Alternative Strategies in Learning Nonlinear Soft Margin Support Vector Machines

Catalina COCIANU[1], Luminita STATE[2], Cristian USCATU[1]
[1]Department of Informatics and Cybernetics
The Bucharest University of Economic Studies, Romania
[2]Department of Mathematics and Informatics,
University of Pitesti, Pitesti, Romania
ccocianu@ase.ro, lstate@clicknet.ro, cristian.uscatu@ie.ase.ro

The aims of the paper are multifold, to propose a new method to determine a suitable value of the bias corresponding to the soft margin SVM classifier and to experimentally evaluate the quality of the found value against one of the standard expression of the bias computed in terms of the support vectors. Also, it is proposed a variant of the Platt's SMO algorithm to compute an approximation of the optimal solution of the SVM QP-problem. The new method for computing a more suitable value of the bias is based on genetic search. In order to evaluate the quality of the proposed method from the point of view of recognition and generalization rates, several tests were performed, some of the results being reported in the final section of the paper.

Keywords: *Non-Linear Support Vector Machines, Kernel Function, Radial Basis Function, Soft Margin SVM, SMO Platt's Algorithm, Genetic Search, Classifier Design and Evaluation*

1 Introduction

Assume that the aim is to discriminate among the instantiations of *m* concepts or classes, conventionally represented by the labels $h_1, \dots, h_m$. Each instantiation comes from one and only one concept, refered as the true provenance class. The usual representation of instantiations is in terms of the values of a pre-selected finite set of *n* descriptors or attributes.

The aim is to find out a set of boundaries separating each pair of classes based on the information supplied by a finite set of examples coming from these classes, conventionally referred as a training set. The framework and the corresponding methodology strongly depend on the additional information concerning the set of examples. In a supervised framework for each example the label corresponding to the provenance class is supplied. Therefore, in this case, the basis for deriving a set of separating boundaries is represented by a finite set of labeled examples (x, y), where *x* is the representation of the particular example and *y* is a code representing the label of the provenance class of *x*.

Let us assume that we found somehow a suitable set of boundaries that correctly separate the available set of boundaries. Since the information concerning the classes is exclusively contained by the finite training set, there are no guarantees that the set of boundaries are "good enough" in the sense that using them, the provenance class of a new, unseen yet example can be inferred. In other words, the problem of generalization capacities arises in a very natural way.

Usually, there is no hint concerning the functional expression one should consider for the separating boundaries. Consequently, we could try to propose some parameterized expressions and fit the parameters against the particular training set. Obviously, the simplest expression of the boundaries is of linear type, but unfortunately, very seldom it happens that the provenance classes can be separated by linear type boundaries and moreover, even the available training set cannot be correctly separated this way.

In the following we consider the binary case that is the task is to find out a suitable separating boundary for two classes. Moreover, the aim is to find out a separating

DOI: 10.12948/issn14531305/18.2.2014.05

boundary of linear type that is a hyperplane in the space of examples that correctly separates the positive examples of the negative ones when this is possible and minimizes the number of misclassifications otherwise. Besides, in order to assure good generalization capacities we would like to find out a separating hyperplane placed at almost equal distance to the positive and negative examples respectively. In order to assure this property, a modern and powerful methodology has been lately developed yielding to the theory of Support Vector Machines (SVM). The fundamentals of this theory were established by Vapnik ([1], [2]). Several refinements have been proposed by many authors as for instance the use of kernels as a tool to maximize the quantity of information extracted from the training data ([3], [4], [5], [6], [7]).

The aims of the paper are multifold, to propose a new method to determine a suitable value of the bias corresponding to the soft margin SVM classifier and to experimentally evaluate the quality of the found value against one of the standard expression of the bias computed in terms of the support vectors. Also, it is proposed a variant of the Platt's SMO algorithm to compute an approximation of the optimal solution of the SVM QP-problem. The new method for computing a more suitable value of the bias is based on genetic search. In order to evaluate the quality of the proposed method from the point of view of recognition and generalization rates, several tests were performed, some of the results being reported in the final section of the paper.

2 Soft Margin SVM

Let $\mathcal{S} = \{(x_i, y_i),\ x_i \in \mathbb{R}^n, y_i \in \{-1,1\}, 1 \leq i \leq N\}$ the training dataset. We say that $\mathcal{S}$ is linearly separable if there exists a hyperplane in the space of inputs separating the positive to the negative examples. There are few known methods that allow to establish whether $\mathcal{S}$ is linearly separable or not, one of them being the celebrated Ho-Kashyap procedure [8], the computational complexity involved by these methods being substantially high. Consequently, the methods for estimating the parameters of the separating hyperplane facing the possibility that $\mathcal{S}$ is not linearly separable are looked for. The use of kernels is one of such methods, a kernel "hiding" a not explicitly given non-linear transform projecting the input data onto a higher dimensional new space in the hope that this way on one hand more information can be extracted from data, and on the other hand the subsets of positive and negative examples, and possibly the representations of the classes become linearly separable.

2.1 The Kernel-Based Learning Theory

One of the fundamental mathematical results underlying the kernel-based learning theory is the celebrated Mercer's theorem.

Definition. Let A be a compact subset of $\mathbf{R^n}$, for some $n \in \mathbf{N}^*$, and σ_n the Lebesque measure on $\left(\mathbf{R^n}, \mathcal{B}_n\right)$, where $\mathcal{B}_n$ stands for the σ-algebra of n-dimensional Borelian sets. The symmetric function $K : A \times A \rightarrow \mathbf{R}$ is said to be a positive defined kernel on A if the following conditions hold,

C1. for any finite number N and for any finite set of points $\{x_i, i = 1,\dots,N\} \subset A$ and for any real numbers $\{a_i, i = 1,\dots,N\}$,

$$\sum_{i,j=1}^{N} a_i a_j K\left(x_i, x_j\right) \geq 0$$

C2. $\int_A \int_A K^2(x, y) d\sigma_n(x) d\sigma_n(y) < \infty$

If K is a positive defined kernel on A, then it induces the integral operator $L_K : L^2\left(\mathbf{R^n}\right) \rightarrow L^2\left(\mathbf{R^n}\right)$ given by, for any $f \in L^2\left(\mathbf{R^n}\right)$,

$$\left(L_K f\right)(x) = \int_{\mathbf{R^n}} K(x,t) f(t) d\sigma_n(t)$$

The integral operator L_K is called the Hilbert-Schmidt operator induced by the kernel K. It can be proved (Mercer, 1908) that L_K is a self-adjoint, positive, compact operator having a countable system of non-negative eigenvalues $\{\lambda_k\}_{k=1,\infty}$ satisfying

DOI: 10.12948/issn14531305/18.2.2014.05

$\sum_{k=1}^{\infty}\lambda_k^2 < \infty$ and the corresponding $L^2(A)$-normalized eigenfunctions $\{\phi_k\}_{k=\overline{1,\infty}}$ form an orthonormal basis of $L^2(A)$.

Theorem Mercer. Let A be a closed subset of $\mathbf{R^n}$ and K be continuous symmetric function such that C1 and C2 hold. Then, for any $x, y \in \mathbf{R^n}$,

$$K(x,y) = \sum_{k=1}^{\infty}\lambda_k\phi_k(x)\phi_k(y),$$

where the series converges absolutely for each pair $(x,y)\in A\times A$ and uniformly on each compact subset of A.

Let $g : A \to \mathbf{F}$, $g(x) = \left(\sqrt{\lambda_k}\,\phi_k(x), k = 1,2,...\right)$, where **F** is called the feature space. Each eigenfunction ϕ_k is conventionally referred as a selected feature, the corresponding eigenvalue λ_k being taken as the value of the feature ϕ_k for the example *x*. The function g is called a feature extractor and for each $x \in$ $\mathbf{R^n}$, $g(x)$ is the representation of the example *x* in the feature space.

By construction, the dimensionality of **F** is determined by the number finite, or denumerable infinite, of positive eigenvalues of the kernel K.

In the particular case when the number of positive eigenvalues of the kernel K is finite, say *m*, then the dimensionality of **F** equals *m* and conventionally $g(x)$ is represented as a *m*-dimensional column vector and for any $(x,y)\in A\times A$, we get

$$g^T(x)g(y) = \sum_{k=1}^{m}\lambda_k\phi_k(x)\phi_k(y) = K(x,y).$$

For simplicity sake, in the more general cases when the dimensionality of **F** is infinite, we extend the notation to represent the inner product defined on **F** by the series

$$g^T(x)g(y) = \sum_{k=1}^{\infty}\lambda_k\phi_k(x)\phi_k(y) = K(x,y).$$

If *g* is a particular selected feature extractor $g : A \to \mathbf{F}$, then the function $K(x,y) = g^T(x)g(y)$ is a semi-positive defined kernel. The kernel “trick” consists in assuming a particular expression for a positive defined kernel K, as for instance a polynomial or exponential expression. According to the Mercer theorem, there exists a feature extractor g such that $K(x,y) = g^T(x)g(y)$ holds, where neither the explicit functional expression of g nor the dimensionality of **F** are known. However, this information is not really needed, because the computations involving the kernel K are carried out in the initial *n*-dimensional space.

Note that the values of $K(x,x')$ increases as *x* and *x*’ become “closer”, that is the kernels given in Table 1 correspond to some similarity measures on $\mathbf{R^n}$.

Table 1. The comparative analysis of the recognition rates

Method	**Recognition rate**
Linear discriminant function	83%
Quadratic discriminant function	84.50%
Mahalanobis-type discriminant function	79%
Soft margin SVM - SMO algorithm using the Gauss kernel and the bias given by (10)	87.25%
Soft margin SVM - SMO algorithm using the Gauss kernel and the bias computed by the genetic algorithm	87.75%
Soft margin SVM - SMO algorithm using the exponential kernel and the bias given by (10)	84.25%
Soft margin SVM - SMO algorithm using the exponential kernel and the bias computed by the genetic algorithm	87.50%

2.2. The non-linear soft margin SVM

The non-linear transform is a vector valued function $g: \mathbb{R}^n \to \mathcal{F}$, the image of $\mathcal{S}$ in the space $\mathcal{F}$ being given by $\mathcal{S}_g = \{(g(x_i), y_i)$, $xi\in\mathbb{R}n,yi\in-1,1,1\leq i\leq N$. The transform g is a feature extractor, and $\mathcal{F}$ is the feature space. When the dimension of $\mathcal{F}$ is finite, say m, a pair $(w,b) \in \mathbb{R}^m \times \mathbb{R}$, defines the classifier $\bar{y}_{w,b}(x) = \begin{cases} 1, w^T g(x) + b \geq 0 \\ -1, w^T g(x) + b < 0 \end{cases}$, for any input data *x*.

The feature extractor g is designed as follows. Let *K* be a function that "hides" the explicit functional expression of g. Then, the evaluation of the expression $(w^T g(x) + b)$ is performed exclusively in terms of K and the resulted feature space cannot be explicitly known. The core result in approaches of this type is the celebrated theorem due to Mercer [9]. The most frequently used kernel functions are the polynomial kernel, the Gauss Radial Basis Function (RBF) and the exponential RBF, their expressions being

$K(x,y) = \left(x^T y + 1\right)^d, \quad d \geq 1$

$K(x,y) = \exp\left(-\gamma\|x-y\|^2\right), \gamma > 0$

$K(x,y) = \exp\left(-\gamma\|x-y\|\right), \gamma > 0$ respectively.

Let us assume that for a selected kernel *K*, $\mathcal{S}_g$ is non-linearly separable. Let us denote by g the feature extractor such that $K(x,x') = g(x)^T g(x')$ and $\bar{y}_{w,b}$a linear classifier in the feature space of parameter (w,b), that is, for the input x, $\bar{y}_{w,b}(x) = 1$ if and only if $w^T g(x) + b \geq 0$. The model of soft-margin SVM assumes a set of slack variables $\xi_1, \xi_2, \dots, \xi_N$, where ξ_i expresses the magnitude of the error committed by $\bar{y}_{w,b}$ for the observation (x_i, y_i), that is $\xi_i = max\{0, 1 - y_i(w^T g(x_i) + b)\}$.

For any misclassified example (x_i, y_i), the value of ξ_i expresses the magnitude of the error committed by the classifier $\bar{y}_{w,b}$ with respect to (x_i, y_i). The overall importance of the cumulated errors usually can be expressed as

$$F\left(\sum_{i=1}^{N} \xi_i^t\right) \qquad (1)$$

where F is a convex and monotone increasing function and $t > 0$ is a weight parameter.

We obtain a QP problem [10]

$$\begin{cases} minimize \left\{\frac{1}{2}\|w\|^2 + CF\left(\sum_{i=1}^{N} \xi_i^t\right)\right\} \\ y_i(w^T g(x_i) + b) \geq 1 - \xi_i, 1 \leq i \leq N \\ \xi_i \geq 0, 1 \leq i \leq N \end{cases} \qquad (2)$$

where C is a conventionally selected constant used to weight the effect of the cumulated errors.

Being given its complexity, the problem (2) cannot be solved in this general form, but only for particular functional expressions of F and the weight parameter t. The simplest model uses $F(u) = u$ and $t = 1$, in this case the problem (2) becomes the constrained QP-problem

$$\begin{cases} minimize \left\{\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i\right\} \\ y_i(w^T g(x_i) + b) \geq 1 - \xi_i, 1 \leq i \leq N \\ \xi_i \geq 0, 1 \leq i \leq N \end{cases} \qquad (3)$$

whose dual QP-problem is

$$\begin{cases} maximize\ Q(\propto) \\ \sum_{i=1}^{N} \propto_i y_i = 0 \\ 0 \leq \propto_i \leq C, \quad 1 \leq i \leq N \end{cases} \qquad (4)$$

where

$$Q(\propto) = \\ = \sum_{i=1}^{N} \propto_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \propto_i \propto_j y_i y_j K(x_i, x_j)$$

To conclude, in order to design a soft-margin SVM, a particular expression of the kernel function *K* and the magnitude of the constant *C* have to be selected in advance, then the optimization QP-problem (4) has to be solved.

If $\propto^* = (\propto_1^*, \propto_2^*, \dots, \propto_N^*)^T$ is a solution of (4), then the parameter w^* is

$$w^* = \sum_{i=1}^{N} \propto_i^* y_i g(x_i), \qquad (5)$$

Since the solutions of (4) do not involve the parameter *b*, its value should be determined such that $1 - \min_{i, y_i = 1} w *^T g(x_i) \leq b^* \leq$

DOI: 10.12948/issn14531305/18.2.2014.05

$-1-\max_{i,y_i=-1} w *^T g(x_i)$holds, therefore more options concerning b^* are allowed ([3], [4]). One of the most used expressions of b^* is

$$b^* =$$

$$= -\frac{1}{2}\left\{\max_{i,y_i=-1}\sum_{j=1}^{N} \propto_j^* y_j K(x_j, x_i) + \min_{i,y_i=1}\sum_{j=1}^{N} \propto_j^* y_j K(x_j, x_i)\right\} \qquad (6)$$

3. Adaptive Algorithms to Approximate the Parameters of a Soft Margin SVM Classifier

According to the arguments supplied in the Section 2, the core problem of the computation of the soft margin SVM separating hyperplane is represented by the QP-problem (4). So far there have been proposed several methods for approximating a solution of (4).

3.1. A Variant of Platt's SMO Algorithm

Sequential minimal optimization (SMO) algorithm was introduced by Platt [11] and further extended by several authors ([12], [13]) is a simple algorithm that allows to solve the SVM-QP problem without extra-matrix storage by decomposing the overall QP-problem into simple QP sub-problems similar to Osuna's method [14]. The idea of the SMO algorithm [11] is to solve the smallest optimization problem at each step, in case of the QP-problem corresponding to the soft margin SVM, the smallest optimization sub-problem involving only two Lagrange multipliers. In this section we present a variant of the Platt's SMO algorithm to approximate a solution of (4).

Let K be a kernel satisfying the conditions of the Mercer's theorem and g its corresponding feature extractor, that is $K(x,x') = g(x)^T g(x'), \forall x, x' \in \mathbb{R}^n$. Let us denote by $f_\propto(x) = w^T g(x) + b$, where $w = \sum_{i=1}^{N} \propto_i \; y_i g(x_i)$ is the parameter of a separating hyperplane. Then

$$f_\propto(x) = b + \sum_{i}^{N} y_i \propto_i K(x, x_i) \qquad (7)$$

The idea of the SMO algorithm is to use a predefined constant $C > 0$, and a tolerance parameter $\tau > 0$, expressing a sort of tradeoff between accuracy and efficiency. At each step two examples (x_p, y_p), (x_q, y_q) are looked for such that the following condition holds,

$$\left(f_\alpha(x_p) - y_p + \tau < f_\alpha(x_q) - y_q - \tau\right) \wedge \Big(\big(\alpha_p < C \wedge y_p = 1\big) \vee \big(\alpha_p > 0 \wedge y_p = -1\big)\Big) \wedge \Big(\big(\alpha_q < C \wedge y_q = -1\big) \vee \big(\alpha_q > 0 \wedge y_q = 1\big)\Big) \qquad (8)$$

Let us assume that, at the current step, there exists at least a pair (x_p, y_p), (x_q, y_q) for which (8) holds. The entries $\propto_p$ and $\propto_q$ of the current parameter α are modified such that to increase $f_\alpha(x_p)$and to decrease$f_\alpha(x_q)$.

Since the updated parameter has to fulfill the constraint $\sum_{i=1}^{N} \alpha_i y_i = 0$, the updating rules are,

$$\propto_p \leftarrow \propto_p + y_p\eta$$

$$\propto_q \leftarrow \propto_q - y_q\eta$$

where

$$\eta = \frac{\left(f_\propto(x_q) - y_q\right)\left(f_\propto(x_p) - y_p\right)}{K(x_p, x_p) - 2K(x_p, x_q) + K(x_q, x_q)} \qquad (9)$$

If the conditions $0 \leq \propto_p + y_p\eta \leq C$ and $0 \leq \propto_q - y_q\eta \leq C$ do not hold, the value of the tolerance parameter η should be decrease accordingly. In case, at a certain step, there are no examples (x_p, y_p), (x_q, y_q) such that (8) holds, the search process is stopped.

Our variant of the Platt's SMO algorithm uses the following updating rules. Let (x_p, y_p), (x_q, y_q) be a pair of examples such that (8) holds, and $\propto^{old}$the current parameter. Then,

$$\propto_p = \propto_p^{old} + y_p\eta$$

$$\propto_q = \propto_q^{old} - y_q\eta$$

where η is given by (9). The value of the parameter η should be adjusted to assure that the updated values $\propto_p$ and $\propto_q$ still belong to $[0, C]$. Our option is for the following adjusting strategy. Assume that at least one of the entries $\propto_p, \propto_q$ does not belong to $[0, C]$.

1. In case $y_p = y_q = 1$, then η is set to $min(\propto_q^{old}, C - \propto_p^{old})$. Indeed, since $\eta > 0$, $\propto_p > 0$ and $\propto_q < C$, therefore at least one of the inequalities $\propto_p > C$, $\propto_q < 0$ holds. If $\propto_p^{old} + \propto_q^{old} < C$, then we set $\eta = \propto_q^{old}$, otherwise we set $\eta = C - \propto_p^{old}$and we get $\propto_q^{old} - (C - \propto_p^{old}) = \propto_q^{old} + \propto_p^{old} - C \le 2C - C = C$.
2. In case $y_p = 1, y_q = -1$, the setting of the parameter η is $C - max\{\propto_p^{old}, \propto_q^{old}\}$. In order to prove that for this setting both parameters $\propto_p$ and $\propto_q$ belong to $[0, C]$, we have to take into account only the case when at least one of the entries $\propto_p, \propto_q$ does not belong to $[0, C]$, that is $\propto_p > C$ and/or $\propto_q > C$. If $\propto_p^{old} < \propto_q^{old}$, then the setting is $\eta = C - \propto_q^{old}$, otherwise $\eta = C - \propto_p^{old}$ and obviously both updated entries belong to $[0, C]$.
3. In case $y_p = -1, y_q = 1$ then η is set to $min\{\propto_p^{old}, \propto_q^{old}\}$. In this case at least one of the inequalities $\propto_p < 0, \propto_q < 0$ holds. If $\propto_p^{old} > \propto_q^{old}$, then we set $\eta = \propto_q^{old}$. Similarly, if $\propto_p^{old} \le \propto_q^{old}$ then $\eta = \propto_p^{old}$, therefore $0 \le \propto_q = \propto_q^{old} - \propto_p^{old} \le C$ and $\propto_p = 0$.
4. If $y_p = y_q = -1$, then η is set to $min(\propto_p^{old}, C - \propto_q^{old})$. In this case at least one of the inequalities $\propto_q > C$, $\propto_p < 0$ holds. If $C - \propto_q^{old} < \propto_p^{old}$, then $\eta = C - \propto_q^{old}$, otherwise $\eta = \propto_p^{old}$, therefore $\propto_p = 0$, and $0 \le \propto_q = \propto_q^{old} + \propto_p^{old} \le C$.

The implementation of this variant of the Platt's SMO algorithm uses the stopping condition $\mathcal{C}$ defined in terms of the tolerance parameter $\tau > 0$ and $\mathcal{C} = true$ when there is no pair of examples (x_p, y_p), (x_q, y_q) such that (8) holds.

3.2. Tuning the Bias Parameter b on the Basis of a Genetic Approach

The class of genetic algorithms (GA) is a relatively new kind of computation, referred as natural computation, providing an alternative in solving hard optimization problems where the high dimensionality determines that the computations involved by the classical optimization methods become intractable ([15], [16]).

The general scheme of genetic algorithms can be briefly described as follows. Let us denote by

$f: \mathfrak{D} \rightarrow \mathbb{R}$ a function whose maximization on $\mathfrak{D}$ is aimed. For each $x \in \mathfrak{D}$, $f(x)$ represents the quality of *x* from a certain point of view, therefore the function *f* is usually referred as a fitness function. The search for a maxima point of the function f is an iterative team-like process, where, at each step, the team consists of the best approximations of a maxima point found so far. At each step, the current team is referred as the current population and consists of a certain finite number of elements not necessarily distinct, belonging to $\mathfrak{D}$, the sizes of populations being either fixed or dynamically computed during the search process. The initial population $\mathcal{P}_0$ used when the algorithm starts consists of a finite number of randomly selected individuals belonging to $\mathfrak{D}$. The genetic algorithm uses a finite number of so-called recombination operators which represents a certain mechanism to generate elements from $\mathfrak{D}$. Let us denote by $\mathcal{P}_i$ the current population at the i-th iteration. The new population is usually computed by retaining a certain number of the best individuals of $\mathcal{P}_i$, a certain number of so-called "parents" $\mathcal{BP}_i \subset \mathcal{P}_i$ and by including the offsprings of $\mathcal{BP}_i$, the individuals resulted by applying the recombination operators to the elements selected in $\mathcal{BP}_i$. The searching process is over when a stopping condition $\mathcal{S}$ holds, usually expressed in terms of a threshold imposed on the number of iterations.

DOI: 10.12948/issn14531305/18.2.2014.05

Accordingly, the general scheme of a genetic algorithm is.

```
    $i \leftarrow 0$
    Step 1. Initialization: $\mathcal{P}_i$ resulted by randomly selet a given number of individuals of $\mathfrak{D}$
    Step2. Evaluation: for each $x \in \mathcal{P}_i$ compute $f(x)$
    Step3. Repeat
    3.1. Select the mating pool $\mathcal{BP}_i$
    3.2. Apply the recombination operators
    3.3. Evaluate the resulted candidates
    3.4. Deterimine the next population $\mathcal{P}_{i+1}$;
        3.5. $i \leftarrow i + 1$
    until $\mathcal{S}$
```

We use a genetic algorithm to find out from data a suitable value of the bias parameter b aiming to maximize the mean recognition rate in discriminating between two classes.

In order to set the parameters of the genetic algorithm we use the fitting function $F(b)$ representing the mean recognition rate of the linear classifier (w^*, b), where w^* is an approximate of the optimal solution of (7) computed by the SMO algorithm.

The search space is established according to the following simple argument. First of all, it is quite natural to assume that for a given setting (w^*, b) the corresponding linear classifier correctly classifies in the feature space at least one sample coming from each class. Then, assume that x_i is correctly classified by the classifier (w^*, b). If $y_i = 1$, then ${w^*}^T g(x_i) + b > 0$, therefore $b > -{w^*}^T g(x_i) \geq -\max_{y_k=1} {w^*}^T g(x_k)$. If $y_i = -1$, then ${w^*}^T g(x_i) + b < 0$, that is $b < -{w^*}^T g(x_i) \leq -\min_{y_k=-1} {w^*}^T g(x_k)$.

We arrive at the conclusion that the values of the bias parameter b lye in the interval $I = \left[-\max_{y_k=1} {w^*}^T g(x_k), -\min_{y_k=-1} {w^*}^T g(x_k)\right]$.

The settings of the fixed-size population of the genetic algorithm are:

- the population sizes depend on the length of the interval I, lying between 10 and 20; larger population size is set in case of longer intervals. In our tests we used relatively small size populations, because a long series of tests proved that, in case of larger size populations, in spite of lack of significant improvements the computational complexity is significantly increased;
- the individuals of the initial population are randomly selected from the interval I;
- the parent selection mechanism is performed according to the roulette strategy based on the fitness proportionally selection probability distribution;
- we use only one recombination operator, the crossover, implemented as a convex combination of two chromosomes;
- the survival generation is obtained in elitist way; we consider two strategies, namely,
  a) the new generation is composed by taking the offsprings, except the case when the best parent is more fitted than the best offspring, in this case the less fitted offspring is replaced by the best fitted parent;
  b) composing the new generation by selecting the best individuals from the current population and the generated offsprings;
- the stopping condition is formulated by imposing a threshold on the number of generations; in our tests the upper limit on the number of generations is 7. The value of the upper limit was set to 7 because many tests pointed out a quick stabilization around the maximum value of the fitness function.

We implemented the previously described genetic algorithm for computing an approximate of the bias value that guarantees the highest mean recognition rate. In our tests we used two datasets, training and test respectively. The parameter vector w^* is determined on the basis of the training set using the SMO algorithm and the genetic algorithm is applied on the overall data set resulted as the union of training and test datasets.

4 Comparative Analysis

The developments in this section analyze the effects of different choices of the bias parameter b^* on the generalization capacities evaluated in terms of the mean recognition rate corresponding to the resulted classifier.

It is well known that the value of the bias parameter b^* cannot be computed by solving the QP-problem (4) and there have been proposed several computation rules expressed in terms of the support vectors, as for instance (6). In our developments we used the expression (10) proposed in [17], where $b_i = y_i - \sum_{j=1}^{N} \propto_j^* y_j K(x_i, x_j), i = 1, \ldots, N$, and SV is the set of support vectors, in order to refine the bias by taking into account the relative importance of the support vectors.

$$b^* = \frac{1}{|SV|} \sum_{x_i \in SV} \propto_i^* b_i \qquad (10)$$

Our tests were performed on artificially generated data from Gaussian repartitions. Also, we used two types of kernels, the Gauss Radial Basis Function (GRBF), $K(x,x') = exp(-\gamma\|x-x'\|^2), \gamma > 0$ and the Exponential Radial Basis Function (ERBF), $K(x,x') = exp(-\gamma\|x-x'\|), \gamma > 0$, where the value of the parameter γ was determined such that the recognition rate is optimized.

For instance, the results of the comparative analysis in case the data were generated from $N\left(\binom{-10}{0}, \begin{pmatrix} 1.69 & 1.350 \\ 1.35 & 2.50 \end{pmatrix}\right)$ and $N\left(\binom{-8}{4}, \begin{pmatrix} 6.27 & 1.02 \\ 1.02 & 12.27 \end{pmatrix}\right)$ are presented in Figure 1, Figure 2 and Figure 3 and they are summarized in Table2. The training data consisted of 150 examples coming from each class and the test data contained 400 examples coming from each class. The training and test data are represented in Figure 1 and Figure 2 respectively. The computed support vectors are depicted in Figure 3. The best recognition rate 87.75% was obtained in case of the variant of SMO algorithm described in Section 3, with GRBF kernel, $\gamma = 0.05$ and the bias computed by the genetic algorithm.

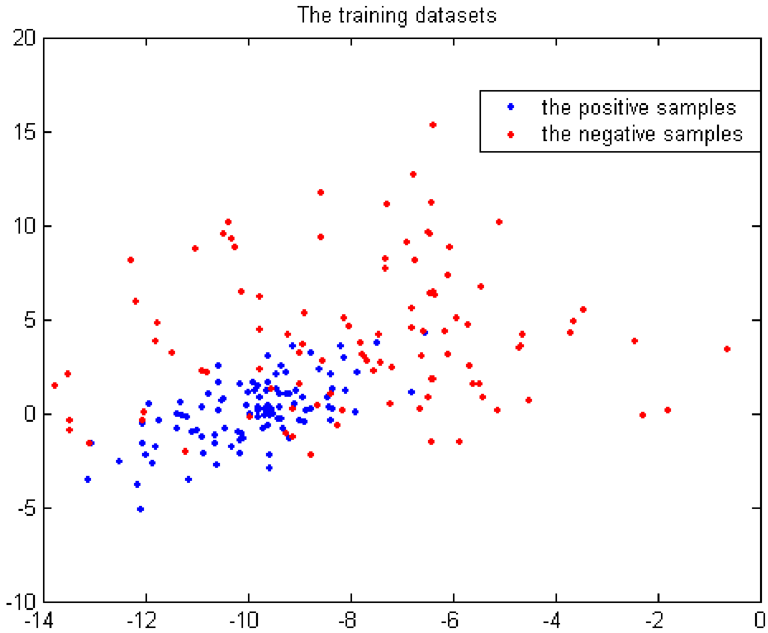

Fig. 1. The training data

DOI: 10.12948/issn14531305/18.2.2014.05

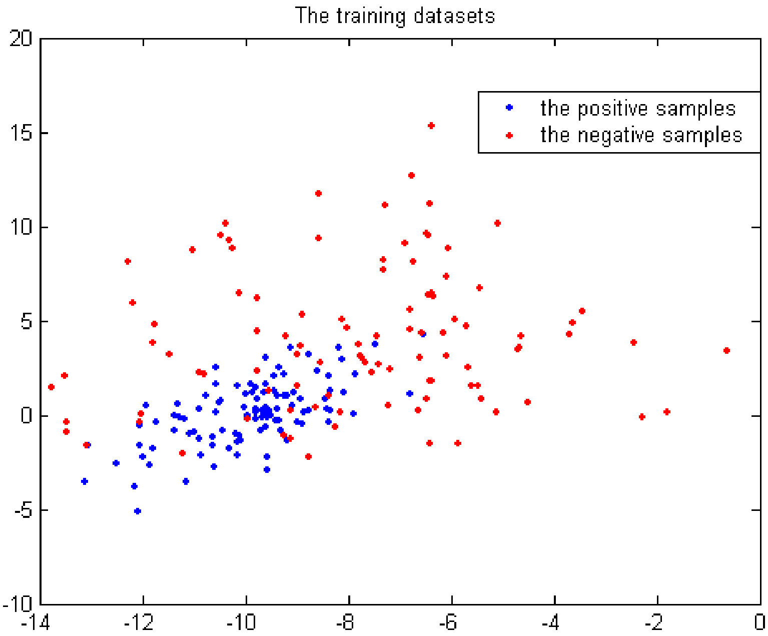

Fig. 2. The test data

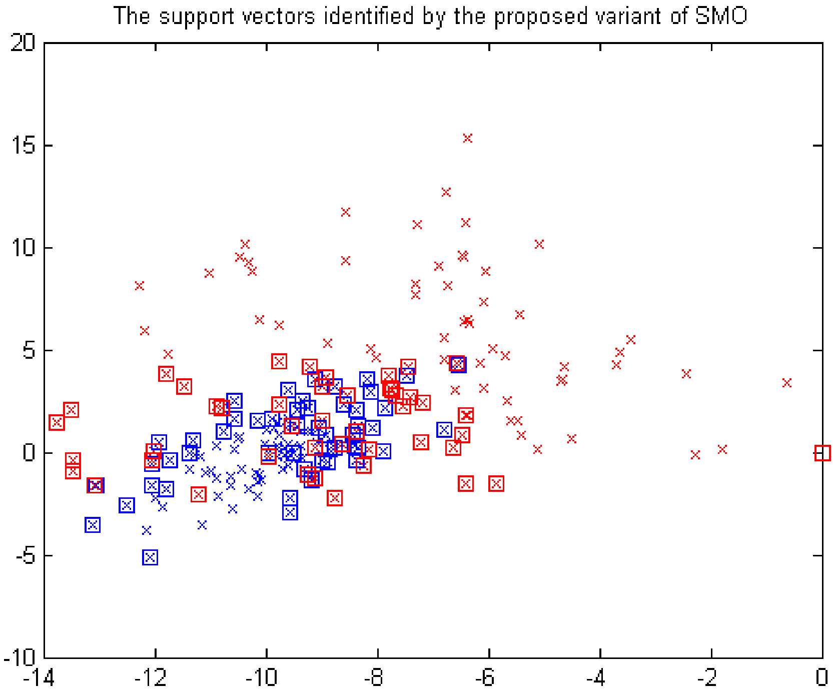

Fig. 3. The support vectors computed by the variant of SMO algorithm

The tests pointed out that the variation of the recognition rates depends on the inner structure of the classes from which the learning data come as well as on their separability degree. Consequently, the results are encouraging and entail future work toward extending these refinements to multi-class classification problems and approaches in a fuzzy-based framework.

References

[1] V. Vapnik, *The Nature of statistical learning theory*. New York: Springer-Verlag, 1995.

[2] V. Vapnik, *Statistical learning theory*. New York: John Wiley, 1998.

[3] S. Abe, *Support vector machines for pattern classification*, in Advances in Pattern Recognition. New York:

DOI: 10.12948/issn14531305/18.2.2014.05

Springer-Verlag, 2010, pp. 1-473.

[4] J. Shawe-Taylor and N. Cristianini, *Support vector machines and other kernel-based learning methods*. UK: Cambridge University Press, 2000, pp. 1-198.

[5] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. UK: Cambridge University Press, 2004, pp. 1-474.

[6] W. Liu, J. Principe and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. New Jersey: John Wiley, 2010, pp. 1-240.

[7] C. Cocianu and L. State, “Kernel-Based Methods for Learning Non-Linear SVM,” Economic Computation and Economic Cybernetics Studies and Research, vol. 47, no. 1, pp. 41-60, 2013.

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. 2nd ed. New York: Wiley, 2001.

[9] J. Mercer, "Functions of positive and negative type, and their connection with the theory of integral equations," Proc. Roy. Soc. London Ser. A , 83, pp. 69–70, 1908.

[10] C. Cortez and V. Vapnik, “Support Vector Networks”, Machine Learning, vol. 20, no. 3, pp. 273-297, September 1995.

[11] J. Platt, “Fast Training of Support Vector Machines Using Sequential Minimal Optimization,” in *Advances in Kernel Methods – Support Vector Learning*. Cambridge, MA: MIT Press, 1998.

[12] S. S. Keerthi and S. K. Shevade, “SMO algorithm for least squares SVM formulations,” Neural Computation, vol. 15, pp. 487-507, 2003.

[13] T. Knebel, S. Hochreiter and K. Obermayer, “An SMO Algorithm for the Potential Support Vector Machine,” Neural Computation, vol. 20, no. 1, pp. 271-287, January 2008.

[14] E. Osuna, R. Freund and F. Girosi, “An improved training algorithm for support vector machines,” in Proc. the IEEE Workshop. Neural Networks for Signal Processing, 1997, pp. 276-285.

[15] A. E. Eiben and J. E Smith, Introduction to Evolutionary Computing, Springer-Verlag, 2003.

[16] G.E. Goldberg, Genetic Algorithms in Search Optimization Machine Learning, Addison-Wesley, New York, USA, 2005.

[17] W. An and M. Liang, “Fuzzy support vector machine based on within-class scatter for classification problems with outliers or noises,” Neurocomputing, vol. 110, pp. 101-110, June 2013.

Catalina-Lucia COCIANU, Professor, PhD, currently working with Academy of Economic Studies, Faculty of Cybernetics, Statistics and Informatics, Department of Informatics in Economy. Competence areas: machine learning, statistical pattern recognition, digital image processing. Research in the fields of pattern recognition, data mining, signal processing. Author of 15 books and more than 90 papers published in national and international journals.

Luminita STATE, Professor, PhD, currently working with University of Pitesti, Department of Mathematics and Computer Science. Competence areas: artificial intelligence, machine learning, statistical pattern recognition, digital image processing. Research in the fields of machine learning, pattern recognition, neural computation. Author of 15 books and more than 120 papers published in national and international journals.

DOI: 10.12948/issn14531305/18.2.2014.05

Cristian Razvan USCATU, Associated Professor, PhD, currently working with the Academy of Economic Studies, Faculty of Cybernetics, Statistics and Informatics, Department of Informatics in Economy. Competence areas: computer programming, data structures. Author/co-author of 10 books and more than 40 papers published in national and international journals.

DOI: 10.12948/issn14531305/18.2.2014.05