

Generic Methods for Evolutionary Ensemble Learning

Christian Gagné*
Informatique WGZ Inc.,
819 avenue Monk,
Québec (QC), G1S 3M9, Canada.
christian.gagne@wgz.ca

Marc Schoenauer
Équipe TAO – INRIA Futurs / CNRS UMR 8623,
LRI, Bat. 490, Université Paris Sud,
91405 Orsay CEDEX, France.
marc.schoenauer@lri.fr

Michèle Sebag
Équipe TAO – INRIA Futurs / CNRS UMR 8623,
LRI, Bat. 490, Université Paris Sud,
91405 Orsay CEDEX, France.
michele.sebag@lri.fr

Marco Tomassini
Information Systems Institute,
Université de Lausanne,
CH-1015 Dorigny, Switzerland.
marco.tomassini@unil.ch

ABSTRACT

Abstract here...

Categories and Subject Descriptors

I.5.2 [Pattern Recognition]: Design Methodology—Classifier design and evaluation

General Terms

Ensemble Learning, Evolutionary Computations

1. INTRODUCTION

Evolutionary learning provides us population of solutions. Why not making ensemble of them given it's (almost) free?

2. RELATED WORKS

Ensemble methods [2, 12], Boosting [4], diversity in ensemble creation [1, 11, 16], evolutionary ensemble learning [13, 8, 9], learning classifier systems [5, 7].

3. METHODS PROPOSED

Two methods for of ensembles creation in evolutionary learning are proposed in this paper. The first one, dubbed *Post-Mortem Evolutionary Ensemble Learning* (EEL-PM), consists in building an ensemble from the population of the evolution's last generation. The second approach, called *Evolutionary Ensemble Learning with an Online Archive* (EEL-OA), consists in having a archive of individuals that is modified in the course of the evolution, the content of the archive of the last generation being used as the final ensemble of classifiers.

*This work has been mainly realized during a postdoctoral fellowship of Christian Gagné at the University of Lausanne.

Both methods have been designed with the same three basic design principles: 1) special fitness measures for enhancing diversity in the classification of data by the ensembles, 2) majority voting decision method for the ensembles classification decision, and 3) a greedy forward ensemble generation from candidate individuals. The differences between the methods being the moment at which the ensemble is build.

3.1 Diversity of the Classifications

The first design principle lies in the use specific fitness measure which made a varied credit assignment for each data used as fitness case. The credit assignment is inversely proportional to the good classification of the data in the ensemble or population evolved. This fitness measure corresponds to a measure of diversity in the ensemble learning sense [1, 11], that is a measure of diversity in the space of the training data. This allow the generation of diverse classifiers with varied performances over the training data, which is recognized as a fundamental issue of ensemble learning.

Let says that $\mathcal{P} = \{h_1, h_2, \dots, h_p\}$ is a population of evolved classifiers to evaluate the fitness and $\mathcal{Q} = \{h'_1, h'_2, \dots, h'_q\}$ is a reference ensemble of classifiers. Suppose now the following decision function:

$$D(a, b) = \begin{cases} 1 & \text{If } a = b \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Then, function $F(h, \mathcal{Q}, \mathcal{E})$ is used to compute the fitness of classifier h using the reference ensemble \mathcal{Q} , the associated data credit assignment function $C(\mathcal{Q}, \mathbf{x}, y)$, and the data set $\mathcal{E} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$:

$$C(\mathcal{Q}, \mathbf{x}, y) = 1 - \frac{1}{q} \sum_{i=1}^q D(h'_i(\mathbf{x}), y),$$
$$F(h, \mathcal{Q}, \mathcal{E}) = \left(\frac{1}{n} \sum_{i=1}^n (D(h(\mathbf{x}_i), y_i) C(\mathcal{Q}, \mathbf{x}_i, y_i))^\alpha \right)^{\frac{1}{\alpha}} \quad (2)$$

Parameter α adjusts the importance of different amplitude values of the weights given by function $C(\mathcal{Q}, \mathbf{x}, y)$. Typical values that can be used are $\alpha = 1$ and $\alpha = 2$.

3.2 Majority Voting

Simple majority voting is used as classification rule from the generated ensembles. Thus, a given data is assigned to the most frequently occurring classification label given by the ensemble constituents, ties being broken arbitrary. This is a simple, widely used approach which do not impose any specific restriction on the type of output (e.g. Boolean values, real numbers, probabilities) of the component classifiers by using directly the final classification label. It also is a non-linear combination of classifiers, which is needed when making ensemble of classifiers – a linear combination of linear classifiers can be simplified to a single linear classifier.

Formally, let says that $\mathcal{L} = \{l_1, l_2, \dots, l_r\}$ are all the different possible labels for the k -classes problem at hand and function $L(i) = l_i$ returning the i th label of \mathcal{L} . Then, our decision function $H(\mathcal{Q}, \mathbf{x})$ for classify data \mathbf{x} with ensemble of classifiers \mathcal{Q} is the following

$$H(\mathcal{Q}, \mathbf{x}) = L \left(\underset{i=1}{\operatorname{argmax}}^r \sum_{j=1}^q D(h'_j(\mathbf{x}), L(i)) \right). \quad (3)$$

3.3 Ensemble Creation

The final design principle is to use heuristics for generating ensembles from the individuals. Both methods use an incremental greedy algorithm to build an ensemble by adding at each iteration the individual that maximizes a given combination criteria. To start the ensemble creation, the best performing classifier (individual) on the training set is selected as member of the ensemble and removed from the candidate solutions. Then, the ensemble is build iteratively by selecting the individual in the remaining candidates that maximize the combination criteria. This incremental process continues while there are remaining individuals with an error rate on the subset of data incorrectly classified by the actual ensemble that is better than a random classification (i.e. error rate less than $1 - (1/r)$ for a r -classes classification).

This approach is similar to a simple greedy sequential forward features subset selection, but applied to the generation of an ensemble of classifiers. In many ways, the problem of features selection and ensemble creation are similar. Indeed, an ensemble of classifiers can be seen as a meta-classifier where the output of each composing classifier is a feature of the ensemble decision taking element. In other words, we have a pool of candidate classifiers (the features) and a ensemble fusing module (the classification rule) and the problem of ensemble creation is to select of subset of candidate classifiers (subset of features) that would maximize the ensemble performances. So using classical features selection methods for ensemble creation makes a lot of sense.

In order to select the optimal ensemble of the greedy algorithm, the original training data set has been split in two subsets: the fitness evaluation subset and the validation subset. The fitness evaluation subset is used to compute the fitness of the individuals and to evaluate the value of combination criteria, while the validation subset is used to evaluate the generalization capabilities of the ensemble built. Indeed, at each iterations of the ensemble construction, the error rate on the validation is computed. Once the algorithm terminated, backtracking is done by returning to the

intermediary ensemble achieving the lowest error rate on the validation set, the smallest ensemble begin selected in case of ties. The use of such methodology allows avoiding some overfitting in the construction of the ensemble of classifiers.

The combination criteria actually used for the ensemble creation is the *voting margin* between the classifiers. This margin corresponds to the minimum *voting difference* of the tested ensemble over all the data used. The voting difference for a given data corresponds to the number of votes given to the most occurring label and the second most occurring label of the ensemble classification. If the data is correctly classified, this difference value is positive, while it is negative otherwise. This approach is in line with modern learning techniques based on margin maximization. In the actual case, links can be made with an methods such the SIMBA algorithm for features selection of Gilab-Bachrach *et al.* [6].

Formally, let function $V_m(\mathcal{Q}, \mathbf{x})$ returning the index of the most occurring label, and functions $V_1(\mathcal{Q}, \mathbf{x})$ and $V_2(\mathcal{Q}, \mathbf{x})$ returning the number of votes given to respectively the most occurring label and second most occurring label obtained by ensemble classification:

$$V_m(\mathcal{Q}, \mathbf{x}) = \underset{i=1}{\operatorname{argmax}}^r \sum_{j=1}^q D(h'_j(\mathbf{x}), L(i)), \quad (4)$$

$$V_1(\mathcal{Q}, \mathbf{x}) = \sum_{i=1}^q D(h'_i(\mathbf{x}), L(V_m(\mathcal{Q}, \mathbf{x}))), \quad (5)$$

$$V_2(\mathcal{Q}, \mathbf{x}) = \max_{\substack{i=1, \dots, r \\ i \neq V_m(\mathcal{Q}, \mathbf{x})}} \sum_{j=1}^q D(h'_j(\mathbf{x}), L(i)). \quad (6)$$

From these votes counting functions, the voting margin $M_v(\mathcal{Q}, \mathcal{E})$ is computed as the minimum voting difference $M_d(\mathcal{Q}, \mathbf{x}, y)$ on data set \mathcal{E} ,

$$M_d(\mathcal{Q}, \mathbf{x}, y) = \begin{cases} V_1 - V_2 & \text{if } L(V_m(\mathcal{Q}, \mathbf{x})) = y \\ V_2 - V_1 & \text{otherwise} \end{cases}, \quad (7)$$

$$M_v(\mathcal{Q}, \mathcal{E}) = \underset{i=1}{\operatorname{argmin}}^n M_d(\mathcal{Q}, \mathbf{x}_i, y_i). \quad (8)$$

As voting margin ties between several candidate classifiers are likely when building ensembles, as secondary criteria is added in order to break ties, the *voting margin residue*:

$$\begin{aligned} V_r(\mathcal{Q}, \mathcal{E}) &= \{(\mathbf{x}_i, y_i) \in \mathcal{E} | M_d(\mathcal{Q}, \mathbf{x}_i, y_i) = M_v(\mathcal{Q}, \mathcal{E})\}, \\ M_r(\mathcal{Q}, \mathcal{E}) &= \operatorname{card}(V_r(\mathcal{Q}, \mathcal{E})). \end{aligned} \quad (9)$$

This measure $M_r(\mathcal{Q}, \mathcal{E})$ computes the number of data that have a voting difference equal to the global voting margin, which can be useful to discriminate candidate classifiers of identical voting margin value. From this criteria, the individual h^* in a population \mathcal{P} to be added to an ensemble \mathcal{Q} which maximize the voting margin (minimize the voting margin residue in case of ties) is selected by the following:

$$\begin{aligned} \mathcal{P}^* &= \left\{ h \in \mathcal{P} | M_v(\mathcal{Q} + \{h\}, \mathcal{E}) = \max_{h_i \in \mathcal{P}} M_v(\mathcal{Q} + \{h_i\}, \mathcal{E}) \right\}, \\ h^* &= \underset{h \in \mathcal{P}^*}{\operatorname{argmin}} M_r(\mathcal{Q} + \{h\}, \mathcal{E}). \end{aligned} \quad (10)$$

3.4 Post-Mortem Ensemble Creation

Figure 1: High-level presentation of the EEL-PM algorithm.

1. Initialize population \mathcal{P}_1 and generation counter $g = 1$;
2. Evaluate fitness of individuals in \mathcal{P}_g using population as reference ensemble;
3. If the evolution termination criterion is reached, jump to step 6;
4. Apply variations and selection operations to \mathcal{P}_g for generating new population \mathcal{P}_{g+1} ;
5. Increment generation counter, $g = g + 1$, and jump to step 2;
6. Construct final ensemble from final population \mathcal{P}_g using algorithm of Figure 2, and return it.

The EEL-PM method for evolutionary ensemble learning constructs the ensemble of classifiers at the end of a run, drawing candidate individuals from the last generation’s population. The evolution process consists in a classical evolution of single independent classifiers, that is to say a Pittsburgh-style evolution using the LCS terminology. It should be stressed that the precise type of classifiers evolved are not limited in any ways, they can be linear discriminants, rules systems, neural networks, genetically-evolved programs or anything else. Only a fitness measure ensuring diversity in the classification results of evolved classifiers is necessary, as well as an evolutionary algorithm that generate as final results a pool of varied classifiers of a reasonable size. Figure 1 is an high-level presentation of the EEL-PM algorithm.

In the following experiments, the fitness measure of Equation 2 is used, having as reference ensemble the whole population of actual generation (i.e. $\mathcal{Q} = \mathcal{P}$). Diversity is thus ensured in the evolving population, giving more credits to the correct classification of “difficult” data according to the global classification of the data in the population. In the experiments of this paper presented in next section, a parameter value of $\alpha = 1$ is used. This corresponds to a fixed credit assignment to each data, which credit is divided between the individuals of the population that are correctly classifying the data. Indeed, the fitness measure of a given individual corresponds to a normalized sum of the credits acquired by the evaluated classifier over all data it correctly classify in the tested set.

The algorithm of Figure 2 details the general ensemble creation method of the EEL-PM and EEL-OA approaches. In this algorithm, a simple greedy selection of classifiers to add to the ensemble is made based on the margin voting and voting margin residue criteria. The algorithm is called using the last generation’s population as pool of candidates \mathcal{S} and returns the ensemble from it. It should be noted that in our implementation, this population is filtered out before being given as pool of candidates to the ensemble creation

in order to remove duplicate individuals, that is multiple identical copies of individuals in the population, as well as poorly-performing individuals with an error rate less than random classification on the complete fitness evaluation subset. The algorithm constructs the ensemble by iterating on the population until there are no individual available with an error rate better than chance. This criteria is to ensure that candidate classifiers are at least *weak learners* [15, 3], so that under some independence hypothesis of the output generated by the weak learners, the continued aggregation of weak learners would allow an asymptotic diminution of the ensemble classifiers’ error rate. Once the iterative construction of the ensemble is finished, some backtracking is done by evaluation the classification error rate of the ensemble of classifiers generated at each iteration on the validation data set, returning as final result the ensemble with the smallest validation error rate.

3.5 Ensemble Creation with an Online Archive

The EEL-PM method previously presented allows a creation of an ensemble from the resulting last generation’s population of an evolution. If the diversity-based fitness measure is not taken into account, the ensemble creation method of EEL-PM is out of the evolutionary loop. For the second method, EEL-OA, a different approach is taken with an *online* approach where the ensemble construction is inside the evolutionary loop. This is done by an ensemble creation from the population at each generation and the use of this reference ensemble for the fitness evaluation. The ensemble created at each generation is kept into an archive independent of the evolving population. At each generation, the combination of the actual population and the ensemble archive is used to construct the new ensemble, which new ensemble replace the archive. Figure 3 presents a high-level view of the EEL-OA algorithms.

As with the EEL-PM method, the fitness measure used in EEL-OA is based on Equation 2, using as reference ensemble the actual ensemble archive of the evolution, $\mathcal{Q} = \mathcal{Q}_t$. During some preliminary experiments, it was observed that an α -value of 1 generates populations with little diversity in the classification of the data. It was thus decided to increase this parameter to $\alpha = 2$, in order to give more credit to “difficult” data (data relatively misclassified by classifiers of the population). Indeed, with this α -value, credit given is no more uniformly distributed between the data, but is rather modulated by the number of classifiers in the reference ensemble misclassifying the data. This change of the α -value appeared to increase sensibly the diversity in the classifiers’ output. This reduction of diversity is easily explainable by the fact that the archive ensemble is rebuild at each generation and is build of a limited set of classifiers that provides a good global ensemble performance. This is very different from using the population as the reference ensemble as it is done with the EEL-PM method.

As mentioned in the previous paragraph, loss of diversity was observed as the reference ensemble is now a well-performing ensemble of limited size, instead of a population of diverse individuals, as with EEL-PM method. This particularities as the impact in the construction of the ensemble, with frequent ties observed between candidate classifiers when using the voting margin and voting margin residue criteria

Figure 2: Algorithm for ensemble creation with EEL-PM and EEL-OA methods. For EEL-PM, the initial pool of candidates is the last generation’s population, $\mathcal{S} = \mathcal{P}$. For the EEL-OA method, the pool of candidates is the actual population with the previous generation’s ensemble archive, $\mathcal{S} = \mathcal{P}_{g+1} + \mathcal{Q}'_g$.

Let \mathcal{S} be the candidate pool of classifiers, \mathcal{E}_f be the fitness evaluation data set, and \mathcal{E}_v be the validation data set.

1. Initialize ensemble with the classifier of the candidate pool that has the smallest error rate, $\mathcal{Q}_1 = \{h^*\}$, and remove it from the pool of candidates, $\mathcal{S}_1 = \mathcal{S} \setminus \{h^*\}$;
2. Set $t = 2$ and **loop while** set \mathcal{S}_{t-1} is not empty:

- (a) Let pool \mathcal{S}'_{t-1} be made of candidate classifiers in \mathcal{S}_{t-1} with an error rate better than random classification (i.e. $1 - (1/r)$, where r is the number of classes) on fitness evaluation data of \mathcal{E}_f misclassified by \mathcal{Q}_{t-1} :

$$R(\mathcal{E}, h) = 1 - \frac{1}{\text{card}(\mathcal{E})} \sum_{(\mathbf{x}, y) \in \mathcal{E}} D(h(\mathbf{x}), y), \quad (11)$$

$$\mathcal{E}_m = \{(\mathbf{x}_i, y_i) \in \mathcal{E}_f | H(\mathcal{Q}_{t-1}, \mathbf{x}_i) \neq y_i\}, \quad (12)$$

$$\mathcal{S}'_{t-1} = \{h_i \in \mathcal{S}_{t-1} | R(\mathcal{E}_m, h_i) < (1 - (1/r))\}; \quad (13)$$

- (b) If \mathcal{S}'_{t-1} is an empty set then **break loop** by jumping to step 3;
 - (c) Select the individual h_{t-1}^* in \mathcal{S}'_{t-1} according to the selection criteria of the method, i.e. Equation 10 for EEL-PM and Equation 18 for EEL-OA.
 - (d) Add selected individual to ensemble, $\mathcal{Q}_t = \mathcal{Q}_{t-1} + \{h_{t-1}^*\}$, remove it from pool of candidates, $\mathcal{S}_t = \mathcal{S}_{t-1} \setminus \{h_{t-1}^*\}$, and increment iteration counter, $t = t + 1$.
3. Backtrack by selecting the ensemble of classifiers with the smallest error rate on the validation data set, selecting the ensemble with the least number of classifiers in case of ties:

$$R_q(\mathcal{E}, \mathcal{Q}) = 1 - \frac{1}{\text{card}(\mathcal{E})} \sum_{(\mathbf{x}, y) \in \mathcal{E}} D(H(\mathcal{Q}, \mathbf{x}), y), \quad (14)$$

$$\mathcal{Q} = \mathcal{Q}_u, \quad u = \underset{i=1}{\overset{t}{\text{argmin}}} R_q(\mathcal{E}_v, \mathcal{Q}_i); \quad (15)$$

4. Return the ensemble of classifiers \mathcal{Q} as result of the algorithm.

Figure 3: High-level presentation of the EEL-OA algorithm.

1. Initialize population \mathcal{P}_1 and generation counter $g = 1$;
2. Construct ensemble archive \mathcal{Q}'_1 from population \mathcal{P}_1 ;
3. Evaluate fitness of individuals in \mathcal{P}_g using ensemble archive \mathcal{Q}'_g as reference ensemble;
4. If the evolution termination criteria is reached, jump to step 8;
5. Apply variations and selection operations to \mathcal{P}_g for generating new population \mathcal{P}_{g+1} ;
6. Construct new ensemble archive \mathcal{Q}'_{g+1} from population and previous ensemble archive, $\{\mathcal{P}_{g+1} + \mathcal{Q}'_g\}$, using algorithm of Figure 2;
7. Increment counter, $g = g + 1$, and jump to step 3;
8. Return as final ensemble the actual ensemble archive \mathcal{Q}'_g ;

alone. To circumvent this, a finer criteria has been used by comparing further the maximum voting difference (i.e. voting margin) and the associated residue, by using the second maximum voting margin and residue in case of ties, and so on until it can be possible to discriminate two solutions.

Formally, this can be stated by first giving a more general formulation of the voting margin given by Equations 8 and 9:

$$\begin{aligned}
 U_v(\mathcal{Q}, \mathcal{E}, 1) &= \{1, \dots, n\}, \\
 U_v(\mathcal{Q}, \mathcal{E}, k) &= \{1, \dots, n\} \setminus \\
 &\quad \{M_v(\mathcal{Q}, \mathcal{E}, 1), \dots, M_v(\mathcal{Q}, \mathcal{E}, k-1)\}, \\
 M_v(\mathcal{Q}, \mathcal{E}, k) &= \underset{i \in U_v(\mathcal{Q}, \mathcal{E}, k)}{\operatorname{argmin}} M_d(\mathcal{Q}, \mathbf{x}_i, y_i), \quad (16)
 \end{aligned}$$

$$\begin{aligned}
 U_r(\mathcal{Q}, \mathcal{E}, k) &= \{(\mathbf{x}_i, y_i) \in \mathcal{E} \mid M_d(\mathcal{Q}, \mathbf{x}_i, y_i) = M_v(\mathcal{Q}, \mathcal{E}, k)\}, \\
 M_r(\mathcal{Q}, \mathcal{E}, k) &= \operatorname{card}(U_r(\mathcal{Q}, \mathcal{E}, k)). \quad (17)
 \end{aligned}$$

So functions $M_v(\mathcal{Q}, \mathcal{E}, k)$ and $M_r(\mathcal{Q}, \mathcal{E}, k)$ with $k = 1$ return the usual voting margin and voting margin residue, while calling them with higher value of k return higher voting differences and voting difference residues, and this in the ascending order. For selecting the best candidate classifier to be added to an ensemble, these functions are used in conjunction of a lexicographical lesser operator, defined as function $\overset{lex}{<}$:

$$\{x_1, x_2, \dots\} \overset{lex}{<} \{y_1, y_2, \dots\} : \text{iff } \exists i \ x_i < y_i \text{ and } \forall j < i \ x_j = y_j,$$

$$\begin{aligned}
 G(\mathcal{Q}, \mathcal{E}) &= \{(M_v(\mathcal{Q}, \mathcal{E}, 0), -M_r(\mathcal{Q}, \mathcal{E}, 0)), \\
 &\quad (M_v(\mathcal{Q}, \mathcal{E}, 1), -M_r(\mathcal{Q}, \mathcal{E}, 1)), \dots\}, \\
 \mathbf{h}^* &= \mathbf{h} \in \mathcal{S} \mid \forall \mathbf{h}_i \in \mathcal{S} \setminus \{\mathbf{h}\}, \\
 &\quad G(\mathcal{Q} + \{\mathbf{h}\}, \mathcal{E}) \overset{lex}{<} G(\mathcal{Q} + \{\mathbf{h}_i\}, \mathcal{E}). \quad (18)
 \end{aligned}$$

This method for getting the best candidates for addition to the ensemble extend the previous one by allowing a fine criteria for selecting classifiers in the greedy forward ensemble creation algorithm.

The ensemble creation algorithm described in Figure 2 is this time applied in EEL-OA at every generation of the evolution. The algorithm is called with a pool of candidates that consists in the actual generation's population, \mathcal{P}_g , merged with the previous generation ensemble archive, \mathcal{Q}'_{g-1} (an empty set at first generation). As with EEL-PM, this pool of candidates is filtered in order to remove duplicate and poorly-performing (poorer than random classification) individuals. The final ensemble of classifiers resulting from the EEL-OA method is the ensemble archive obtained at the last generation.

4. EVALUATION OF THE METHODS

The methods proposed for evolutionary ensemble learning are tested in a context of simple Linear Discriminant (LD) functions evolved by a real-valued Genetic Algorithm (GA). This has been selected as it is a really simple and well-known classification setup that should allow a good characterization of the ensemble learning techniques proposed. Indeed, LD classifiers have limited capabilities that disallow them discriminating some arrangements of data (e.g. XOR problem), while having several simple learning algorithms with well-understood convergence properties [3]. Of course, the evolutionary ensemble learning techniques proposed can be used in conjunction of non-linear classifiers such Support Vector Machines (SVM) or neural networks. But these method have been consciously discarded for the moment as their inherent complex nature can interfere the results obtained by the proposed approaches.

The proposed EEL-PM and EEL-OA methods are compared with single LDs classifiers trained by Least-Mean Square (LMS) [3], and ensemble technique of AdaBoost [4] with LDs trained by LMS as base classifiers. These are classical methods for respectively classical (single) classifier and ensemble learning from which good performances are expected. The ensemble learning methods are also compared to a classical evolution of single LDs classifier with the same real-valued GA algorithm.

Experiments are conducted on the six UCI data sets [14] presented in Table 1. These data sets have been split into ten stratified folds of equal size for a 10-folds cross-validation performance evaluation.

Four two-classes and two three-classes data sets are used. As LDs are limited to discriminate two-classes, three-classes problems have been encoded into two two-classes problems, where final decision is taken with a simple encoding of the two classifiers output into one of the three class labels. Suppose that $\{0, 1, 2\}$ are the three possible labels of our three-classes problem. This problem is divided into two simpler two-classes problems, says that classification sub-problem A is to discriminate data of classes $\{0, 1\}$ from data of class $\{2\}$ and classification sub-problem B is to discriminate data of class $\{0\}$ from data of classes $\{1, 2\}$. An LD is trained for each of these sub-problem, and final classification is given by the following encoding.

Table 1: UCI data sets used for the experimentations.

Data set	Size	# of features	# of classes	Application domain
bcw	683	9	2	Wisconsin's breast cancer, 65 % benign and 35 % malignant.
bld	345	6	2	BUPA liver disorders, 58 % with disorders and 42 % without disorder.
bos	508	13	3	Boston housing, 34 % with median value $v < 18.77$ K\$, 33 % with $v \in]18.77, 23.74]$, and 33 % with $v > 23.74$.
cmc	1473	9	3	Contraceptive method choice, 43 % not using contraception, 35 % using short-term contraception, and 23 % using long-term contraception.
pid	768	8	2	Pima indians diabetes, 65 % tested negative and 35 % tested positive for diabetes.
spa	4601	57	2	Junk e-mail classification, 61 % tested non-junk and 39 % tested junk.

Figure 4: Least-mean square training algorithm.

<ol style="list-style-type: none"> 1. Initialize $\mathbf{w} = 0$ and $b = 0$; 2. For epochs $t = 1 \dots t_{max}$: <ol style="list-style-type: none"> (a) For each data $(\mathbf{x}_i, y_i) \in \mathcal{E}$, in random order: $a_i = \mathbf{w}\mathbf{x}_i^T - b,$ $\Delta_i = 2\nu(t, i)(a_i - y_i),$ $\mathbf{w} = \mathbf{w} + \Delta_i\mathbf{x}_i,$ $b = b - \Delta_i;$ (b) If the variation in the RMS error $(\sqrt{\frac{1}{n} \sum_{i=1 \dots n} (a_i - y_i)^2})$ since last epoch is less than θ, then stop the algorithm.

Case	LD-A	LD-B	Class
1	-	-	0
2	-	+	1
3	+	-	X
4	+	+	2

For three classification cases, label to assign to the data is logically deduced, but for one these cases (in this example, case 3), the label to assign is ambiguous, as there is no logical class that can be decided. In these case, it can be either decided to reject the classification of the data, or to assign the data to an arbitrary label. The last solution is used here, which correspond to a very simple case of Error-Correcting Output Codes (ECOC) [10].

4.1 Least-Mean Square and Boosting

The first standard benchmark setup used to is to train LDs with a least-mean square rule given in Figure 4. For the training of simple LDs, the training rate is given by $\nu(t, i) = 1/n\sqrt{t}$, thus decreasing over the training epochs. The maximum number of epochs allowed is $t_{max} = 10000$, and the training is stopped when the variation of the RMS error is less than $\theta = 10^{-7}$. Data are normalized for all dimensions before training such that $\mathbf{x}_i \in [-1, 1], \forall_i$.

The classical AdaBoost algorithm [4] is also tested as standard benchmark for ensemble learning, using the LMS algo-

gorithm to train each composing LDs. Algorithm described in Figure 4, with the differences that maximum of $t_{max} = 1000$ epochs is allowed and the training rate is given by function $\nu(t, i) = W_k(i)/\sqrt{t}$, where $W_k(i)$ if the AdaBoost weight given to the i th data. The maximum number of boosted LDs for each AdaBoost iterations is limited to 100. The description given in [3, p. 478] was used to implement the algorithm. For the three-classes problems with both LMS and AdaBoost, the simple ECOC encoding described at the end of the previous section is used, with a training/boosting on each sub-problem before making final decision according to the encoding.

4.2 Evolution of Linear Discriminants

4.3 Evolutionary Ensemble Learning with Linear Discriminants

4.4 Experimental Results

5. CONCLUSION

6. ACKNOWLEDGMENTS

This work was supported by postdoctoral fellowships from the ERCIM-SARIT (Europe), the Swiss National Science Foundation (Switzerland), and the FQRNT (Québec) to C. Gagné.

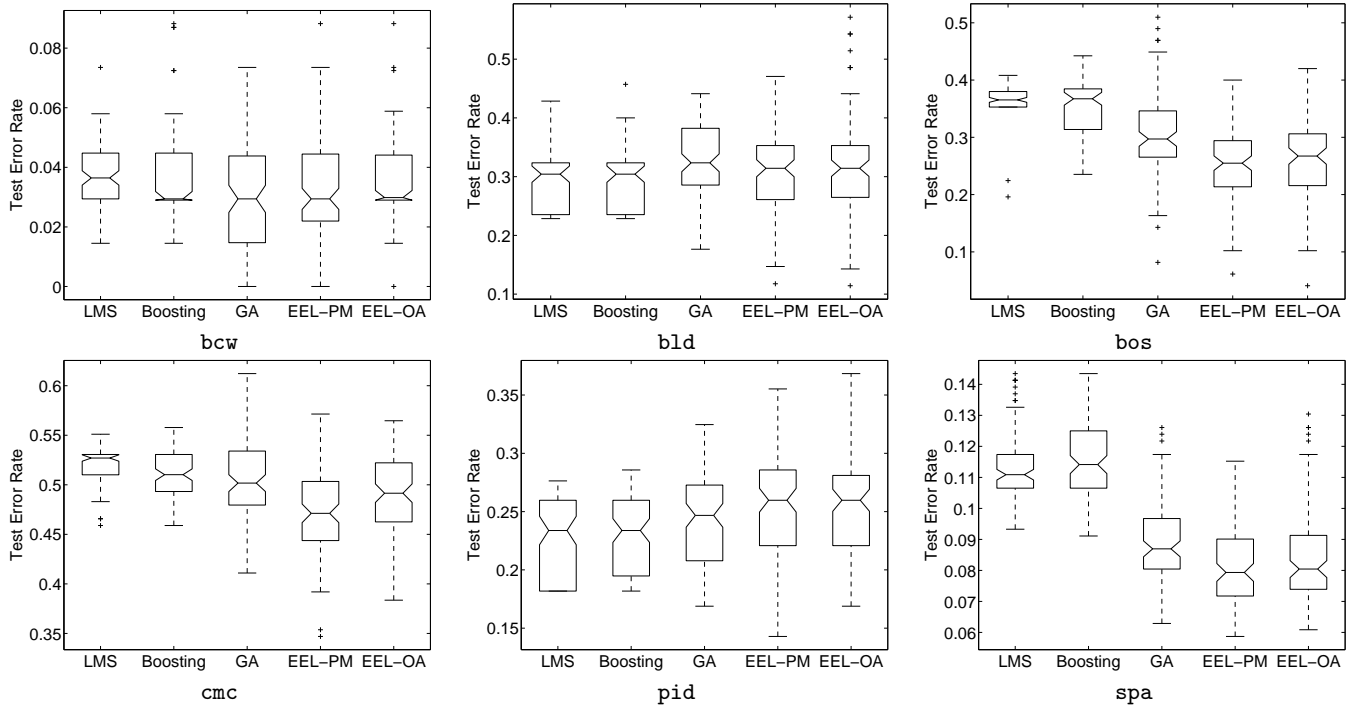
7. REFERENCES

- [1] G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods : a survey and categorisation. *Information Fusion*, 6(1):5–20, 2005.
- [2] T. Dietterich. Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.
- [3] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York (NY), USA, 2001.
- [4] Y. Freund and R. Shapire. Experiments with a new boosting algorithm. In *Proc. of the 13th. Int. Conf. on Machine Learning*, pages 148–156, 1996.
- [5] Y. Gao, J. Z. Huang, H. Rong, and D. Gu. Learning classifier system ensemble for data mining. In *Workshops of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 63–66, 2005.
- [6] R. Gilad-Bachrach, A. Navot, and N. Tishby. Margin based feature selection - theory and algorithms. In *Proc. of the 21st Int. Conf. on Machine Learning*, pages 43–50, 2004.

Table 2: Results on the UCI data sets.

	LMS	Boosting	GA	EEL-PM	EEL-OL
bcw					
Train error	3.9% (0.2%)	3.6% (0.3%)	1.8% (0.2%)	1.5% (0.5%)	1.6% (0.9%)
Validation error	–	–	–	2.9% (1.0%)	2.0% (0.9%)
Test error	4.0% (1.6%)	4.0% (2.4%)	3.2% (1.7%)	3.7% (1.9%)	3.6% (1.7%)
Ensemble size	–	3.4 (2.9)	–	16.7 (24.9)	9.7 (16.0)
bld					
Train error	29.8% (0.9%)	29.4% (0.9%)	25.4% (1.2%)	20.0% (4.0%)	18.3% (8.0%)
Validation error	–	–	–	27.3% (3.4%)	22.8% (3.4%)
Test error	30.4% (6.6%)	30.0% (5.9%)	32.7% (6.6%)	31.3% (7.4%)	31.5% (8.4%)
Ensemble size	–	1.8 (1.3)	–	23.2 (19.3)	47.0 (71.1)
bos					
Train error	32.2% (1.3%)	31.5% (1.4%)	23.4% (4.1%)	16.2% (2.7%)	20.8% (3.0%)
Validation error	–	–	–	22.9% (3.7%)	19.3% (3.1%)
Test error	34.0% (6.7%)	35.7% (5.4%)	30.7% (7.5%)	25.1% (5.9%)	25.9% (6.6%)
Ensemble size	–	6.7 (4.9)	–	105.4 (68.5)	53.4 (61.1)
cmc					
Train error	51.6% (0.4%)	50.9% (0.4%)	45.7% (1.4%)	41.2% (1.8%)	44.4% (2.2%)
Validation error	–	–	–	45.2% (2.2%)	44.1% (1.9%)
Test error	51.8% (2.5%)	51.2% (2.7%)	50.4% (3.9%)	47.2% (4.2%)	49.1% (3.9%)
Ensemble size	–	9.8 (12.4)	–	44.4 (21.6)	75.9 (79.2)
pid					
Train error	22.0% (0.6%)	21.9% (0.6%)	20.2% (0.7%)	19.4% (1.7%)	21.8% (1.9%)
Validation error	–	–	–	23.1% (2.6%)	20.7% (2.1%)
Test error	22.8% (3.5%)	23.1% (3.6%)	24.2% (3.9%)	25.4% (4.3%)	25.5% (4.5%)
Ensemble size	–	3.1 (2.2)	–	11.8 (7.6)	26.0 (38.9)
spa					
Train error	11.1% (0.4%)	11.4% (0.4%)	7.9% (0.5%)	6.9% (0.5%)	7.1% (0.6%)
Validation error	–	–	–	7.8% (0.7%)	7.5% (0.6%)
Test error	11.3% (1.2%)	11.6% (1.3%)	9.0% (1.3%)	8.1% (1.3%)	8.4% (1.6%)
Ensemble size	–	8.3 (18.6)	–	10.5 (5.4)	41.1 (87.4)

Figure 5: One-way analysis of variance (ANOVA) box plots of the best-of-run solutions test set error rates.



- [7] J. Holmes, P. Lanzi, W. Stolzmann, and S. Wilson. Learning classifier systems: New models, successful applications. *Information Processing Letters*, 82(1):23–30, 2002.
- [8] H. Iba. Bagging, boosting, and bloating in genetic programming. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1053–1060, 1999.
- [9] K. Jong, E. Marchiori, and M. Sebag. Ensemble learning with evolutionary computation: Application to feature ranking. In *Parallel Problem Solving from Nature VIII*, pages 1133–1142, 2004.
- [10] L. Kuncheva. Using diversity measures for generating error-correcting output codes in classifier ensembles. *Pattern Recognition Letters*, 26(1):83–90, 2005.
- [11] L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.
- [12] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.
- [13] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.
- [14] D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases. <http://www.ics.uci.edu/~mllearn/MLRepository.html>, 1998.
- [15] R. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
- [16] E. K. Tang, P. N. Suganthan, and X. Yao. An analysis of diversity measures. *Machine Learning*, 65:247–271, 2006.