Journal of Intelligent & Fuzzy Systems 18 (2007) 445–454
IOS Press

Evolutionary design of multiclass support vector machines

Ana C. Lorena* and André C. P. L. F. Carvalho
Departamento de Ciências de Computação, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, campus de São Carlos, Caixa Postal 668, CEP 13560–970, São Carlos, SP, Brazil

Abstract. Support Vector Machines constitute a Machine Learning technique originally designed for the solution of 2-class problems. For multiclass applications, several strategies divide the original problem into a set of binary subtasks, whose results are combined. This work introduces the use of Genetic Algorithms to determine binary decompositions of multiclass problems. Experimental results on benchmark and Bioinformatics multiclass datasets indicate the potential of the proposed approach, which is able to produce good multiclass solutions with the use of simple decompositions.

Keywords: Support vector machines, multiclass classification

1. Introduction

Multiclass classification by Machine Learning (ML) techniques consists of inducing a function $f(\mathbf{x})$ from a dataset composed of pairs $(\mathbf{x}_i, y_i)$, where $y_i \in \{1, \ldots, k\}$ and $k > 2$. Some popular learning techniques are originally binary, being able to carry out classifications only when $k = 2$. Among these techniques, one can mention Support Vector Machines (SVMs) [11].

In order to generalize SVMs to multiclass problems, several strategies have been proposed. A standard strategy is the one-against-all (OAA) approach, where k binary classifiers are induced. Each classifier is responsible to separate a class i from the remaining classes [11]. Other common extension is known as one-against-one (OAO). In this approach, given a problem with k classes, $k(k-1)/2$ classifiers are induced. Each induced classifier distinguishes a pair of classes (i, j) [17]. Dietterich and Bariki [16] suggested the use of error-correcting output codes (ECOC) to represent each class in a multiclass problem. In this approach, binary classifiers are trained to learn the binary values in the codes. In [7], Allwein et al. proposed a framework that unifies the previous decomposition strategies as code based approaches.

In practice, none of the previously mentioned techniques can be considered the best for every multiclass problem. Although most of the current works adopt the OAA method, there are scenarios where other decompositions of the multiclass problem can provide better results (see, for example, [7,16]). Based on this observation, the present work introduces the use of Genetic Algorithms (GAs) [10], a search technique based on principles of genetics and natural evolution, to determine binary classifiers combinations according to their joint performance in the multiclass solution. The introduced strategy is evaluated using benchmark and Bioinformatics datasets. Initial experiments with a previous version of this algorithm in a protein structural classification dataset have already shown promising results [1], indicating that basic decomposition solutions of multiclass problems could be improved by considering their performance on the problem solution.

This paper is structured as follows: Section 2 presents the materials and methods employed in this work. Section 3 describes the experimental results. Section 4 discusses the results obtained and Section 5 concludes this paper.

*Corresponding author. Tel.: +55 16 3373 9646; Fax: +55 16 3371 2238; E-mail: aclorena@icmc.usp.br.

2. Materials and methods

This section introduces the SVMs and describes some multiclass strategies reported in the literature. It also presents the algorithm that implement the strategy proposed in this work and the datasets used in the experiments.

2.1. Support vector machines

Support Vector Machines (SVMs) constitute a learning technique based on the Statistical Learning Theory [11]. Given a dataset with n instances $(\mathbf{x}_i, y_i)$, where each $\mathbf{x}_i \in \Re^m$ is an instance and $y_i \in \{-1, +1\}$ corresponds to $\mathbf{x}_i$'s label, this technique looks for an hyperplane $(\mathbf{w} \cdot \mathbf{x} + b = 0)$ able to separate data with a maximal margin. In order to perform this task, it solves the following optimization problem:

$$\textbf{Minimize} : ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i$$

$$\textbf{Restrictedto} : \begin{cases} \xi_i \geqslant 0 \\ y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \geqslant 1 - \xi_i \end{cases}$$

where C is a constant that imposes a tradeoff between training error and generalization and each ξ_i is a slack variable. These variables relax the restrictions imposed to the optimization problem, allowing a set of patterns to be within the margins and the presence of some training errors. The decision frontier obtained is given by Eq. (1).

$$f\left(\mathbf{x}\right) = \sum_{i=1}^{n} y_i \alpha_i \mathbf{x_i} \cdot \mathbf{x} + b \tag{1}$$

where the constants α_i are named Lagrange multipliers and are determined in the optimization process.

When a non-linear separation of the dataset is needed, a mapping procedure is applied to its examples. In the new high-dimensional space, also named feature space, the dataset can be separated by a linear SVM with a low training error. This mapping process is performed with the use of Kernel functions, which compute dot products between any pair of patterns in the feature space. Therefore, the only modification necessary to deal with non-linearity in the dataset is to substitute any dot product among patterns by a Kernel function. The Kernel function used in this work was the Gaussian function, illustrated in Eq. (2).

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\sigma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) \tag{2}$$

2.2. Multiclass strategies

SVMs were originally formulated for the solution of problems with two classes ($+1$ and -1, respectively). In order to extend them to multiclass problems, several strategies have been proposed.

The most straightforward strategy is the one-against-all (OAA) decomposition. Given a problem with k classes, k binary classifiers are generated. Each classifier is responsible to distinguish a class i from the remaining classes. The final prediction is usually given by the classifier with the highest output value.

Another standard strategy, named one-against-one (OAO), consists of building $k(k-1)/2$ predictors, each differentiating a pair of classes i and j, where $i \neq j$. To combine the output produced by these classifiers, a majority voting scheme can be applied [17].

Dietterich and Bariki [16] proposed the use of a distributed output code to represent the k classes associated with a multiclass problem. For such, a codeword of length l is assigned to each class. Commonly, the codewords have more bits than needed in order to represent each class uniquely. The additional bits can be used to correct eventual classification errors. For this reason, this method is named error-correcting output coding (ECOC). The generated codes are stored in a matrix $\mathbf{M} \in \{-1, +1\}^{k \times l}$. The rows of this matrix represent the codewords of each class and the columns correspond to the desired outputs of the l binary classifiers $(f_1(\mathbf{x}), \ldots, f_l(\mathbf{x}))$ induced. A new pattern $\mathbf{x}$ can be classified by evaluating the predictions of the l classifiers, which generate a vector $\mathbf{f}(\mathbf{x})$ of length l. This vector is then compared with the rows of $\mathbf{M}$. The example is assigned to the class with the closest row according to a given measure, like the Hamming distance. This process is also known as decoding.

In [7], Allwein et al. presented a framework that unifies the previous decomposition strategies. The OAA and OAO techniques were also reduced to code based methods. For such, a value from the set $\{-1, 0, +1\}$ is assigned to each element of the code matrix $\mathbf{M}$. In the OAA case, $\mathbf{M}$ has dimension kxk, with the diagonal elements equal to $+1$. The remaining elements are equal to -1. In the OAO decomposition, $\mathbf{M}$ has dimension kx$k(k-1)/2$ and each column corresponds to a binary classifier for a pair of classes (i, j). In each column representing a pair (i, j), the value of the elements corresponding to lines i and j are defined as $+1$ and -1, respectively. All other elements receive the value 0, indicating that patterns from the other class-

es do not participate in the induction of this particular binary classifier.

Binary classifiers are trained to learn the labels presented in the columns of **M**. The prediction of a new pattern's class involves a decoding step. The decoding occurs through the comparison of the joint predictions of the binary classifiers with the codewords of **M**. For this comparison, Allwein et al. [7] proposed the use of a margin based loss measure, that considers the pattern's margins obtained by each individual SVM. This formulation is motivated by the fact that the Hamming distance ignores the cost function used in the SVMs training, as well as confidences attached to the predictions made by these classifiers, which can be quantified by the margins associated to each pattern.

There are also a few works that modify the SVM internal procedures to perform the multiclass classification task directly. Since the focus on this work is on decomposition strategies, the direct strategies will not be discussed in this paper.

2.3. Evolutionary design of multiclass SVMs

The determination of an adequate combination of binary predictors for a given multiclass solution is a current research issue in ML. This problem can be formulated as a search for codes to represent each class. Another issue to be addressed is the size of these codewords (the number of binary predictors in the multiclass solution). The search of code combinations in conjunction with the number of binary classifiers to compose the multiclass solution constitutes a combinatorial problem. This section proposes the use of Genetic Algorithms (GAs) [10] to solve this problem.

GAs are search and optimization techniques based on genetics and natural selection. They deal with optimization problems by investigating populations of possible solutions or individuals. The optimization process usually takes several generations. At each generation, the fittest individuals from the current population are selected for the application of genetic operators. These operators produce a new population of individuals. The most common genetic operators are elitism, which sends the best individuals to the next generation, cross-over, which combines features from pairs of individuals, and mutation, which changes features of selected individuals. The principle of using various individuals representing possible solutions, allied to the processes of cross-over and mutation, allows a large search space to be covered in multiple directions, making GAs a global search technique.

Next, the authors show how GAs were applied to the multiclass decomposition problem.

2.3.1. Individuals representation

In the representation adopted, each individual corresponds to a possible code matrix $\mathbf{M} \in \{-1, 0, +1\}^{k \times l}$. The number of columns in the matrix is defined by the size of the individual. Individuals of different sizes were allowed. Two individuals that represent possible solutions to a problem with four classes are illustrated in Fig. 1.

The initial population was composed by individuals with random values and sizes. A consistency test was applied to these individuals, so that each binary classifier had positive and negative class labels.

2.3.2. Fitness function

The individuals were evaluated by their predictive power in the multiclass solution. The goal was to minimize a multiclass error rate measured by using validation sets. Unknown classifications were also computed as errors. An unknown classification occurs when two or more rows of the code matrix simultaneously have the minimum distance to the predictions of the binary classifiers. The decoding of the binary classifiers outputs was performed with the use of the margin based loss measure [7].

If more than one solution presented the same validation error rates, preference was given to smaller individuals, which represent simpler models. This criterion follows the Occam's razor, which states that, among several correct hypothesis, the simpler should be chosen [19]. The minimization of the number of binary classifiers can be considered, thus, a second objective of the GA.

The presence of equal classifiers in the GA solutions was also avoided. Repeated and complementary columns in a code matrix represent multiple uses of identical binary classifiers in the same decomposition. For the incorporation of this restriction to the GA matrices, the authors employed a method proposed in [9]. All individuals that violate the restriction (infeasible) must have worst fitness than those that respect it (feasible). The fitness of the infeasible individuals was then given by the sum of the proportion of equal classifiers and the maximum validation error of the feasible individuals. For the feasible individuals, the fitness was given only by their validation error.

2.3.3. Elitism

The elitism operator selected, at each generation, a fraction of the best individuals of the current population.

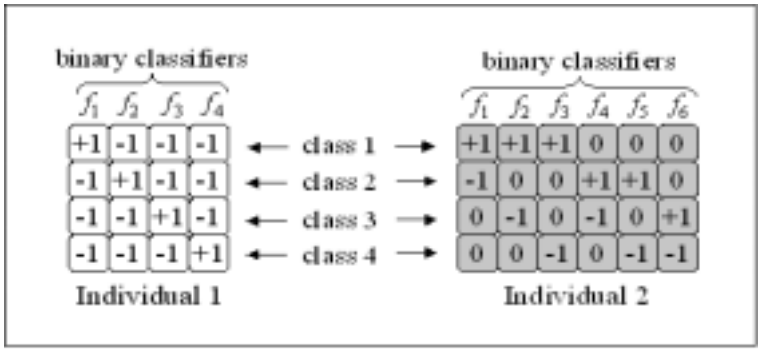

Fig. 1. Examples of two individuals for a problem with four classes.

2.3.4. Crossover

The following crossover operators were employed:

- Exchange of individuals columns. This operator exchanges binary classifiers between two individuals, motivated by the fact that a binary predictor can be more efficient in an alternative combination. It is illustrated in Fig. 2(a).
- Exchange of groups of columns between individuals. In this case, given two individuals, their offsprings are produced permuting all columns from the parents from randomly selected positions. Its application enables the production of individuals with new sizes. This operator is illustrated in Fig. 2(b).

The selection of parents for crossover was performed with tournament selection [10]. To select one individual, two solutions are randomly chosen from the population. If they are both infeasible, the individual with the smallest number of violations is chosen. If only one individual is feasible, it is automatically chosen. Otherwise, a number between 0 and 1 is generated. If this number is lower than 0.75, the fittest individual is chosen. Otherwise, the other individual is selected.

2.3.5. Mutation

Four types of mutation operators were investigated:

- Change the value of a matrix element, randomly selected. This operator is illustrated in Fig. 3(a).
- All the values in a matrix column can be modified. Figure 3(b) illustrates this procedure.
- Generate a new column (binary classifier), with random values for its genes. Figure 3(c) illustrates this case.
- Remove a random column from an individual, eliminating a binary classifier, as illustrated in Fig. 3(d).

Table 1
Datasets main characteristics

Dat	♯Tr	♯Test	♯Cl	♯Attrib num/nom	mea/min/max ex/class
car	1728	–	4	0/6	432/65/1210
lun	203	–	5	12600/0	40.6/6/139
seg	2310	–	7	19/0	330/330/330
fun	2355	–	9	40/0	261.7/34/814
pen	7494	3498	10	16/0	749.4/719/780

It should be noticed that the application of the mutation operators may generate columns (binary classifiers) without positive or negative labels. A consistency phase was employed to correct these cases, defining new positive/negative labels.

As there are multiple types of crossover and mutation operators, it is necessary to define which operator to apply. For such, a criterion used in [13] was applied. Each possible operator was selected probabilistically according to its performance in previous generations. By this scheme, operators that produced better solutions in previous generations have higher chances of being applied again and the importance of each operator is adapted by the GA.

2.4. Datasets

The main characteristics of the datasets employed in this work are presented on Table 1. This table presents, for each dataset, the number of training data (♯Tr), the number of test data (♯Test), the number of classes (♯Cl), the number of numeric and nominal attributes (♯Attrib num/nom) and the mean, minimum and maximum number of examples per class (mea/min/max ex/class).

The datasets car, segment (seg) and pen-digits (pen) were obtained from the UCI repository [5], which provides benchmarks for ML algorithms. The other datasets are related to Bioinformatics applications.

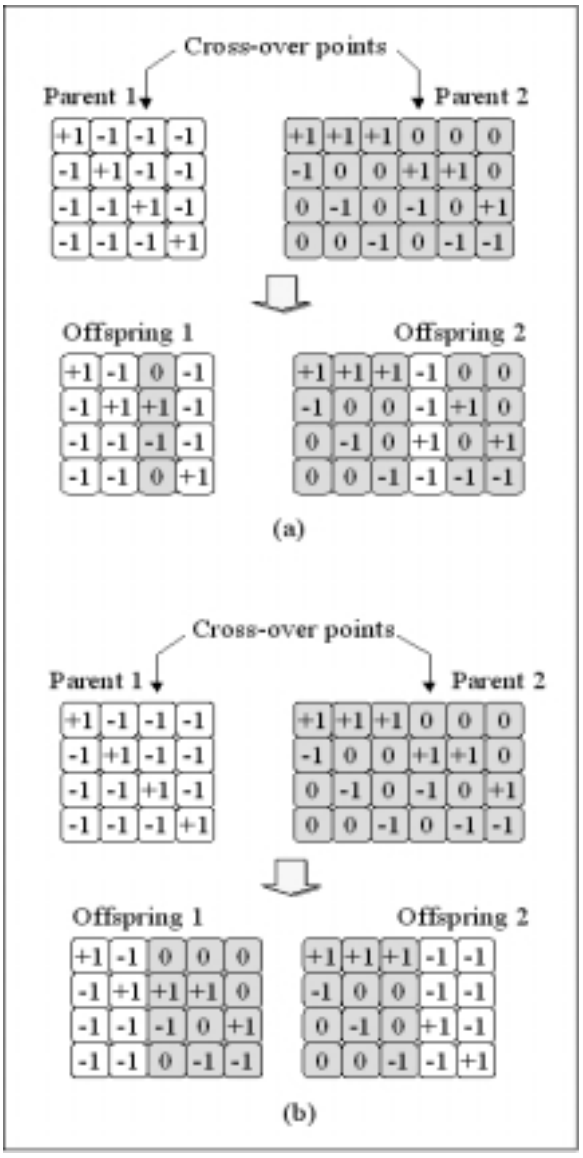

Fig. 2. Types of crossover operators employed.

The data in the car dataset represent cars classified into four categories. The segment dataset has patterns divided into seven classes of outdoor images. In the pen-digits dataset, the data are handwritten digits.

The lung[1] (lun) dataset has gene expression data for the classification of lung tissues. There are five classes. One of them corresponds to normal tissues and the remaining constitute four types of lung tumors. An attribute selection step was applied to this dataset, to reduce its number of attributes and thus simplify the posterior classifiers induction process. As a result, the 500 most relevant genes for discriminating the tissue types were selected. The method employed for attribute selection is described in [15] and considers the gene expression ratios between groups and within groups.

The fungi[2] dataset (fun) has proteins classified into nine possible locations. Protein localization is an important problem of Bioinformatics, since many cellular functions are carried out in specific compartments of the cell. The proteins contained in this dataset were pre-processed for the extraction of numerical attributes according to a method proposed in [12]. First, each protein was divided into two halves. A composition vector, which presents the proportion of the 20 amino acids in the sequences, was then built for each half. The two vectors obtained were merged to form the final representation of the protein, with 40 attributes. This representation allows the consideration of the sequence order of the amino acids in the proteins.

3. Experiments

This section presents the experiments conducted in order to evaluate the algorithm proposed in this work.

3.1. Data Pre-processing

All datasets, except for pen-digits, were divided according to the r-fold cross validation methodology. Each one of these datasets was divided into r disjoint subsets of approximately equal size. In each train/test round, $r-1$ subsets were used for training and the remaining was left for test. This makes a total of r pairs of training and test sets. In order to ensure that all folds had elements from every class, a stratified approach was adopted, in which each partition presented the same class distribution found in the original dataset. The number of folds was defined according to each dataset class distribution, in order to ensure the presence of elements of every class in the test sets. For the car, segment and fungi datasets, the value adopted was $r = 10$. For the lung dataset, $r = 3$. For the pen-digits dataset, the original train/test split present in the UCI repository was used.

To evaluate the individuals in the GA, each training set was further divided with the holdout method into

[1] This dataset was obtained from http://www.gems-system.org.

[2] This dataset was obtained from http://www.cs.ualberta.ca/~bioinfo/PA/Subcellular.

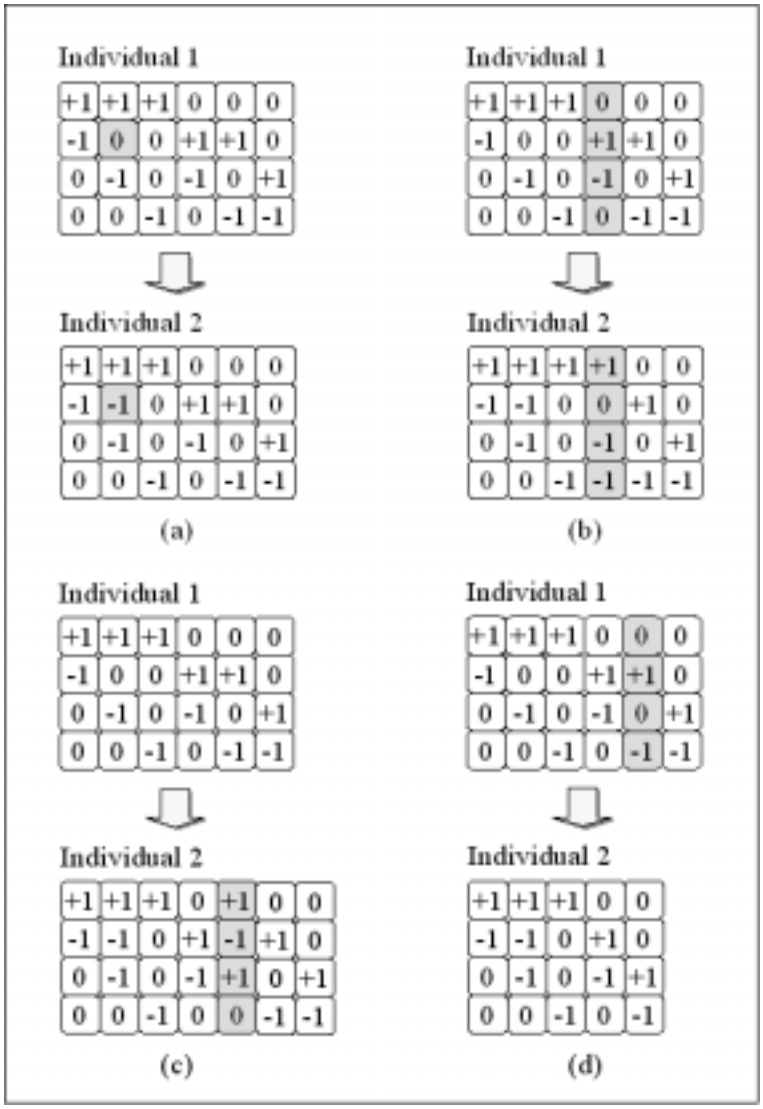

Fig. 3. Types of mutation operators employed.

two subsets. The first subset, with 70% of the data, was used for training the classifiers in the GA, while the 30% remaining defined a validation set used to evaluate the error rates of the code matrices in the multiclass solution.

For the SVMs, all data attributes must be in numerical format. Therefore, the nominal attributes in the car dataset were codified accordingly. A canonical encoding was adopted, where there is a bit for each possible attribute value.

Finally, some datasets were normalized, preventing the domination of attributes in higher numerical ranges over those in lower levels. The training sets' attributes were normalized to have zero mean and unit variance. Their corresponding test and validation sets were preprocessed according to the normalization factors extracted from the training sets. This normalization step was not applied to the fungi dataset, because it would oversight the amino acid order information imposed by the protein encoding employed. The car dataset, whose attributes were codified in canonical format, was also not normalized.

3.2. Experiments configuration

The OAA strategy is the most frequently used in the generalization of SVMs to multiclass problems. Therefore, the GA was employed to find code matrices with

a maximum number of classifiers equal to k, which corresponds to the number of binary predictors in the OAA strategy. The OAA matrix was also supplied as an initial solution to the GA. The objective of this study can be summarized as to determine decompositions adequate to each multiclass problem that are simpler than OAA, requiring less binary classifiers and with accuracy superior or similar to OAA. The number of binary predictors allowed for the GA matrices was constrained to the interval $[\lceil \log_2(k) \rceil, k]$, where $\lceil \log_2(k) \rceil$ corresponds to the minimum number of binary classifiers necessary to distinguish k classes.

The GAs employed a crossover rate equal to 0.8, a mutation rate equal to 0.05, a population size equal to $5 * k$ and a maximum number of cycles equal to $50 * k$, where k denotes the number of classes of the multiclass problem. The population size and the maximum number of cycles were defined according to the characteristics of each dataset. For problems with more classes, they were larger, since the search space grows with the number of classes.

As the GAs experimental results depend on the initial solutions provided, they were executed 30 times for each train/test partition of a dataset. The results presented correspond to the average of the results obtained in the 30 runs.

Besides the OAA and GA matrices, other two types of code matrices were generated: the OAO and the ECOC. The ECOCs were built with the exhaustive method described in [16]. Given a problem with k classes, k codewords of length $2^{k-1}-1$ were constructed. The codeword of the first class is composed of only $+1$ labels. For the other classes i, where $i > 1$, it is composed of alternate runs of 2^{k-i} negative (-1) and positive $(+1)$ labels. The application of this procedure corresponds to produce classifiers for each possible binary partition of the k classes.

To verify the behavior of the GA-based strategy and its ability to find adequate solutions, random matrices with numbers of classifiers in the interval $[\lceil \log_2(k) \rceil, k]$ were also generated. As in the GA, a total of thirty matrices of this type were produced for each train/test partition of each dataset. The goal was to verify if the GA solutions were more adequate than those obtained through a random search. In each matrix obtainment, a procedure similar to the one adopted in [7] for the creation of random matrices was followed. A total of 10000 random matrices were built, with elements in $\{-1, 0, +1\}$ and varying sizes. The matrix with the highest minimum Hamming distance between lines and without identical columns was chosen. Although the generation of each one of the thirty matrices was not completely random, for simplicity, they will be denoted as random in the subsequent analysis.

All simulations with binaries SVMs were performed with the LibSVM library [6], using a C value of 100 and a Gaussian kernel function with standard deviation σ of 0.01. Although the best values of the SVM parameters may differ for each multiclass strategy, they were kept the same to allow a fair evaluation of the differences between the investigated techniques.

3.3. Results and analysis

The accuracy rates of the previously mentioned strategies are shown on Table 2. The mean accuracy rates of all solutions are illustrated for the GAs and the random matrices. Standard deviation rates are reported in parenthesis.

It is important to observe the stability of the solutions produced by GA. The accuracy rates of the GA matrices were clearly more stable than the random matrices, showing lower standard deviation rates. For datasets divided with cross validation, the GA standard deviation rates were similar to the OAA, OAO and ECOC rates. Most of the variations observed can then be attributed to the use of the cross validation procedure, that is, that the accuracy rates have been calculated from distinct partitions of the datasets.

To verify more clearly the differences among the results obtained, the accuracy rates of all strategies in each dataset were compared using a statistical test for multiple comparisons [2]. Among the several random and GA solutions obtained in a given dataset partition, the one with accuracy rate closer to the mean accuracy of all solutions was chosen to represent the tests.

For all datasets, the random matrices showed accuracy rates statistically different from and inferior to the other strategies. The GA search was therefore better than a random process. In the car dataset, OAA and ECOC accuracies were statistically different from those produced by GA and OAO, which were higher. In the lung dataset, the GA mean accuracy was statistically different from the OAO accuracy, which was inferior. In the fungi dataset, the OAA accuracy was statistically different from the OAO, ECOC and GA accuracies, which were higher. In the segment and pen-digits datasets, the results of OAA, OAO, ECOC and GA were all statistically similar.

From the statistical analysis and from the results on Table 2, the GA solutions either showed the highest accuracies or were among the strategies with highest

Table 2
Datasets results

	Accuracies					♯ Bin. Classifiers					Time
Dat.	OAA	OAO	ECOC	Rand.	GA	OAA	OAO	ECOC	Rand.	GA	GA
car	94.0 (2.6)	98.1 (1.3)	93.8 (2.8)	85.1 (20.1)	97.8 (1.4)	4	6	7	3.2 (0.7)	3.6 (0.5)	9.0 (0.2)
lun	82.8 (3.0)	76.4 (1.3)	82.8 (3.0)	65.9 (22.1)	85.5 (4.8)	5	10	15	4.1 (0.8)	3.6 (0.7)	25.2 (0.2)
seg	95.4 (1.5)	96.6 (1.0)	94.9 (1.2)	67.8 (16.1)	96.2 (1.1)	7	21	63	4.9 (1.4)	6.5 (0.7)	68.7 (3.5)
fun	56.1 (2.1)	60.4 (2.4)	60.2 (3.4)	39.8 (11.7)	62.0 (2.6)	9	36	255	6.9 (1.9)	7.7 (1.2)	402.5 (32.4)
pen	97.9	97.3	97.8	69.5 (20.8)	97.4 (0.3)	10	45	511	6.4 (2.2)	9.8 (0.6)	573.0 (24.2)

accuracies in all datasets. The ECOC and OAO accuracies were among the best in four datasets, while OAA showed good accuracies in three datasets. The GA was consequently able to either maintain or improve the accuracies of OAA. Thus, it accomplished the first goal of the experiments. Another goal of the experiments with the GA was to minimize the number of binary classifiers in the decompositions. Table 2 also presents this information, showing the number of binary classifiers of all matrices generated.

The GA was also able to reduce the number of binary classifiers when compared to OAA in all cases. However, in the pen-digits dataset, the mean number of classifiers of the GA solutions was close to that of the OAA strategy. An exam of the GA solutions in this case showed that the obtained matrices were similar to the OAA matrix. This fact agrees with a recent affirmation that the OAA decomposition is adequate to multiclass solutions with SVMs [14]. Moreover, it indicates that a reduction in the number of binary classifiers may not be appropriate for this particular dataset.

Among the matrices produced by OAA, OAO and ECOC, the lowest number of binary classifiers was required by OAA, with k predictors for k classes. Next comes the OAO matrix, with $k(k-1)/2$ predictors. ECOC had the highest number, $2^{k-1}-1$. The random matrices, as expected, showed variations in the number of binary classifiers.

It is also interesting to notice that, in the fungi and the lung datasets, where the GA matrices presented the best accuracies, there were large reductions in the number of binary classifiers. This is an interesting result, since it illustrates that, through the use of a search mechanism that adapts code matrices to the multiclass problem solution, it is possible to obtain higher accuracy rates with simpler decompositions.

The time required by the GA, in seconds, to obtain its solutions for each dataset is also illustrated in Table 2. They varied according to the problem size, measured by the number of data instances and classes. To speed up the GA, two lists were implemented. The first list stored classifiers already trained and evaluated on the validation set. The second list kept entire code matrices previously evaluated.

As the genetic operators were probabilistically applied according to their performance, the genetic operators with the highest contribution to the achievement of the final solutions were also recorded. The crossover operator that exchanges pairs of columns was the most frequently used. In the mutation case, there was a prominance of the first type, that randomly changes the value of an element in a code matrix. They can be then considered the most adequate operators for this search problem for the investigated datasets.

4. Discussion

From the results and analysis of the previous section, the authors observed that the GA was able to obtain matrices with good accuracy rates and less binary classifiers than the OAA, OAO and ECOC strategies.

The datasets with the best results for the GA strategy were lung and fungi, which represent real Bioinformatics applications. The GA produced code matrices whose mean accuracies were the highest among all tested strategies.

The reported accuracy rates on the lung dataset were inferior to those previously published in [4], although a fair comparison is not possible, since they performed a model selection to adjust the SVMs parameters. However, as in [4], the OAA strategy was slightly better than the OAO. The GA was able to further improve the accuracy rate achieved, which was in turn statistically different from the OAO one.

The results for the fungi dataset were also inferior to those reported in a previous work using the same dataset [18]. However, the superior performance obtained in this other work can be attributed to the use of a more complex protein representation than the one employed in this paper.

Although the overall performance of the different algorithms were similar in several cases, there were differences in the performance obtained for each class.

Each strategy tended to favor one or more classes. This knowledge can be used to choose a particular technique in domains where the classes have different relevance. The complete results are not presented here due to space limitations. The GAs had the tendency to increase the accuracy rates for classes with less data.

Clearly, the adaptation of code matrices through GA has a computational cost, that increases for problems with more classes. This deficiency must be taken into account, since the performance of the obtained matrices was similar to the ones achieved by other strategies. However, GAs can benefit from the increasing use of parallel technologies in order to reduce their processing cost.

The objectives of minimizing the validation error rates with a matrix of minimum size were also formulated and solved by means of a multi-objective GA. However, the results were disappointing. Although the code matrices obtained had less binary classifiers, they were not able to maintain the accuracy rates of other decomposition strategies found in the literature.

5. Conclusion

This work presented a new approach for multiclass SVMs generation based on GAs. For such, GAs were applied to determine combinations of binary classifiers in a multiclass solution. This new technique allows the construction of multiclass solutions more related to the problem's characteristics. The matrices obtained in benchmark and Bioinformatics applications were able to either maintain or improve the accuracy of common code matrix strategies through the use of simpler decompositions, that required less binary classifiers.

Future experiments will consider the modification of the SVMs and GAs parameters, since this procedure can improve the results obtained in the experiments carried out.

Other modification being considered include using leave-one-out bounds from the SVM literature (like the ones in [3]) in the GA's fitness evaluation. Works involving the use of GAs in conjunction with SVMs have proved that these bounds can be more effective to evaluate the SVMs fitness than a cross validation methodology (ex.: [8]).

It should be noticed that the multiclass approaches used in this paper are general and can be applied to other multiclass problems. They can also employ other base learning techniques, requiring only a change in the decoding function used in the classifiers integration.

Acknowledgement

To FAPESP and CNPq for their support.

References

[1] A.C. Lorena and A.C.P.L.F. Carvalho, *A GA/SVM Approach for Multiclass Classification Applied to Protein Structural Class Prediction*, Proceedings of the VIII Brazilian Symposium on Neural Networks, CD, 2004.

[2] A. Feelders and W. Verkooijen, *On the Statistical Comparison of Inductive Learning Methods*, Learning from data: Statistical intelligence V, Springer Verlag, 1996, 272–279.

[3] A. Passerini, M. Pontil and P. Frasconi, New results on error correcting output codes of Kernel machines, *IEEE Transactions on Neural Networks* **15** (2004), 45–54.

[4] A. Statnikov, C.F. Aliferis, I. Tsamardinos, D. Hardin and S. Levy, A comprehensive evaluation of multicategory methods for microarray gene expression cancer diagnosis, *Bioinformatics* **21**(5) (2005), 631–643.

[5] C.L. Blake and C.J. Merz, UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[6] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[7] E.L. Allwein, R.E. Shapire and Y. Singer, *Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers*, Proceedings of the 17th International Conference on Machine Learning, 2000, 9–16.

[8] H. Fröhlich, O. Chapelle and B. Schölkopf, *Feature Selection for Support Vector Machines by Means of Genetic Algorithms*, Proceedings of the 15th IEEE International Confenrence on Tools with AI, 2003, 142–148.

[9] K. Deb, An efficient constraint handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering* **186** (2000), 311–338.

[10] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1999.

[11] N. Cristianini and J.S. Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, 2000

[12] Q. Cui, T. Jiang, B. Liu and S. Ma, Esub8: a novel tool to predict protein subcellular localizations in eukaryotic organisms, *BMC Bioinformatics* **5**(1) (2004), 1–7.

[13] R. Martí, M. Laguna and V. Campos, *Scatter Search vs. Genetic Algorithms: An Experimental Evaluation with Permutation Problems*, In Metaheuristic optimization via adaptive memory and evolution: Tabu Search and Scatter Search, Kluwer Academic Publishers, 2005, 263–282.

[14] R. Rifkin and A. Klautau, In defense of One-Vs-All classification, *J of Machine Learning Res* **5** (2004), 1533–7928.

[15] S. Dudoit, J. Fridlyand and T.P. Speed, Comparison of discrimination methods for the classification of tumors using gene expression data, Technical Report 576, Department of Statistics, University of Berkeley, 2000.

[16] T.G. Dietterich and G. Bariki, Solving multiclass learning problems via error-correcting output codes, *J of Artificial Intelligence Res* **2** (1995), 263–286.

[17] U. Kreβel, *Pairwise Classification and Support Vector Machines*, In Advances in Kernel Methods – Support Vector Learning, MIT Press, 1999, 185–208.

[18] Z. Lu, D. Szafron, R. Greiner, P. Lu, D.S. Wishart, B. Poulinand, J. Anvik, C. Macdonell and R. Eisner, Predicting subcellular localization of proteins using machine-learned classifiers, *Bioinformatics* **20**(4) (2004), 547–556.
[19] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.