Multi-Objective Evolutionary Optimization for Generating Ensembles of Classifiers in the ROC Space

Julien-Charles Lévesque †
julien-charles.levesque.1@ulaval.ca

Christian Gagné †
christian.gagne@gel.ulaval.ca

† Laboratoire de vision et systèmes numériques
Dép. de génie électrique et de génie informatique
Université Laval, Québec, QC, Canada

Audrey Durand †
audrey.durand.2@ulaval.ca

Robert Sabourin ⋆
robert.sabourin@etsmtl.ca

⋆Laboratoire d'imagerie, de vision et d'intelligence artificielle
École de technologie supérieure
Université du Québec
Montréal, QC, Canada

ABSTRACT

In this paper, we propose a novel approach for the multi-objective optimization of classifier ensembles in the ROC space. We first evolve a pool of simple classifiers with NSGA-II using values of the ROC curves as the optimization objectives. These simple classifiers are then combined at the decision level using the Iterative Boolean Combination method (IBC). This method produces multiple ensembles of classifiers optimized for various operating conditions. We perform a rigorous series of experiments to demonstrate the properties and behaviour of this approach. This allows us to propose interesting venues for future research on optimizing ensembles of classifiers using multi-objective evolutionary algorithms.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Search**]: Heuristic methods; I.5.2 [**Design Methodology**]: Classifier design and evaluation

Keywords

Machine learning, ROC curve, Multi-objective evolutionary algorithms, Ensembles of classifiers, Genetic programming

1. INTRODUCTION

Designing classifiers for prediction of events or pattern recognition is a complex task, traditionally conducted by optimizing a single criterion: prediction accuracy. This criterion falls short of expectations when working with skewed class distributions, unequal classification error costs, and in presence of dynamic operating conditions. This problem can be solved by using a more complex criterion for training classifiers: the Receiver Operating Characteristics (ROC) curve.

A ROC curve is a tool for visualizing and selecting classifiers based on error rates per class, instead of just the errors of both classes. This graph provides a lot more information than the single accuracy measurement and allows the development of methods that better adapt to varying conditions [10]. ROC curves can be described with a single value by computing the Area Under the ROC Curve (AUC). This has been used by many machine learning researchers to replace accuracy as the objective during optimization, with a certain degree of success [5, 23, 25].

One problem with using the AUC as the principal objective for optimization is that two classifiers can produce the same AUC, but their ROC curves can be very different. These two 'identical' classifiers can be optimal for very different operating conditions (class skew and error costs), meaning that the AUC criterion is not rich enough. In this particular case, a way to solve the problem is to keep both classifiers, thus creating a pool of classifiers, and to select the better classifier at a later time based on operating conditions [13, 20].

Such a problem is very close to a Multi-Objective Problem (MOP) because it is the maximization of many different objectives at the same time. One of the best tools available today to solve MOPs are elitist Multi-Objective Evolutionary Algorithms (MOEA). They have been shown to perform extremely well on classical MOPs [7, 26].

In this paper, we introduce a novel approach that targets the multi-objective optimization of simple classifiers to maximize ROC performance, where a simple classifier refers to a computionally simple classifier (e.g., a linear discriminant). Our method begins with an adaptation of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [7] to evolve simple classifiers. The resulting classifiers are then combined at the decision level using Boolean functions [16, 24]. The evolution of the pool of classifiers aims to provide a better input for the combination layer.

Our experiments show that the joining of such simple classifiers with Boolean combination provides interesting performances compared with more complex approaches. We extensively study the behavior of those evolved simple clas-

GECCO'12, July 7-11, 2012, Philadelphia, Pennsylvania, USA.

sifiers combined at the decision level and draw some insights and suggestions from our observations.

A review on the usage of ROC curves as an objective for classifier training is presented in the next section. Our proposed evolution methodology is described in Section 3, and the Boolean combination of the resulting classifiers based on [16] is covered in Section 4. Finally, Section 5 and 6 details the experiment and analysis of the proposed approach.

2. OPTIMIZING THE ROC CURVE

As stated in the introduction, the ROC curve is a better objective for training classifiers than accuracy. Let us introduce some concepts and notation concerning ROC curves before going further. First, let us take input samples $X = \{x_1, x_2, ..., x_n\}$ with labels $Y = \{y_1, y_2, ..., y_n\}$, where $y_i \in \{-1, 1\}$, and a classifier emitting predictions as to the labels of those input samples. The True Positive Rate (TPR or TP rate) is the proportion of positive samples ($y = 1$) correctly classified, while the False Positive Rate (FPR or FP rate) is the proportion of negative samples ($y = -1$) incorrectly classified.

A *discrete* or *binary* classifier directly outputs labels for each input sample provided and corresponds to a fixed point in the ROC space. A *scoring* or *ranking* classifier produces scores or probabilities. The labels are obtained by applying a threshold on those scores, turning the classifier into a discrete classifier (typically the threshold is 0 for scores and 0.5 for probabilities). By varying the threshold, we can obtain all possible FP/TP rates for a single classifier. The ROC curve is the plotting of the TP rates against the FP rates - the space defined by FP and TP rates is called the ROC space.

An important concept for ROC curve optimization and exploitation is the ROC Convex Hull (ROCCH) (also called maximum realizable ROC in other works), introduced by Provost and Fawcett [21]. A classifier is potentially optimal if and only if it lies on the convex hull of the set of points in ROC space [10]. From this, the pool of classifiers can be pruned by only keeping classifiers that lie on the convex hull.

Ensembles of classifiers have been used in the past based on the idea that many classifiers may be preferable to a single classifier when solving a given problem [19]. One could argue that it is easier to optimize many classifiers locally than to optimize one classifier globally, and that the same also holds for ROC curves. Some great work has been conducted recently on the combination of classifiers with the objective of improving ROC performance. Marrocco et al. studied the linear combination of several dichotomizers or linear discriminants trained on pairs of features [20]. They show that their combination method geared towards ROC performance is superior than other combination methods.

Kumar et al. use particle swarm optimization to evolve score-level combination rules [18]. However, they keep a fixed threshold producing only a single operating point, thus removing the ability to adapt to changing conditions. Sebag et al. evolve hypotheses using evolution strategies and the AUC as an optimization criterion [23]. They also fix the thresholds of their classifiers.

2.1 ROC optimization using MOEA

An interesting means of optimizing the ROC performance is to formulate the problem as a multi-objective one, with the objectives being the minimization of FP rates and the maximization of TP rates. This formulation provides richer information than a single numeric value such as the AUC. Consider the example of two different classifiers that have the same AUC but excel in different areas (as shown in Figure 1). If the operating conditions are not known beforehand, both of these classifiers are equally valuable. The goal of using MOEA in this setting is to produce classifiers that are locally optimal, remembering the conditions in which each classifier is optimal to select the appropriate one at operation time. Using multi-objective terminology, the two classifiers of Figure 1 are both non-dominated and Pareto optimal.

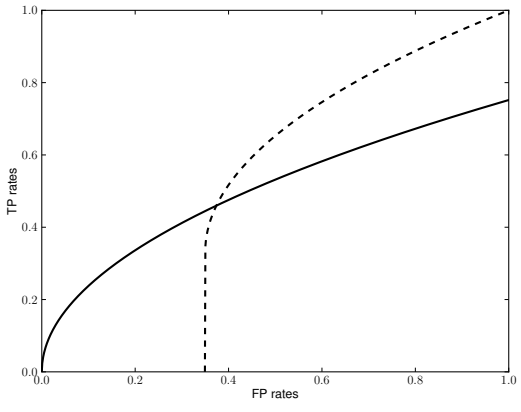


Figure 1: Two very different ROC curves that are equal according to the AUC criterion.

Bhowan et al. use MOEAs to evolve expression trees and then combine them using ensemble learning techniques such as majority vote or negative correlation learning [1, 2]. In order to be able to use their classifiers as a classical ensemble, they force each classifier to adopt a single threshold, thus they evolve single points in ROC space instead of entire curves. This is because traditional ensemble fusion techniques are not adapted to be used in ROC space. Khreich et al., however, have studied and developed such fusion techniques and we use them in this work [16].

Everson and Fieldsend consider a multi-class generalization of ROC analysis from a multi-objective optimization perspective [9]. They use SPEA to find Pareto optimal solutions of their multi-class ROC surfaces. Their formulation of the multi-class ROC curve is of interest, but here we consider only binary classification problems.

Chatelain et al. also explore the multi-objective evolution of classifiers. They optimize the hyperparameters of SVM classifiers using their resulting FP and TP rates as performance criteria [4]. They look for a pool of parametrized classifiers corresponding to the optimal set of FP/TP trade-offs. This optimal set of trade-offs is viewed as the Pareto front of a multi-objective optimization problem. Chatelain et al. use NSGA-II because it converges close to the optimal Pareto front while preserving a diverse pool of solutions and also because it executes faster than other algorithms. NSGA-II is a population-based, elitist approach that preserves diversity through the integration of a distance metric in the selection process. Each individual in the population

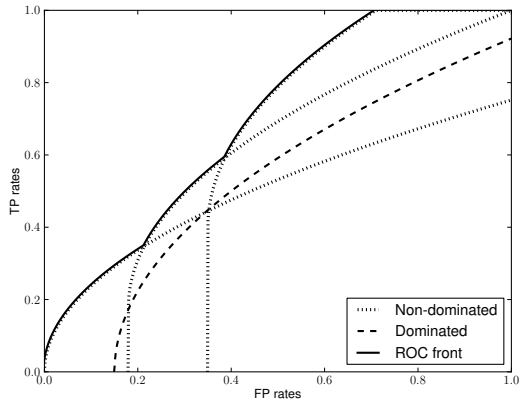


Figure 2: Four different ROC curves, their domination status and their pareto Front (also called ROC front).

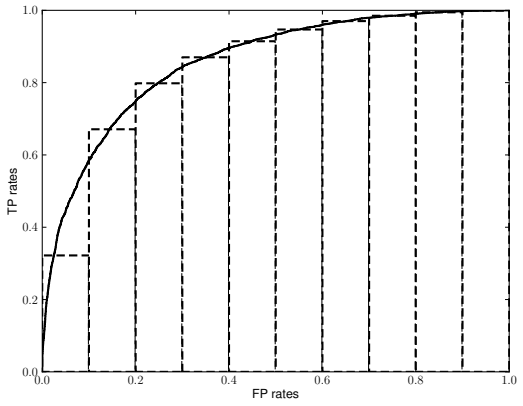


Figure 3: Estimating a ROC curve from a dataset as a histogram. For each bin, we compute the average of the TP rates contained within.

represents a classifier by encoding the classifier's hyperparameters. Their fitness is represented by a ROC curve, thus we have multiple pairs of FP/TP rates for each classifier. The difference with traditional MOPs is that each individual provides multiple fitness values in the form of a ROC curve. Adapting the dominance concept from multi-objective optimization theory, an individual is said to be non-dominated if at least one FP/TP pair present in its ROC curve sits on the Pareto front of the ROC space (a front that is also called ROC front). Figure 2 shows examples of dominated and non-dominated ROC curves.

The result of such an optimization process is a pool of Pareto optimal classifiers, each residing on the ROC front (as shown in Figure 2). These classifiers are all optimal for their respective operating conditions, enabling us to select an optimal classifier for every condition.

3. EVOLVING A POOL OF SIMPLE CLASSIFIERS

Globally, our model consists of the evolution of a pool of simple classifiers using the ROC curve as an objective and their combination using Boolean fusion functions. The evolved pool of classifiers should provide a better input to the Boolean fusion stage than classically trained classifiers. We select simple classifiers as the basis for this system to keep inference and training times low. This constitutes another novel aspect of this work - the evolution of simple classifiers in a multiobjective setting (as opposed to complex ones).

The proposed method begins with the training of a pool of simple classifiers using NSGA-II similarly to what is explained in Section 2.1. In this work, the simple classifiers are either linear discriminants or expression trees, although they could be any type of classifiers. Then, taking the pool of evolved classifiers, their class predictions are combined using different Boolean functions (AND, OR, XOR, etc.) to further improve the ROC performance.

In our approach, classifiers are formulated as models in the MOEA, rather than being represented by their hyperparameters as in the work by Chatelain et al. [4]. The parameters or models are now directly affected by mutation and crossover operators. This removes the need to run a separate training for each individual before evaluating their fitness, since the individuals become directly usable models. For example, when evolving SVM hyperparameters, one needs to fit a model on the data before computing the ROC curve (training an SVM is approximately $O(n^2)$, where n is the training set size, thus computing costs can escalate quickly).

Another slightly different point in our approach is the formulation of the problem and its fitness. Each ROC curve is presented as a histogram of TP rates against fixed FP rates (as demonstrated in Figure 3). The number of bins used is the maximum allowed by the dataset, thus there is no estimation error. For two different ROC curves estimated with the same dataset, the bins are exactly aligned and this allows the direct comparison of two regions. Those values become the objectives of our MOP (there are as many objectives as there are bins for the ROC curves). The MOP becomes the maximization of the TP rate in every bin, which implicitly minimizes the overall FP rates.

The comparison between two curves changes with this new formulation. To better illustrate this, let us recall the domination concept (Algorithm 1). This operator is applied to each pair of individual in the population, and non-dominated individuals constitute the first front. A second front is then drawn in the same manner on the set of all individuals excluding the ones on the first front, a third front with the remaining, and so on. With the previous formulation of Chatelain et al. [4], a ROC curve was non-dominated if and only if it touched the ROC front. In our formulation, a ROC curve is non-dominated only if it is better than all the other ROC curves *in at least one bin* (for each comparison it can be a different bin). For a ROC curve to be dominated, it needs to be below or equal to another ROC curve for *all* bins or operating points. We compare complete *curves* instead of just *operating points* (single FP/TP pairs).

NSGA-II sorts individuals into fronts based on their Pareto dominance, then within a single front sorts the individuals again based on a crowding distance, an estimate of solution diversity. The crowding distance for a given individual is the

Algorithm 1 Domination between two ROC curves

Input: Two curves r_1 and r_2, the bins of each curve must be aligned
Output: True if r_1 dominates r_2, False otherwise.

```
for bin_i ∈ 1, ..., num_bins do
  if r1.bin_i > r2.bin_i then
    return True
  end if
end for
return False
```

sum over all objectives of the Manhattan distance between the individual and its two closest neighbours. This crowding distance attributes ∞ to the lowest and best individual for each objective. In our case, these correspond to the lowest and highest individual of each bin - this means that for every bin, the worst and the best individual are strongly advantaged. This helps generate more diversity in the pool.

Such a pool of classifiers can be used by itself to produce predictions and a ROC curve. Given the pool of classifiers resulting from the optimization, only the ones on the ROC front should be stored. The thresholds required to produce the FP/TP values that reside on the ROC front are then stored. Finally, these stored classifiers and their thresholds are applied on the testing dataset, creating new testing FP/TP pairs that will become the testing ROC curve. In Section 5, we will use this methodology to assess what type of gain is provided by the combination covered in the next section.

4. BOOLEAN COMBINATION

We now study the combination of the resulting classifiers in ROC space to further push the ROC front. To each of the evolved classifiers corresponds a ROC curve, i.e. a set of FP/TP rates that represent the performance obtained by a series of different thresholds. Khreich et al. have shown that the Boolean combination of two classifiers can produce new operating points in the ROC space, possibly yielding better performance than the ROC curve of the pool [16]. The method has also been used and explained in greater detail in [17] and [14].

The Boolean combination of two crisp classifiers is rather straightforward - for every input sample provided, apply a given Boolean function (e.g., AND or OR) to the output of those two classifiers and this becomes the new prediction of the pair. New FP and TP rates are computed from these predicitions, and only the ones improving performance are saved.

The combination of classifiers with real-valued outputs requires their ROC curves and corresponding thresholds. For every threshold of each ROC curve, the predictions are computed and combined with a Boolean function, resulting in a new point of operation in ROC space. Once the computation of these FP/TP rates is finished, we find the points lying on the convex hull. These points correspond to the optimal combinations - they are saved while the others are discarded. To ensure that the method provides equal or better performance, the ROC curves of the initial classifiers (before combination) are always added to the pool of ROC points before drawing the convex hull. The Boolean combination of two classifiers with real-valued outputs is described

Algorithm 2 BC_ALL : Boolean combination of two ROC curves (introduced in [16])

Input: Thresholds of ROC curves, T_a and T_b, and labels
Output: ROCCH and fused responses (R_{ab}) of combined curves, where each point is the result of two fused thresholds along with the corresponding Boolean function (bf)

```
1. m ← length(T_a)
2. F ← empty array of size [2, m^2]
3. boolean_functions ← [a ∧ b, ¬a ∧ b, a ∧ ¬b, ¬(a ∧ b), a ∨ b, ¬a ∨ b, a ∨ ¬b, ¬(a ∨ b), a ⊕ b, a ≡ b ]
4. for bf ∈ boolean_functions do
5.   for i ∈ 1, ..., m do
6.     R_a ← (T_a ≥ T_ai)
7.     for j ∈ 1, ..., m do
8.       R_b ← (T_b ≥ T_bi)
9.       R_c ← bf(R_a, R_b)
10.      Compute new ROC curve using R_c and labels
11.      Push FP/TP rates onto F
12.    end for
13.    Compute ROCCH_new of F
14.    Store thresholds and corresponding Boolean functions that exceeded the ROCCH_old in S_global
15.    Store the responses (predictions) of these emerging points in R
16.    ROCCH_new ← ROCCH_old
17.  end for
18. end for
19. return ROCCH_new, R, S_global
```

in Algorithm 2. The result of a typical run on two simple classifiers for a synthetic dataset is presented in Figure 4.

The complete algorithm, Iterative Boolean Combination (IBC), begins with the combination of a pair of classifiers using BC_ALL. The outcome of this combination is a series of responses (predictions) along with their thresholds and Boolean functions. This series of responses is then combined with a third classifier, and again with a fourth, and so on until all classifiers have been merged.

The resulting responses are then recombined with each individual ROC curve, drawing a convex hull after each step. This step is repeated iteratively until a maximum iteration number is reached or the gain in AUC drops below a certain threshold. This step can further improve the ROC performance and usually converges after two or three iterations. An interesting property of IBC is that the ROC curve on the training dataset has no concavities. Another property is that the AUC of the ensemble increases monotonically with each additional ROC curve provided, i.e., the performance can only improve as more ROC curves are added. Each vertex on the convex hull resulting from the application of IBC on N ROC curves represents an ensemble of models and a Boolean fusion function adapted to their context. The cardinality of each of those ensembles varies between 1 and N.

5. EXPERIMENTS

In order to study the impact of the proposed method on ROC performance, we have conducted a series of experiments on standard datasets. We evolve two types of simple classifiers: linear discriminants and expression trees. Even simple classifiers such as these should achieve good ROC fronts with low computing costs when trained with MOEAs.

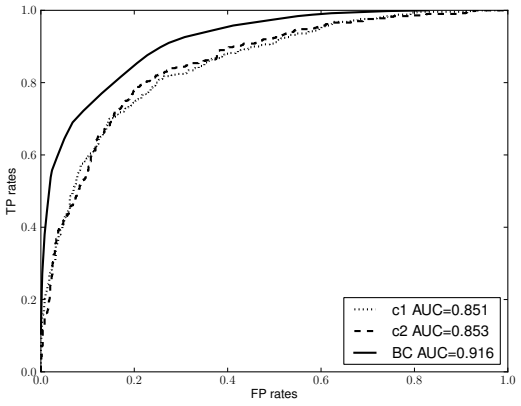


Figure 4: Boolean combination of two simple classifiers.

We use the DEAP library [6] which provides implementations of NSGA-II and Genetic Programming (GP).

In all cases, both the base classifiers and the Boolean combination are determined using the same training dataset since empirical testing did not reveal any gain from splitting the training dataset. The number of bins for the ROC curves in the evolution section is fixed to the number of negative samples in each case, thus no approximations of the ROC curves are conducted given the dataset used for estimating it. Finally, the number of thresholds for IBC evaluation of ROC curves is empirically set to 100.

We use six datasets from the UCI machine learning repository [12], namely Australian Credit (Aust.), WDBC, Breast Cancer (Bc.), Ionosphere (Ion.), Heart, and Pima. All tests are executed using 5-fold cross-validation.

5.1 Linear discriminants

The first type of base classifiers used for constructing the pools are linear discriminants. They consist of a vector of weights in $[-1, 1]$, one for each dimension of the input samples. The same crossover and mutation operators were used as in NSGA-II: simulated binary crossover and Gaussian mutation [7]. The probability of crossover is fixed to 0.9, and the probability of mutation is fixed to $1/nb\_weights$.

The population size in this case is 500 and the maximum number of generations is 500 - we will further discuss the impact of these parameters in Section 6.

5.2 Expression trees

The expression trees are evolved by GP and require different crossover and mutation operators, but the selection operators and fitness remain the same. The operators used for the GP classifiers are $+$, $-$, $\times$, a protected division operator, which returns zero in case of division by zero, and ephemeral random constants picked from a uniform distribution spanning $[-1, 1]$. Variables used for building the expression trees are $F1, F2, \ldots, Fn$, where Fi corresponds to the i-th feature of the dataset.

The expression trees are generated completely full and have a maximum depth of 5. A standard GP crossover operator is used which swaps subtrees between individuals. The mutation operator is standard subtree mutation, where one randomly selects a node in the tree and replaces it and its subtree by a randomly generated subtree. The probablilities for mutation and crossover remain the same. For expression trees, we fix the population size to 500 and a smaller maximum number of generations, 100, is used.

5.3 Results

The testing performance is compared with the method developed by Chatelain et al. [4] and results for the best single classifier in the literature are also borrowed from their paper. We must rely on AUC as a criterion for comparison because of space restrictions and also to simplify analysis. We can safely say that given a choice, without prior knowledge of operating conditions, the classifier (or ensemble of classifiers) with the highest AUC would be the better choice. The AUC performance is thus presented in Table 1 for the single best classifier found in literature, a pool of SVMs [4] evolved by NSGA-II (SVM-P), linear discriminants evolved by our adaptation of NSGA-II combined with IBC (L-IBC), and the same with expression trees (GP-IBC). As a measure of the performance stability, the standard deviation is computed on each fold.

The comparison with Chatelain's methods requires additional comments, as they state in their paper that they used "the area under the ROC front" (on the test dataset). This way of measuring performance is overly optimistic. Our method's performance was not evaluated in this manner, so this might introduce some imbalances in performance.

We can see that by evolving very simple classifiers and combining them with IBC, we obtain comparable performances to the literature. For the Ionosphere and Heart datasets, the performance is worse than the basic approach in the literature. This might be due to the fact that these are extremely small datasets (351 and 270 data points), thus a 10-fold testing approach might have been more appropriate.

Our approach manages to perform well considering the low computational cost for its operation. SVMs have a training complexity of $O(n^2)$, where n is the number of training samples, while our approach requires no training, making it $O(1)$ in training. In both cases, the time complexity for NSGA-II is comparable because it requires the comparison of all FP/TP pairs in each ROC curve. The time complexity for IBC is $O(t^2 + t \cdot m)$, where t is the number of thresholds for each classifier and m is the number of classifiers in the pool. The complexity of our approach does not grow significatively with the dataset size, while the complexity of the SVM pool grows quadratically with it. However, the complexity of inference is comparable for both models. The complexity of operation for our model is the same as IBC ($O(m)$), and the complexity for SVMs is $O(s)$, where s is the number of support vectors.

If we consider that the pool of SVMs classifiers reaches the optimal Pareto front on the test set, we can say that our method nearly obtains this performance. Essentially, these results show that there is a benefit to combining evolved simple classifiers in ROC space. We conducted further experiments to develop a better understanding of our method, they are presented in the next section.

6. INTERNAL COMPARISONS

As was stated in Section 3, we can evaluate the testing performance of the base classifiers by placing each classifier's ROC curve in the same space, drawing the convex hull of

Table 1: AUC performance for two previous methods and our two ensembles.

Dataset	Single classifier		SVM-P [4]	L-IBC	GP-IBC
Australian	90.25 ± 0.6	[25]	96.22 ± 1.7	94.36 ± 2.38	94.79 ± 1.02
WDBC	94.70 ± 4.6	[11]	99.59 ± 0.4	99.23 ± 0.46	99.72 ± 0.33
Breast cancer	99.13	[3]	99.78 ± 0.2	98.66 ± 1.32	99.36 ± 0.54
Ionosphere	98.70 ± 3.3	[22]	99.00 ± 1.4	93.92 ± 5.06	94.46 ± 3.47
Heart	92.60 ± 0.7	[25]	94.74 ± 1.9	91.85 ± 4.90	91.79 ± 4.49
Pima	84.80 ± 6.5	[5]	87.42 ± 1.2	85.53 ± 2.23	85.22 ± 1.23

this set and keeping the operating points that were located on the convex hull. These operating points are represented by a classifier and a fixed threshold. By applying the same thresholds to the same classifiers, we can evaluate testing performance.

Table 2 presents the performance for each type of base classifier, alone, and combined, on the same datasets as the experiments in the previous section. We can see that the base classifiers wield a poor performance, especially the expression trees. This means that combination by IBC is a crucial element of our approach. Even though the linear discriminants appear better, the gain provided by combining the expression trees is higher, making them better overall.

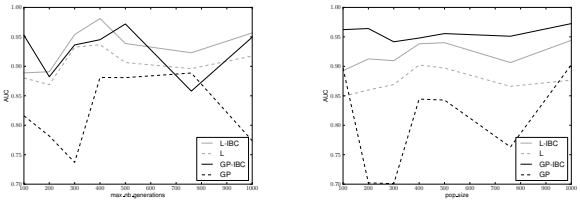
Table 2: Comparison of AUC performance for pools of base classifiers with their combination using IBC. L stands for linear base classifiers and GP for expression trees.

Data	L	L-IBC	GP	GP-IBC
Austr.	88.83±2.70	94.36±2.38	79.72±8.28	94.49±1.02
WDBC	94.53±3.67	99.23±0.46	92.13±3.80	99.72±0.34
Bc	93.59±6.83	98.66±1.32	94.60±2.73	99.36±0.54
Ion.	85.56±5.27	93.92±5.06	83.69±8.11	94.46±3.47
Heart	86.17±5.00	91.85±4.90	80.29±6.53	91.79±4.49
Pima	79.51±2.52	85.53±2.33	73.62±3.95	85.22±1.23

Experiments were run to study the impact of the number of generations and the population size on performance using the Australian credit dataset. Figure 5(a) shows the evolution of performance for a fixed population size (100) and varying numbers of generations in $[100, 1000]$ and Figure 5(b) shows the evolution of performance for a fixed number of generations (100) and varying sizes of populations in $[100, 1000]$. We can see that while the pool of linear discriminants has a higher performance than expression trees for almost all settings, GP-IBC is often superior to L-IBC. Interestingly, performance between L and L-IBC is strongly correlated which is not the case for GP and GP-IBC. Another conclusion to draw from these graphs is that the performance of GP-IBC does not improve with a high number of generations.

6.1 Comparison with diverse classifiers

To assess whether or not our evolved classifiers form a better input to IBC than regular classifiers, we computed the performance of IBC executed on a pool of classifiers directly fit on the input data instead of using an MOEA. In a manner inspired by random subspaces [15], the linear discriminants are trained on a subspace of the input dimensions different for each classifier, providing diversity for IBC. Each discriminant is assigned a random number of input dimensions to use for learning uniformly sampled between 2 and the total number of dimensions of input data. Each of these dimensions is also sampled uniformly. The classifiers are then trained on their respective subspaces by least squares regression, and used as base classifiers for IBC. The number of such linear discriminants is set to 500. Table 3 presents the performance obtained using such linear discriminants trained on random subspaces (LRS-IBC) compared with the performance of expression trees (GP-IBC).



(a) Number of generations (b) Population sizes

Figure 5: Evolution of performance for varying numbers of generations and population sizes using the Australian credit dataset.

Table 3: Comparison of AUC performance for GP-IBC (expression trees) and LRS-IBC (linear classifiers trained on random subspaces - not evolved)

Dataset	GP-IBC	LRS-IBC
Australian	94.79 ± 1.02	94.72 ± 2.54
WDBC	99.72 ± 0.34	99.38 ± 0.74
Breast cancer	99.36 ± 0.54	99.39 ± 0.65
Ionosphere	94.46 ± 3.47	94.46 ± 3.47
Heart	91.79 ± 4.49	92.43 ± 4.41
Pima	85.22 ± 1.23	85.02 ± 1.57

We can see that the performance of LRS-IBC is very close to that of GP-IBC, a perhaps counterintuitive result. Thus the most performant classifiers are not necessarily the best input for IBC. According to these results, we should put more emphasis on keeping a strong diversity in the pool of base classifiers. This explains why GP-IBC performs slightly better than L-IBC - because the pool is more diverse.

6.2 Overfitting

The final experiments are aimed at analyzing the occurence of overfitting. Figures 6 and 7 present examples of outcomes for the evolution chained with IBC on two UCI datasets : Australian credit and German credit [12]. The subfigures on the left represent the coverage in ROC space

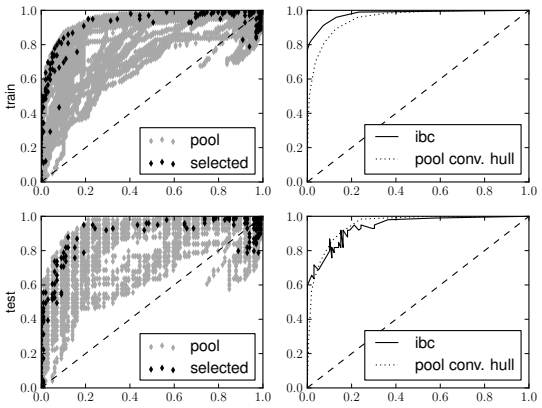


Figure 6: The final result for the evolution of a pool of expression trees and their combination with IBC (Australian credit dataset).

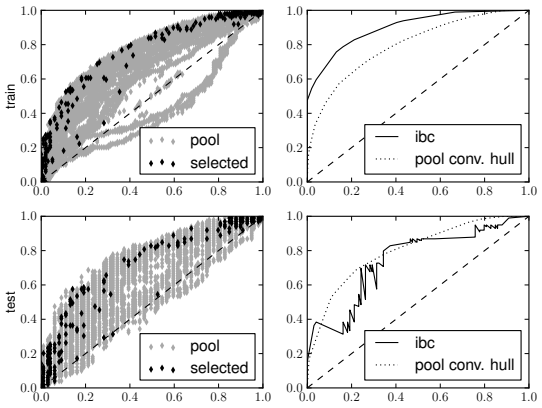


Figure 7: The final result for the evolution of a pool of expression trees and their combination with IBC (German credit dataset).

by the final classifier pool in grey along with the classifiers selected by IBC in black. The two upper subfigures represent training performance while the two lower subfigures represent testing performance. In the right part, the pool convex hull (conv. hull) represents the best performance achievable by the pool of classifiers. Each point on the IBC ROC curves represents an ensemble of simple classifiers, with different thresholds and different Boolean functions. Those are the best operating points found by IBC during its training phase.

In the right-side subfigures, we can see that the classifiers selected by IBC provide a strong improvement over the pool of classifiers in training, but this is lost in the testing phase, indicating that there is overfitting to some degree. This overfitting is also the source of the concavities observed on the IBC testing ROC curves. A possible explanation to this phenomenon is that the base classifiers evolved by NSGA-II are overfit on the training data, thus are unable to generalize properly. If the base classifiers cannot generalize, then their operating point in ROC space will change significantly between training and testing. From this, it is logical that their combination would also become flawed.

Knowing that overfitting causes concavities in the testing ROC curves, we could say that a proper training terminated before overfitting occurs would produce a similar ROC curve without concavities. Thus, a way to estimate the performance obtainable by removing overfitting would be to use the convex hull of the ROC curves obtained on the testing dataset. Results for convex hulls are presented in Table 4 for the GP-IBC method. From this, we observe that there is room for much improvement for our method by reducing overfitting. To reduce overfitting, an external validation set for evolution could be used, similarly to work done by Dos Santos et al. [8]. By reducing this overfit, it is believed that performance would become comparable with that of the SVM pool (see Table 1), with the added benefit of using relatively simple classifiers.

Table 4: GP-IBC performance and its convex hull

Dataset	GP-IBC	Conv. Hull
Australian	94.79 ± 1.02	96.67 ± 0.60
wdbc	99.72 ± 0.34	99.84 ± 0.25
Breast cancer	99.36 ± 0.54	99.90 ± 0.14
Ionosphere	94.46 ± 3.47	98.81 ± 0.90
Heart	91.79 ± 4.49	97.21 ± 0.90
Pima	85.22 ± 1.23	89.49 ± 2.04

7. CONCLUSION

In this paper, we studied the evolution of simple classifiers in ROC space with NSGA-II and their Boolean combination using IBC. To determine the fitness for the evolution of the simple classifiers, we used the binning of their ROC curves. This allows for a wider range of individuals to be considered as non-dominated. We then combined these individuals at the decision level, i.e. combining the outcome of each threshold for each individual, using Boolean functions, a technique called iterative Boolean combination [16].

We found that this method can perform well compared to previous approaches in the literature. We also found that diversity might be as important or more important than the ROC performance as an objective for evolution especially when the final goal is to combine the evolved classifiers at the decision level. It would be interesting to study the behavior of IBC when trying to feed it only extremely diverse classifiers. However, to our knowledge, there is no diversity measurement in ROC space - this would be an interesting direction for future research. It would also be beneficial for the development of an evolutionary algorithm geared towards optimizing diversity in ensembles.

An important problem encountered was the overfitting of the ensemble on the training data, resulting in concavities in the testing ROC curves. We estimated the performance achievable by removing overfitting as the convex hull of the overfit ROC curve. Further work should consider ways to reduce overfitting such as modifying NSGA-II to use an external validation dataset, as in the work of Dos Santos et al. [8].

8. ACKNOWLEDGMENTS

This research was supported in part by NSERC-Canada, as well as the FQRNT-Québec. It also benefitted from the computing resources provided by the CLUMEQ/Compute Canada. Lastly, we would like to thank Annette Schwerdtfeger for proofreading this paper.

9. REFERENCES

[1] U. Bhowan, M. Johnston, and M. Zhang. Evolving ensembles in multi-objective genetic programming for classification with unbalanced data. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO)*, pages 1331–1338. ACM, 2011.

[2] U. Bhowan, M. Zhang, and M. Johnston. AUC analysis of the pareto-front using multi-objective GP for classification with unbalanced data. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation (GECCO)*, pages 845–852. ACM, 2010.

[3] H. Boström. Maximizing the area under the ROC curve using incremental reduced error pruning. In *Proceedings of the International Conference on Machine Learning 2005 workshop on ROC analysis in machine learning*, 2005.

[4] 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*, 43(3):815–823, 2010.

[5] C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. *Advances in Neural Information Processing Systems (NIPS)*, 16:313–320, 2004.

[6] F.-M. De Rainville, F.-A. Fortin, M.-A. Gardner, M. Parizeau, and C. Gagné. Distributed Evolutionary Algorithms in Python (DEAP). http://deap.googlecode.com.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[8] E. M. Dos Santos, R. Sabourin, and P. Maupin. Overfitting cautious selection of classifier ensembles with genetic algorithms. *Information Fusion*, 10(2):150–162, Apr. 2009.

[9] R. M. Everson and J. E. Fieldsend. Multi-class ROC analysis from a multi-objective optimisation perspective. *Pattern Recognition Letters*, 27(8):918–927, June 2006.

[10] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006.

[11] C. Ferri, P. Flach, and J. Hernández-Orallo. Learning decision trees using the area under the ROC curve. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 139–146, 2002.

[12] A. Frank and A. Asuncion. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml, 2010.

[13] S. Gao, C. Lee, and J. Lim. An ensemble classifier learning approach to ROC optimization. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 679–682. IEEE, 2006.

[14] E. Granger, W. Khreich, R. Sabourin, and D. Gorodnichy. Fusion Of Biometric Systems Using Boolean Combination: An Application to Iris-Based Authentification. *International Journal of Biometrics (IJBM)*, 2011.

[15] T. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[16] W. Khreich, E. Granger, A. Miri, and R. Sabourin. Iterative Boolean combination of classifiers in the ROC space: An application to anomaly detection with HMMs. *Pattern Recognition*, 43(8):2732–2752, 2010.

[17] W. Khreich, E. Granger, A. Miri, and R. Sabourin. Adaptive ROC-based ensembles of HMMs applied to anomaly detection. *Pattern Recognition*, 45:208–230, July 2012.

[18] A. Kumar, V. Kanhangad, and D. Zhang. A new framework for adaptive multimodal biometrics management. *IEEE Transactions on Information Forensics and Security*, 5(1):92–102, 2010.

[19] L. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiler-Interscience, 2004.

[20] C. Marrocco, M. Molinara, and F. Tortorella. On linear combinations of dichotomizers for maximizing the area under the ROC curve. *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics*, 41(3):610–20, June 2011.

[21] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.

[22] A. Rakotomamonjy. Optimizing AUC with support vector machine. In *Proceedings of the European Conference on Artificial Intelligence Workshop on ROC Curve and AI*, 2004.

[23] M. Sebag, J. Azé, and N. Lucas. ROC-based evolutionary learning: Application to medical data mining. In *Artificial Evolution*, pages 384–396. Springer, 2004.

[24] Q. Tao and R. Veldhuis. Threshold-optimized decision-level fusion and its application to biometrics. *Pattern Recognition*, 42(5):823–836, 2009.

[25] S. Wu and P. Flach. A scored AUC metric for classifier evaluation and selection. In *Proceedings of the International Conference on Machine Learning 2005 workshop on ROC analysis in machine learning*, 2005.

[26] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. In *Proceedings of the EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control With Applications to Industrial Problems*, 2001.