if different algorithms are used for the MO and SO optimization, then differences between the two approaches may be due to differences in the effectiveness of the particular optimizers used. This problem can be avoided by using MSOPS-II for the implementation of both approaches. Specifically, the Pareto front (PF) approximation obtained from the MO approach is directly filtered to select the specific solution that optimizes the SO criterion $e_{0,1} + e_{1,0}$ (also referred to as the $fp+fn$ solution). In this way, the influence of optimizer performance and statistical variation are reduced.

3.4 Encoding

Prototype-based classifiers are directly encoded in the MS-OPS-II chromosome, by encoding their coordinates. The class of the prototypes is encoded indirectly: prototypes are distributed uniformly among the classes, with the first prototype assigned to class 0, the second to class 1, the third to class 0, and so on. All classifiers in the MSOPS-II population contain the same number of prototypes. This is similar to the approach in [13] and [4], where all neural networks in the population had the same number of hidden units. (Differently to this, the work reported in [8] used classifiers of varying complexity within the same population.)

In previous paragraphs we have assumed that the MO approach operates in the decision space (w, t) (optimizing both the classifier's parameters w and its threshold t). In the present study, a further simplication was made, as

[1]The ROC curve of a discrete classifier is made of three points: the discrete classifier itself (TPR, TNR) and the two trivial classifiers at (TPR=1, TNR=0) and (TPR=0, TNR=1). The AUC of this ROC curve is made of a triangle and a trapezoid, whose areas can be added to compute the AUC [6].

the decision space was restricted to the set of feature vectors w only. More specifically, the scorers were used as discrete classifiers with a fixed (default) threshold of $t = \frac{1}{2}$. This approach reduces the dimensionality of the decision space, by removing one degree of freedom from the scorer. The empirical results in Section 4 show that the optimization of feature vectors alone is sufficiently powerful to obtain good classification results in all of the domains tested. The reason for this is that changes in the location of prototypes can account for the lack of changes in the threshold. In other words, $s_{w,t}$ represents a boundary in feature space (i.e. a discrete classifier), which may also be approximated by a discrete classifier $s_{w',t'}$ with different set of parameters w', but a default threshold of $t' = \frac{1}{2}$.

4 Empirical comparison

4.1 Methodology

This section describes the methodology applied for comparing the ROC curves generated using the MO and the SO approach. We describe a metric based on cost-curves, as well as the cross-validation procedure used in the experiments.

4.1.1 Average expected cost

The ROC curves obtained by MSOPS-II and threshold-moving are learned on the training set, and do then need to be compared on independent test data. A popular scalar measure to evaluate ROC curves is the area under the curve (AUC) computed on the test set, which can also be interpreted as the probability that the score for positive instances is larger than the score for negative ones. The AUC provides a natural way of quantifying the ranking abilities of the scorer associated to the ROC, but it does not entirely reflect the way ROC curves are typically used. ROC curves can be used in classification when future operating conditions are not known at training time and the operating point must be selected once the target operating conditions have become known [19]. If there are few data points available, it is likely that there will be significant mismatches between the ROC curves obtained on the training and testing data (most likely as a result of overfitting on the training data). This mismatch can introduce additional errors because the operating point is selected based on the ROC curve on training data, but this may not accuractely reflect the best operating point from the perspective of the ROC curve on the testing data.

The following (simplistic) example is designed to illustrate this:

- Case 1: The ROC curve on the training data contains just two $(e_{0,1}, e_{1,0})$ points: (0.35, 0.25) and (0.25, 0.35). When tested on the test data, the corresponding classification performance for the corresponding operating points is (0.25, 0.35) and (0.35, 0.25), respectively. In other words, the training errors on these data fail to represent accurately the testing errors, a problem that commonly occurs with small data sets.

- Case 2: The ROC curve on the training data contains the two points (0.25, 0.35) and (0.35, 0.25), and the ROC curve obtained for the testing data is identical to this ROC curve on the training data. This situation would be expected to be the case for large data sets.

In both cases, the AUC obtained from testing will be identical because the ROC curves on the test data are identical. However, in the first case, the selection of an operating point based on the training ROC may be misleading, due to the mismatch between the ROC curves on the training and testing data. This problem is absent in the second case. Evaluation based on the testing AUC only does not reflect the difference between the two situations.

To avoid this limitation of the AUC, this paper adopts a different approach to the comparison of ROC curves , which is inspired by the average expected cost metric (TEC) suggested by Drummond and Holte [3]. TEC essentially repeats the operating point selection process for every possible operating condition, computes the corresponding cost, and averages across all conditions [3] [2]. Thus, TEC measures the expected cost (the average over all possible operating conditions) of the ROC curve on a given data set. Here, we propose to use the testing data for computing TEC, but carrying out the operating point selection on the ROC built from the training set, to ensure that the effect of mismatches between training and testing ROCs on operating point selection can be taken into account.

We refer to this measure as 'average expected cost' (AEC) and it is computed by Eq. 1, where E is the expectation operator, and $\mathbf{P} = (P_0, P_1)$ (class distribution) and $\mathbf{C} = (C_{10}, C_{01})$ (misclassification costs) represent the operating conditions. The expression $\hat{\mathbf{e}}_{(\mathbf{P},\mathbf{C})}$ represents the errors on the testing set of the optimal point $\mathbf{e}_{(\mathbf{P},\mathbf{C})}$ selected from the training ROC for operating conditions $(\mathbf{P}, \mathbf{C})$. It is assumed that all possible operating conditions are equally likely:

$$AEC = E_{(\mathbf{P},\mathbf{C})}[Cost(\hat{\mathbf{e}}_{(\mathbf{P},\mathbf{C})}, \mathbf{P}, \mathbf{C})] \tag{1}$$

where $Cost(\hat{\mathbf{e}}_{(\mathbf{P},\mathbf{C})}, \mathbf{P}, \mathbf{C})$ in Eq. 1 is computed in three steps: first select the minimum cost point $\mathbf{e}_{(\mathbf{P},\mathbf{C})}$ from the training ROC, second compute its errors $\hat{\mathbf{e}}_{(\mathbf{P},\mathbf{C})}$ on the test set, third compute its expected cost.

Under the operating conditions $\mathbf{P} = (P_0, P_1)$ and $\mathbf{C} = (C_{10}, C_{01})$, the expected cost of any point $\mathbf{e}$ (from the training ROC) can be computed as shown in Eq. 2 (assuming that correct classification has no costs):

$$Cost(\mathbf{e}, \mathbf{P}, \mathbf{C}) = e_{10}P_1C_{10} + e_{01}P_0C_{01} \qquad . \tag{2}$$

Operating point selection is carried out by selecting the point from the training ROC that minimizes the expected cost. In short, $\mathbf{e}_{(\mathbf{P},\mathbf{C})} = \arg\min_{e \in \mathrm{ROC}} Cost(\mathbf{e}, \mathbf{P}, \mathbf{C})$. Eq. 2 is then used to compute the testing cost, once the errors $\hat{\mathbf{e}}_{(\mathbf{P},\mathbf{C})}$ on the testing set have been evaluated.

[2] Details are slightly more complicated as this computation is done in terms of cost curves, as will be explained later.

A minor technical point about the testing cost is that in principle, $\mathbf{P}$ should describe the class distribution of the testing set. In practice, this is not important, since the operating conditions $(\mathbf{P}, \mathbf{C})$ are never used independently. Rather, the products P_1C_{10} and P_0C_{01} are employed. Given that missclassification costs can have any positive value, it does not really matter the actual values of $\mathbf{P}$.

It turns out that the average expected cost of Eq. 1 can be given a geometrical interpretation by means of cost curves [3] (just like AUC being equivalent to the probability of ranking instances correctly). Cost curves were introduced by Drummond and Holte in 2006 [3] as an alternative way of visualizing ROC curves, but focusing on expected cost. For two-class problems, cost curves display (normalized) expected cost on the y-axis for all possible (normalized) operating conditions on the x-axis. [3] shows that TEC is equivalent to the area under the cost curve. [3] should be consulted for the technical details. This provides an elegant way of computing Eq. 1 that has been used in the present paper: for every possible normalized operating condition, select the operating point on the training ROC and compute its normalized expected cost. Finally, determine the area under this curve.

A graphical representation of what is computed in Eq 1 can be seen in Figures 4 and 5, later in the paper. For instance, the top left figure of Figure 4 displays the ROC curves obtained by the SO and MO approach on the training data. These would be used to select operating points (i.e. classifiers from the MO ROC curve, or thresholds from the SO ROC curve) once the operating conditions become known. The top right figure of Figure 5 displays the cost curve computed by simulating each operating point selected on the training ROC curve, on the test data. The average expected cost of Eq. 1 is equivalent to the area under the cost curve in the top right figure of Figure 5. It must be stressed that the top right figure of Figure 5 is not the testing cost curve, but the curve obtained by selecting operating points on the training ROC and computing the cost on the testing data.

4.1.2 Cross-validation procedure and statistical significance

In order to obtain statistically significant conclusions, several training/testing partitions were employed for each domain, according to the following procedure:

- If the size of the dataset is large, overfitting should not pose a significant problem, and, therefore, the following training/testing procedure was followed: 40% of the data were used for training, and 60% were used for testing. This was repeated 5 times with different training/testing partitions. In order to take into account the stochastic nature of MSOPS-II, five independent runs of the optimizer were used for each partitions. Therefore, for these domains, MO was run 5x5=25 times, and the results are averages over these 25 runs. This approach was used for the two synthetic domains where sufficient data was available.

- In the real domains, a 5x2(x2) cross-validation procedure [2] was used

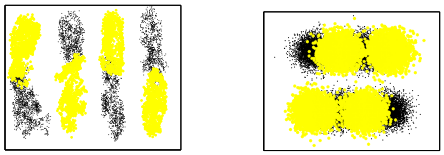

Figure 2: Irregular chess domain (left) and Gaussian chess domain (right).

using the optimally balanced stratified cross-validation [16]. For every training/testing partition, MSOPS-II was run twice (with different random seeds) and the results were averaged. In total, MSOPS-II was run 5x2x2=20 times, and the results displayed in the tables are averages over these 20 runs.

- Given that the multi-objective and fp+fn solutions are obtained from the same run, under exactly the same conditions, a paired t-test was used to test for statistical significance, with $\alpha = 0.05$, following the procedure described in [24] (chapter 5.3).

4.2 Domains and parameters

Two synthetic domains and four real domains (from the UCI database [1]) have been tested. All domains are 2-class problems with numerical attributes. The synthetic domains are: irregular chess and gaussian chess (see Figure 2). Irregular chess (8590 instances) is a two-attribute problem drawn by hand with the MLdemos tool [18] with 8 irregular overlapping clusters. Gaussian chess contains 8 symmetrical and identical gaussians. The four real domains are: Blood (748 instances, 5 attributes), Diabetes (768 instances, 8 attributes), German Number (1000 instances, 24 attributes), Estate (5322 instances, 12 attributes).

Distance-based classifiers are known to be very sensitive to irrelevant attributes. Therefore, prior to running MSOPS-II, a correlation-based filter method (CFS + best-first search [24]) was used to reduce the number of attributes.

4.3 Parameter settings

Table 1 displays the main MSOPS-II parameters used in the runs. In order to help MSOPS-II to converge more quickly, 25% of the individuals of the initial population were generated using a stochastic gradient descent algorithm called Generalized Learning Vector Quantization (GLVQ) [20], using the MATLAB implementation of [23]. GLVQ may converge to local optima [9], which was seen as an advantage in this context, since a set of diverse initial solutions was generated. The remaining 75% of the initial population were generated randomly. Both the MO and SO approach were tested across a range of classifier complexities (number of prototypes).

Parameter	Value
Population size	100
Iterations	1000
Tau	0.001
Aggregation method	weighted min-max
Number of target vectors	**50** (default)
Number of decision space targets	**20** (default)
Crossover rate	**0.5** (default)
Probability of mutation	**0.9** (default)

Table 1: MSOPS-II parameter settings

4.4 Results

Figure 3 displays the results for the synthetic domains: irregular chess (top) and gaussian chess (bottom). The y-axis shows how the average expected cost changes as the number of prototypes increases. The solid line represents the results obtained for the MO approach while the dashed one corresponds to the SO solution. A circle means that the difference between the MO and SO results is statistically significant. Despite the fact that these two synthetic data sets are structurally similar, the results obtained are markedly different. For the irregular chess domain, it can be seen that the MO approach outperforms the SO approach, for all classifier complexities. The differences are significant on both the training and testing data. It can be seen that after 8 prototypes, there is almost no improvement in the expected cost. This is reasonable because the domain contains 8 irregular clusters. Contrariwise, in the gaussian chess domain, for complexities equal or larger than 8, the average expected cost is comparable for the SO and MO approach.

The different behavior of the SO and MO approach in both domains might be explained because in the gaussian chess domain there is a unique optimal ranking of instances (or ROC curve): this is the classification obtained by locating 8 prototypes in the gaussian centers. It is therefore not possible for the MO approach to identify solutions that outperform this optimal classifier. In the irregular case, there is either no unique optimal ranking (ROC curve) that can be represented by prototypes (due to the irregularity of the domain) or it is very hard for MSOPS-II to locate it. So, in this case, the MO approach achieves an advantage over the SO approach through the inclusion of different classifiers in the Pareto approximation set; each one of these will be appropriate for a different range of operating conditions. A visual inspection of all 25 runs for the Irregular Chess domain with 12 prototypes shows that the solutions returned by the MO and SO approaches fall into one of the two categories illustrated in Figures at the top (first category) and figures at the bottom (second category) of Figure 4 (which has been zoomed to $[0, 0.3] \times [0, 0.3]$). It can be observed that the MO approach outperforms the SO method either because it obtains better results for operating conditions near the x-axis (first category, figures at the top in Figure 4) or for operating conditions close to the y-axis (second category, figures at the bottom in Figure 4). Contrariwise, in the Gaussian Chess domain, the ROC curves returned from the MO and SO approach are very similar in all runs (results not shown).

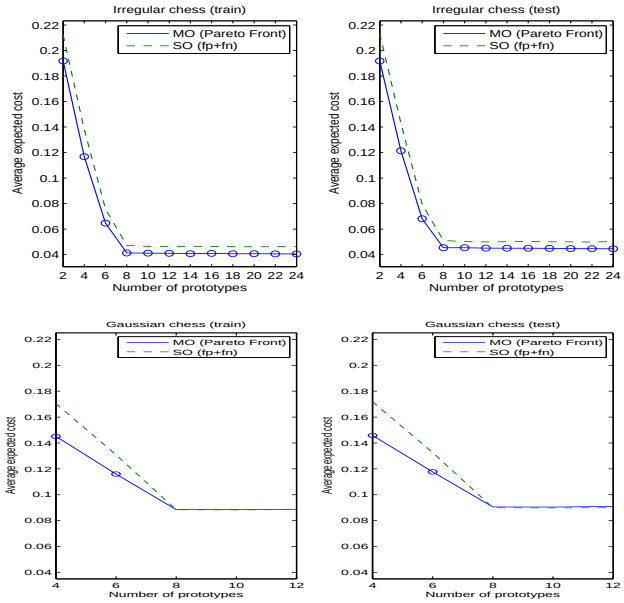

Figure 3: Average expected cost for the synthetic domains (training data on the left, testing data on the right): irregular chess (top) and gaussian chess (bottom).

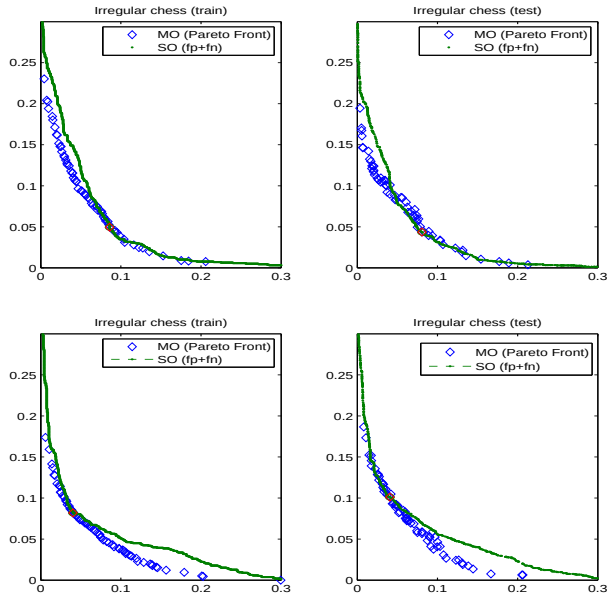

Figure 4: MO (diamonds) vs SO (dots) for the Irregular Chess domain with 12 prototypes. Category 1 (top). Category 2 (bottom).

As an illustration, Figure 5 show the cost curves corresponding to the ROC curves in Figure 4. The area under the cost curve is equivalent to the average expected cost computed in Section 4.1.

Figures 6 and 7 display the results obtained for the real domains. It can be seen that, on the training data, the solutions returned by the MO approach perform significantly better than those from the SO approach, for all of the domains tested. This result continues to hold for the testing data. Specifically, the solutions returned by the MO approach have a lower cost than those for the SO approach, for the entire range of complexities tested. This includes the number of prototypes that leads to the best performance on the test set.

5 Conclusions

This paper extends previous investigations regarding multi-objective approaches to ROC curve generation, by providing empirical results concerning the performance of a multi-objective approach to ROC curve generation in comparison to a standard threshold-moving approach. In particular, the Pareto front approximations generated by MSOPS-II are compared with the ROC curves generated through threshold-moving for a single solution selected from MSOPS-II solution set. Results on a number of synthetic and real domains show that the multi-objective approach consistently provides ROC curves that are superior to those

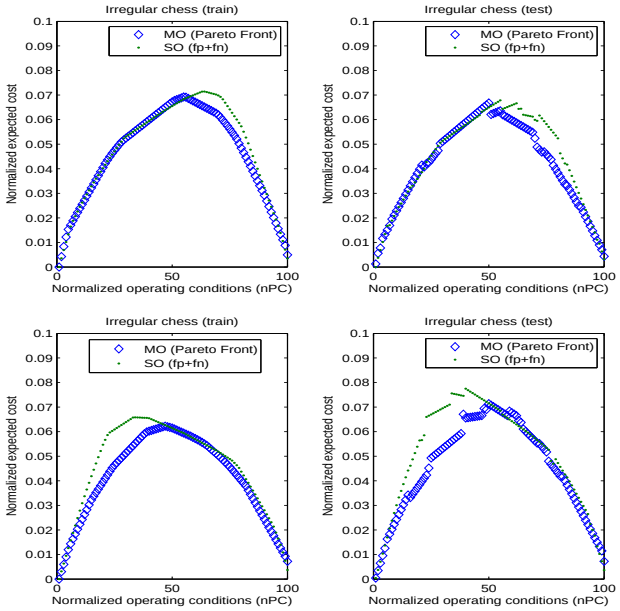

Figure 5: Cost curves corresponding to ROC curves of figures in Figure 4. Category 1 (top). Category 2 (bottom).

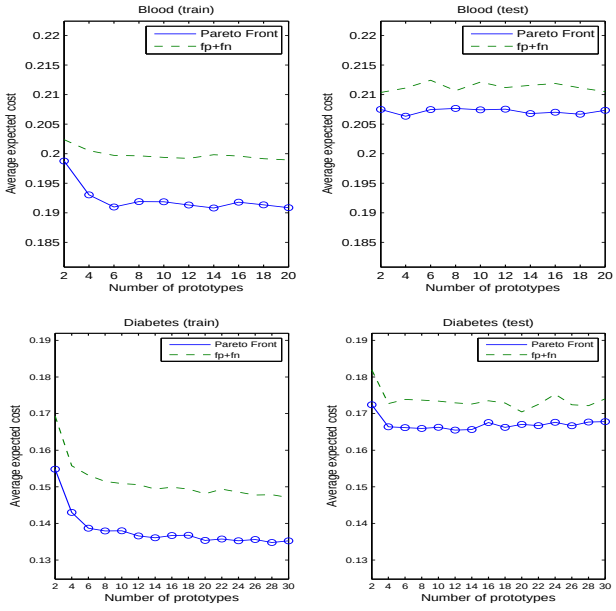

Figure 6: Average expected cost for domains Blood (top) and Diabetes (bottom). Training data on the left, testing data on the right.

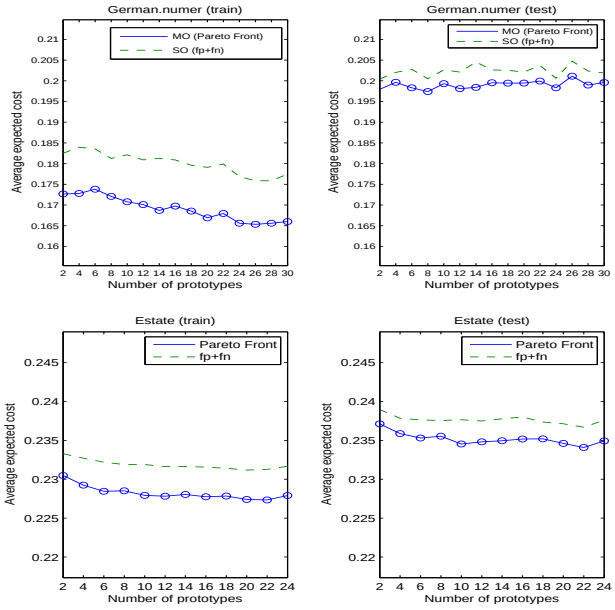

Figure 7: Average expected cost for domains German Number (top) and Estate (bottom). Training data on the left, testing data on the right.

obtained from standard threshold-moving. The experiments are limited to the use of a prototype-based classifiers and future research may extend this analysis to more powerful classification techniques.

The experimental design of this study included a couple of aspects that warrant some further discussion.

First, MSOPS-II is able to approximate concave Pareto fronts, in principle. Given that only the convex hull of the fronts is important in ROC curve generation, it would be interesting to study if simpler aggregated weighted functions, which can only approximate convex fronts, may be able to improve the efficiency of the search.

Second, the paper has developed a novel methodology to compare the ROC curves obtained from the MO and SO approaches. In order to account for possible discrepancies between the ROC curves obtained on testing and training data, the ROC curves are compared by computing the average expected cost (or equivalently, the area under the cost curve) on the test set, while selecting operating the conditions based on the training ROC curve.

Third, by selecting solutions for threshold-moving from MSOPS-II's solution set, we have tried to ensure that the MO approach and the threshold-moving ROC are obtained under the same conditions. This means that differences observed between the MO and SO approach are not due to differences in the optimization algorithm used. It is, however, possible that the multi-objective formulation of the problem may also be advantageous in the sense of changing

the difficulty of the search. Dependent on the formulation of the problem, a SO optimizer may sometimes get stuck in local optima, where a MO optimizer can escape [11]. In the context of ROC curves, some evidence for this is given in [13] for a simple NN example, and it may be worth exploring this issue further.

Two-class domains have been used in this paper because simple thresholding becomes NP-hard for more than two classes. In the future, it is intended to extend the comparison for more than two classes by using heuristic approximations to threshold-moving (such as hill-climbing [14]).

Acknowledgements

This work has been funded by the Spanish Ministry of Science under contract TIN2011-28336 (MOVES project).

References

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

[2] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923, 1998.

[3] Chris Drummond and Robert C. Holte. Cost curves: An improved method for visualizing classifier performance. *Machine Learning*, 65(1):95–130, 2006.

[4] Richard M. Everson and Jonathan E. Fieldsend. Multi-class ROC analysis from a multi-objective optimisation perspective. *Pattern Recogn. Lett.*, 27(8).

[5] Tom Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8).

[6] Tom Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical report, 2004.

[7] Peter A. Flach and Edson Takashi Matsubara. On classification, ranking, and probability estimation. In *Probabilistic, Logical and Relational Learning - A Further Synthesis*, 2007.

[8] Lars Gräning, Yaochu Jin, and Bernhard Sendhoff. Generalization improvement in multi-objective learning. In *IJCNN*, pages 4839–4846, 2006.

[9] Barbara Hammer, Marc Strickert, and Thomas Villmann. Relevance LVQ versus SVM. In *Artificial Intelligence and Soft Computing, Volume 3070 of Springer Lecture Notes in Artificial Intelligence*, pages 592–597. Springer, 2004.

[10] Evan J. Hughes. MSOPS-II: A general-purpose many-objective optimiser. In *IEEE Congress on Evolutionary Computation*, pages 3944–3951, 2007.

[11] J. Knowles, R. Watson, and D. Corne. Reducing local optima in single-objective problems by multi-objectivization. In *Evolutionary Multi-Criterion Optimization*, pages 269–283. Springer, 2001.

[12] Joshua D. Knowles and David Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[13] Matthew A. Kupinski and Mark A. Anastasio. Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characteristic curves. *IEEE Transactions on Medical Imaging*, 18:675–685, 1999.

[14] Nicolas Lachiche and Peter Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves, 2003.

[15] Thomas C. W. Landgrebe and Robert P. W. Duin. Approximating the multiclass ROC by pairwise analysis. *Pattern Recogn. Lett.*, 28(13).

[16] Jose G. Moreno-Torres, José A. Sáez, and Francisco Herrera. Study on the impact of partition-induced dataset shift on k -fold cross-validation. *IEEE Trans. Neural Netw. Learning Syst.*, 23(8):1304–1312, 2012.

[17] E. Mwebaze, P. Schneider, F.M. Schleif, JR Aduwo, JA Quinn, S. Haase, T. Villmann, and M. Biehl. Divergence-based classification in learning vector quantization. *Neurocomputing*, 74(9):1429–1435, 2011.

[18] Basilio Noris. Mldemos, 2011. `http://mloss.org/software/view/278/`.

[19] Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Mach. Learn.*, 42(3).

[20] Atsushi Sato. Discriminative dimensionality reduction based on generalized LVQ. In *ICANN*, pages 65–72, 2001.

[21] Panu Somervuo and Teuvo Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10(2):151–159, 1999.

[22] Ashwin Srinivasan. Note on the location of optimal classifiers in n-dimensional ROC space. Technical report, 1999.

[23] Marc Strickert. Generalized matrix relevance learning vector quantization grlvq, 2011. `http://mloss.org/software/view/323/`.

[24] Ian H. Witten, Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with java implementations, 1999.