Professor Catalina COCIANU, PhD
E-mail: Catalina.Cocianu@ie.ase.ro
The Bucharest Academy of Economic Studies
Professor Luminita STATE, PhD
University of Pitesti
E-mail: lstate@clicknet.ro

KERNEL-BASED METHODS FOR LEARNING NON-LINEAR SVM

Abstract. *The paper reports a model-free approach to the design of a non-linear classifier of SVM type. SVMs are usually viewed as "non-parametric" models, where the attribute non-parametric does not refer to the lack of parameters in the SVMs models, the learning problem of the SVM's parameters being of crucial importance. The novelty of the proposed method is given by a new expression for the bias parameter and a refined version of gradient ascent method for solving the SVM - QP problem. The tests pointed out good convergence properties and, moreover, the proposed modified variants proved higher convergence rates as compared to the standard SMO algorithm. Also, the performance of the resulted classifier proves better as compared to the standard choice of the bias.*

Key words: *Support Vector Machine, kernel functions, gradient ascent algorithm.*

JEL classification: C02, C14, C19, C45, C49, C61

1. Introduction

Let us denote by *S* a system of unknown input-output dependency, the unknown dependency being of deterministic/non-deterministic, linear/non-linear type. Besides, it is possible that the output is influenced by the observable input as well as a series of unobservable latent factors. There is no information about the underlying joint probability distribution corresponding to the (possible) non-linear dependency $y = f(x)$ between the high dimensional space of inputs *x* and the output space of *S*, therefore one is forced to perform a distribution-free learning. The only information available is a finite size set of training data $S_N = \{(x_i, y_i), x_i \in \mathbf{R}^n, y_i \in \{-1,1\}, 1 \le i \le N\}$ consisting of input-output observations.

The basic components in a probabilistic setting for supervised learning from data are *Generator (G)*, *System (S)* and *Learning Machine (LM)* (Figure 1), where

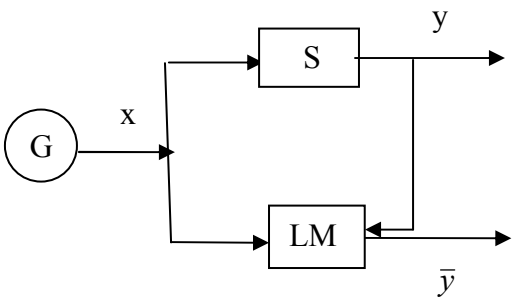

Figure 1.The basic components of a learning system

- G is the component responsible with the generation of samples from a n-dimensional input space. The samples are generated by G according to a probabilistic distribution that can be either known (observational setting) or unknown (un-observational setting) by the experimenter;
- S is a system that computes for each input x a m-dimensional output y according to an unknown input/output dependency;
- LM is a component that aims to find out (discover, learn) the unknown input/output dependency of S. Each input x is fed into S and LM and the component LM emits the m-dimensional output $\bar{y}$ according to the current hypothesis. The computed output y is fed into LM, and the learning algorithm implemented at the level of LM re-computes a new hypothesis on the basis of the current hypothesis, x, y and $\bar{y}$.

The indexed set of hypothesis Ω can be finite or infinite (numerable/continuous), each particular hypothesis being usually expressed in terms of a finite set of parameters.

In other words, the learning component LM implements an indexed family Ω of possible hypothesis about the unknown input/output dependency of S, and the learning algorithm corresponds to a search strategy in the space of hypothesis aiming to tune to the available history of S represented by pairs (x, y) that were explicitly made known to LM. In the particular case when the output space is $\{-1,1\}$, we say that the scheme represented in Figure 1 corresponds to a classification problem. In case of a two-class classification problem, S discriminates between the examples coming from two classes, say h_1, h_2, that is the unknown input-output dependency of S corresponds to the true provenance class of the input *x*, S computes $y = 1$ if *x* comes from h_1, and -1 otherwise. The component LM is a classifier, that is, to each hypothesis corresponds a decision rule concerning the provenance class of the input.

The simplest classifiers are of parametric type, in this case the decision function involves the components of the input and a finite set of parameters, each hypothesis corresponding to a particular parameter value. Being given their

simplicity, the linear classifiers are of a particular interest, in this case the decision concerning the provenance class being based on the value of a linear combination of the input components. The parameter corresponding to each hypothesis is the set of coefficients involved in the decision function. In other words, in case of linear classifiers each hypothesis corresponds to a hyperplane in the space of inputs assumed to separate the two classes. Usually, only incomplete information about the classes h_1, h_2 is available, as for instance characterizations in probabilistic or fuzzy terms or finite size sets of examples such that for each example, the true provenance class is known. The concept of Support Vector Machine (SVM) was introduced by Vapnik and Chervonenkis (Vapnik, 1998), and the problem of learnig SVM is, from mathematical point of view, a quadratic problem (QP) (Vapnik, 1998; Abe, 2010).

SVMs are usually viewed as "non-parametric" models, where the attribute non-parametric does not refer to the lack of parameters in the SVMs models, the learning problem of the SVM's parameters being of crucial importance. However, unlike in classic statistical inference, the parameters are not predefined, their number depending essentially on the particular training data, that is the parameters of SVMs are data driven and define the capacity of the model to match to data complexity.

In the simplest pattern recognition task, SVM uses a linear separating hyperplane to determine a classifier with a maximum margin. In order to accomplish this aim, the learning problem for the SVM becomes a constrained non-linear optimization problem, the cost function being quadratic and the constraints of linear type. In more complex cases, when the classes among which it is intended to discriminate can not be linearly separated in the initial input space, the first step is to transform the original input space into a higher dimensional feature space by using various non-linear mappings; polynomial, sigmoid, radially symmetric functions, multiquadrics of different spline functions, in the hope that the data would become linearly separable in the resulted new higher dimensional feature space.

Unfortunately, in spite of the advantage that the images of the initial non-linearly separable data in the feature space correspond to a linearly separable training sequence, the computational complexity could become intractable. The "kernel trick" prevents the increase of computational complexity and consists in using kernels that hide both the explicit expression of the feature extractor (the transform) and the dimension of the feature space. Moreover, the computation in the feature space can be expressed in terms of computations in the initial space, that is the computational complexity is not increased.

The work reported in this paper is a model-free approach to the design of a non-linear classifier of SVM type.

2. Kernel-based approaches in the design of a SVM classifier

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 $K : A \times A \to \mathbf{R}$ 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 $\{x_i, i = 1,..., N\} \subset A$ and for any real numbers $\{a_i, i = 1,..., N\}$,

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

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 $L_K : L^2\left(\mathbf{R^n}\right) \to L^2\left(\mathbf{R^n}\right)$ 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 $\sum_{k=1}^{\infty} \lambda_k^2 < \infty$ and the corresponding $L^2(A)$-normalized eigenfunctions $\{\phi_k\}_{k=1,\infty}$ 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 $x, y \in \mathbf{R^n}$,

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

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

Note that Mercer's theorem still holds if A a finite set, as for instance $A = \{-1,1\}^n$ and K is pointwise-defined positive defined.

Let $g : A \to \mathbf{F}$, $g(x) = \left(\sqrt{\lambda_k}\phi_k(x), k = 1,2,...\right)$, where $\mathbf{F}$ is referred as the feature space. Each eigenfunction ϕ_k is conventionally referred as a selected feature (attribute or characteristic), 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, 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 of strictly positive eigenvalues which can be finite or infinite. For instance, in case of Gaussian kernel, the dimensionality of **F** is infinite, that is, for any $x \in A$, $g(x)$ is a denumerable sequence of real numbers.

In the particular case when the number of strictly positive eigenvalues of the kernel K is finite (as it is for instance the case of polynomial kernels), say *m*, then the dimensionality of **F** equals *m* and conventionally $g(x)$ is represented as a *m*-dimensional column vector and for any $(x, y) \in A \times A$, the equality $g^T(x)g(y) = \sum_{k=1}^{m} \lambda_k \phi_k(x)\phi_k(y) = K(x, y)$ holds. 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

$$g^T(x)g(y) = \sum_{k=1}^{\infty} \lambda_k \phi_k(x)\phi_k(y) = K(x, y). \quad (2)$$

It is interesting to point out the relations of kernel-based approaches with Principal Component Analysis (PCA). Let *X* be a n-dimensional random vector and *S* the autocorrelation matrix of the repartition of *X*, $S = E\left(XX^T\right)$. Then the spectral representation of S is

$$S = \sum_{i=1}^{n} \lambda_i \psi_i \psi_i^T \quad (3)$$

where $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ are the eigenvalues and $\{\psi_1, \psi_2, ..., \psi_n\}$ is an orthogonal basis of unit eigenvectors of *S*. The random vector *X* is represented in terms of $\{\psi_1, \psi_2, ..., \psi_n\}$ as,

$$X = \sum_{i=1}^{n} Y_i \psi_i \quad (4)$$

where the random variables Y_i's are linear combinations of the components of *X*, $Y_i = \psi_i^T X, i = 1,2,...,n$, where each eigenvector ψ_i is a feature and Y_i is the value of this feature for *X*. Let $g = \left(\sqrt{\lambda_1}\phi_1, \sqrt{\lambda_2}\phi_2, ..., \sqrt{\lambda_n}\phi_n\right)$, where ϕ_i : $\mathbf{R}^n \rightarrow \mathbf{R}$, $\phi_i(x) = \psi_i^T x$. Then

$$K(x, y) = g^T(x)g(y) = \sum_{k=1}^{n} \lambda_k \phi_k(x)\phi_k(y) = x^T \left(\sum_{k=1}^{n} \lambda_k \psi_k \psi_k^T \right) y = x^T S y \quad (5)$$

is a positive defined kernel on any *A* compact *n*-dimensional Borelian set. Indeed,

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

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

because S is a symmetric positive defined matrix.

Also, for any compact set *A*, $\int_A \int_A K^2(x,y)dxdy < \infty$ holds because K is a continuous function.

Obviously, if *g* is a particular selected feature extractor $g : A \to \mathbf{F}$, where the dimensionality of **F** is possibly infinite, then the function *K* defined by (2) is a 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 (2) 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.

Some of the most frequently used kernels are presented in Table 1.

Table 1. Standard expressions of kernels

	$K(x,x')$
Polynomial of degree *d*	$(x^T x'+1)^d, \quad d \geq 1$
Gauss RBF	$\exp(-\gamma\|x-x'\|^2)$
Exponential RBF	$\exp(-\gamma\|x-x'\|)$

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 $\mathbf{R^n}$.

In case of a two-class discrimination problem, being given a finite set of examples coming from these classes such that for each example the true provenance class is known, the problem is to find a classification decision function that correctly identifies the true provenance class for each particular example. Moreover, in case the only information available about the classes is given by a finite set of examples, the classification decision function is aimed to posses generalization capacities, that is, to correctly identify the provenance class for new, unseen data. The simplest classification models correspond to linear decision functions, in this case the surface separating the regions assigned to the classes being a hyperplane. Let $S_N = \{(x_i, y_i), x_i \in \mathbf{R}^n, y_i \in \{-1,1\}, 1 \leq i \leq N\}$ be the set of labeled examples, where the value of y_i is the indicator for the provenance class

of the example x_i. We say that S_N is linearly separable if there exists an hyperplane $H(w,b): w^T x + b = 0$ that separates the positive and negative examples, that is $y_i\left(w^T x_i + b\right) > 0 \quad 1 \le i \le N$. If (w,b) were known then the classification decision function $h_{b,w}: \mathbf{R}^n \to \{-1,1\}$,

$h_{b,w}(x) = \begin{cases} 1, & w^T x + b > 0 \\ -1, & w^T x + b < 0 \end{cases}$ would correctly identify the true provenance class for each example x_i. However, usually, the information whether S_N is or is not linearly separable is not available and, moreover, even when S_N is linearly separable the parameters (w,b) are unknown. If S_N is linearly separable then the information contained by S_N should be used to find the parameters (w,b) such that $H(w,b)$ correctly separates positive and negative examples. When either S_N is not linearly separable or it is not known how S_N is, the aim is to look for a feature extractor *g* in the hope that, in the feature space **F**, the representations of the examples of S_N are linearly separable, that is $\left\{(w,b) \middle| w \in \mathbf{R}^m, b \in \mathbf{R}, \left(w^T g(x_i) + b\right) y_i > 0, 1 \le i \le N\right\} \ne \varnothing$. Being given that in general g is a non-linear function, in such a case, if S_N is linearly separable in the feature space, then the surface correctly separating the classes in the initial space is not linear anymore. Using the kernel "trick", instead of looking for a suitable feature extractor *g*, different expressions of kernels can be considered.

For given *g*, in order to assure the best generalization capacity of the classifier, that is to predict well the true provenance class for new test examples, the parameters (w,b) should be such that the surface of equation $H_{b,w}(x): w^T g(x) + b = 0$ separates the representations in the feature space of the positive and the negative examples from S_N with the largest "gap" between them, that is the minimum distance of the representations to the hyperplane $H(w,b)$ should be maximized.

The SVM approach to the computation of an optimal margin classifier yields to the quadratic programming (QP) problem (Abe, 2010),

$$\begin{cases} \text{minimize } \frac{1}{2}\|w\|^2 \\ y_i\left(w^T g(x_i) + b\right) \ge 1, \quad 1 \le i \le N \end{cases} \tag{6}$$

The dual optimization problem yields to the QP problem on the objective function $Q(\alpha)$ (Abe, 2010),

$$\begin{cases} \text{maximize} Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \\ \alpha_i \geq 0, 1 \leq i \leq N \end{cases} \quad (7)$$

If $\alpha^* = (\alpha_1^*, \alpha_2^*, ..., \alpha_N^*)^T$ is a solution of (7), then the optimal value of the parameter w is $w^* = \sum_{i=1}^{N} \alpha_i^* y_i g(x_i)$. The examples $g(x_i)$ for which $\alpha_i^* \neq 0$ are referred as support vectors. The parameter b can not be explicitly computed by solving the SVM problem, a convenient choice of the bias b being derived in terms of the support vectors and w^*. The value of the bias b is usually set to

$$b^* = -\frac{1}{2} \left\{ \max_{\substack{i \\ y_i = -1}} w^{*T} g(x_i) + \min_{\substack{i \\ y_i = 1}} w^{*T} g(x_i) \right\}. \quad (8)$$

Apparently the explicit functional expression of the feature extractor g is required in order to determine the optimal classification decision function. However, this information is not needed because, using straightforward computations, the parameters (w^*, b^*) can be expressed only in terms of the kernel K. Indeed, since $w^* = \sum_{j=1}^{N} \alpha_j^* y_j g(x_j)$, we get

$w^{*T} g(x) = \sum_{j=1}^{N} \alpha_j^* y_j g^T(x_j) g(x) = \sum_{j=1}^{N} \alpha_j^* y_j K(x_j, x)$, that is

$b^* = -\frac{1}{2} \left\{ \max_{\substack{i \\ y_i = -1}} \sum_{j=1}^{N} \alpha_j^* y_j K(x_j, x_i) + \min_{\substack{i \\ y_i = 1}} \sum_{j=1}^{N} \alpha_j^* y_j K(x_j, x_i) \right\}$ and the equation of the hyperplane $H_{b^*, w^*}(x)$ is

$$H_{b^*,w^*}(x): \sum_{j=1}^{N}\alpha_j^* y_j K(x_j,x)-\frac{1}{2}\left\{\max_{\substack{i\\ y_i=-1}}\sum_{j=1}^{N}\alpha_j^* y_j K(x_j,x_i)+\min_{\substack{i\\ y_i=1}}\sum_{j=1}^{N}\alpha_j^* y_j K(x_j,x_i)\right\}=0 \quad (9)$$

Consequently, the computation of the optimal margin classifier is reduced to solving the QP problem (7). There have been proposed a series of methods to solve (7) as for instance Sequential Minimal Optimization (SMO) (Platt, 1998), decomposition methods (Smola, Schőlkopf, 1999) and (Laskov, 2002), and methods to solve the least squares SVM formulations (Cawley, Talbot, 2002), (Keerthi, Shevade, 2003) and (Suykens, De Brabanter, Lukas, 2002) as well as software packages as SVM$^{\text{light}}$ (Joachims,1998), a new class of approaches of solving the QP optimization problem in learning SVMs being developed recently in (Steinwart, Hush, Scovel, 2011) and (Zhang, 2011). Several attempts have been proposed to design a SVM type classifier in case the training data is not linear separable yielding to the soft margin linear classifiers L1-SVM and L2-SVM (Abe, 2010), fuzzy SVM (Lin, Wang, 2002; Tsujinishi, Abe, 2003). The aim of the research reported in the next section of the paper is to investigate the performance of new refined variants of gradient ascent type using kernels algorithms in learning SVMs by developing a comparative analysis against the standard SMO algorithm.

3. Heuristic Learning of the Optimal Margin Classifier

If $S_N = \left\{(x_i, y_i), x_i \in \mathbf{R}^n, y_i \in \{-1,1\}, 1 \le i \le N\right\}$ is the given finite set of labeled examples coming from the classes h_1, h_2, for simplicity sake we assume that the first *m* examples come from the first class (that is their labels are 1) and the next $N-m$ examples come from the second class (all of them are labeled by -1). The entries of the gradient $\nabla_\alpha Q(\alpha)$ and the Hessian matrix $H(Q(\alpha)) = \left\| \frac{\partial^2 Q(\alpha)}{\partial\alpha_k \partial\alpha_p} \right\|_{k,p}$ of the objective function in the QP-problem (7) are,

$$(\frac{\partial Q(\alpha)}{\partial \alpha_k} = 1 - y_k \sum_{i=1}^{N} \alpha_i y_i K(x_k, x_i), \quad 1 \le i \le N, \qquad (10)$$

and

$$\frac{\partial^2 Q(\alpha)}{\partial\alpha_k \partial\alpha_p} = -y_p y_k K(x_k, x_p) \qquad (11)$$

respectively.

Obviously, $H(Q(\alpha))$ is a negative semi-defined matrix.

Let $\rho > 0$ be a fixed learning rate.

Since the standard updating rule

$$\alpha = \alpha^{old} + \rho \nabla_{\alpha} Q(\alpha)\Big|_{\alpha=\alpha^{old}} \tag{12}$$

in gradient ascent approach does not generally yield to a parameter that satisfies the constraint $\sum_{i=1}^{N} \alpha_i y_i = 0$, a modified learning rule of gradient ascent should be considered instead.

Briefly, in (Cocianu, State, Vlamos, 2011), we proposed that the updating step should be performed as follows.

Assume that p_1, p_2 are the components of the current parameter vector α^{old} selected for being updated, $1 \le p_1 \le m, \quad m+1 \le p_2 \le N$. If ρ_1 is a weighting parameter expressing the relative "influence" of $\frac{\partial Q(\alpha)}{\partial \alpha_{p_1}}\Big|_{\alpha=\alpha^{old}}$ and $\frac{\partial Q(\alpha)}{\partial \alpha_{p_2}}\Big|_{\alpha=\alpha^{old}}$ to the direction of the updating displacement, then the updated parameter satisfying the constraint $\sum_{i=1}^{N} \alpha_i y_i = 0$ is $\alpha = (\alpha_1, \alpha_2, ..., \alpha_N)^T$, where:

$$\alpha_i = \alpha_i^{old}, \ 1 \le i \le N, i \ne p_1, p_2 \tag{13}$$

$$\alpha_{p_i} = \alpha_{p_i}^{old} + \rho \left[\rho_1 \frac{\partial Q(\alpha)}{\partial \alpha_{p_1}}\Big|_{\alpha=\alpha^{old}} + (1-\rho_1) \frac{\partial Q(\alpha)}{\partial \alpha_{p_2}}\Big|_{\alpha=\alpha^{old}} \right], \quad i = 1,2,$$

$$0 \le \rho_1 \le 1 \tag{14}$$

Consequently, the indices p_1, p_2 should be selected such that to maximize the difference $Q(\alpha) - Q(\alpha^{old})$. Using the first order approximations, the indices p_1, p_2 should be determined such that the following conditions hold (Cocianu, State, Vlamos, 2011),

$$1 \le p_1 \le m, \quad m+1 \le p_2 \le N, \tag{15}$$

$$\rho \left[\rho_1 \frac{\partial Q(\alpha)}{\partial \alpha_{p_1}}\Big|_{\alpha=\alpha^{old}} + (1-\rho_1) \frac{\partial Q(\alpha)}{\partial \alpha_{p_2}}\Big|_{\alpha=\alpha^{old}} \right] \left(\frac{\partial Q(\alpha)}{\partial \alpha_{p_1}}\Big|_{\alpha=\alpha^{old}} + \frac{\partial Q(\alpha)}{\partial \alpha_{p_2}}\Big|_{\alpha=\alpha^{old}} \right) > 0$$

and

$Q(\alpha) - Q(\alpha^{old})$ is maximized.

The search process is over when at least one of the conditions $\rho\left[\rho_1 \frac{\partial Q(\alpha)}{\partial \alpha_{p_1}}\Big|_{\alpha=\alpha^{old}} + (1-\rho_1)\frac{\partial Q(\alpha)}{\partial \alpha_{p_2}}\Big|_{\alpha=\alpha^{old}}\right]\left(\frac{\partial Q(\alpha)}{\partial \alpha_{p_1}}\Big|_{\alpha=\alpha^{old}} + \frac{\partial Q(\alpha)}{\partial \alpha_{p_2}}\Big|_{\alpha=\alpha^{old}}\right) \le 0$ for all $1 \le p_1 \le m, \quad m+1 \le p_2 \le N$, and $\left|Q(\alpha) - Q\left(\alpha^{old}\right)\right| < \varepsilon$, respectively holds, where $\varepsilon > 0$ is a given threshold, in each case the current value of the parameter α being taken as an approximation of a local maximum point of the objective function.

4. A refined gradient ascent search method

The learning rate ρ and the weight ρ_1 should be taken such that the search process is optimized from both point of views, accuracy and efficiency. In order to obtain good approximations of the maxima values of the objective function, small values of the learning rate are recommended, for instance in our tests we used $\rho \in \left[10^{-6}, 10^{-2}\right]$.

The weight parameter ρ_1 should be taken as to include some information about the statistical properties of the subsets of examples coming from the two classes, as for instance class variability and inter-class distance.

The expression of ρ_1 proposed in our work includes the information contained by the first and the second order sample moments computed in the unknown feature space **F**. Let us denote by $\hat{\mu}_{g,1}, \hat{\mu}_{g,2}$ the sample means and $\hat{\Sigma}_{g,1}, \hat{\Sigma}_{g,2}$ the sample covariance matrices. We denote by *K* the kernel generated by *g*, that is $K(x,x') = g(x)^T g(x')$.Since we assumed that the first *m* examples come from the first class and the next $N-m$ examples come from the second class, we get,

$$\hat{\mu}_{g,1} = \frac{1}{m}\sum_{i=1}^{m} g(x_i),\ \hat{\mu}_{g,2} = \frac{1}{N-m}\sum_{i=1}^{N-m} g(x_{m+i})$$

$$\hat{\Sigma}_{g,1} = \frac{1}{m-1}\sum_{i=1}^{m}\left(g(x_i) - \hat{\mu}_{g,1}\right)\left(g(x_i) - \hat{\mu}_{g,1}\right)^T,$$

$$\hat{\Sigma}_{g,2} = \frac{1}{N-m-1}\sum_{i=1}^{N-m}\left(g(x_{m+i}) - \hat{\mu}_{g,2}\right)\left(g(x_{m+i}) - \hat{\mu}_{g,2}\right)^T.$$

$$\theta_{g,i} = \frac{\left(\hat{\mu}_{g,i}^T \hat{\mu}_{g,i}\right)^2}{\hat{\mu}_{g,i}^T \hat{\Sigma}_{g,i} \hat{\mu}_{g,i}},\ i = 1,2 \qquad (16)$$

Let the weight parameter ρ_1 be defined as:

$$\rho_1 = \frac{\theta_{g,2}}{\theta_{g,1} + \theta_{g,2}} \tag{17}$$

Lemma. The expression of ρ_1 in terms of the kernel K is:

$$\rho_1 = \frac{(N-m)^2(N-m-1)\dfrac{\left\|\hat{\mu}_{g,2}\right\|^4}{\sum_{k=1}^{N-m}(S(k))^2}}{m^2(m-1)\dfrac{\left\|\hat{\mu}_{g,1}\right\|^4}{\sum_{k=1}^{m}(T(k))^2} + (N-m)^2(N-m-1)\dfrac{\left\|\hat{\mu}_{g,2}\right\|^4}{\sum_{k=1}^{N-m}(S(k))^2}}, \tag{18}$$

where

$$T(k) = \sum_{i=1}^{m}\left(K(x_i, x_k) - \frac{1}{m}\sum_{p=1}^{m}K(x_i, x_p)\right) \text{ and}$$

$$S(k) = \sum_{i=1}^{N-m}\left(K(x_{m+i}, x_{m+k}) - \frac{1}{N-m}\sum_{p=1}^{N-m}K(x_{m+i}, x_{m+p})\right)$$

Proof. Using straightforward computation, we get,

$$\hat{\mu}_{g,1}^T\hat{\Sigma}_{g,1} = \left[\frac{1}{m}\sum_{i=1}^{m}(g(x_i))^T\right]\left[\frac{1}{m-1}\sum_{k=1}^{m}\left(g(x_k) - \frac{1}{m}\sum_{p=1}^{m}g(x_p)\right)\left(g(x_k) - \hat{\mu}_{g,1}\right)^T\right] =$$

$$= \frac{1}{m(m-1)}\sum_{k=1}^{m}\left(\sum_{i=1}^{m}\left(K(x_i, x_k) - \frac{1}{m}\sum_{p=1}^{m}K(x_i, x_p)\right)\right)\left(g(x_k) - \hat{\mu}_{g,1}\right)^T$$

Also,

$$\hat{\mu}_{g,1}^T\hat{\Sigma}_{g,1}\hat{\mu}_{g,1} = \frac{1}{m(m-1)}\sum_{k=1}^{m}T(k)\left(g(x_k) - \hat{\mu}_{g,1}\right)^T\hat{\mu}_{g,1} =$$

$$= \frac{1}{m(m-1)}\sum_{k=1}^{m}T(k)\left((g(x_k))^T\hat{\mu}_{g,1} - \left\|\hat{\mu}_{g,1}\right\|^2\right)$$

and,

$$(g(x_k))^T\hat{\mu}_{g,1} = \frac{1}{m}\sum_{i=1}^{m}(g(x_k))^T g(x_i) = \frac{1}{m}\sum_{i=1}^{m}K(x_k, x_i)$$

hence,

$$\hat{\mu}_{g,1}^T \hat{\Sigma}_{g,1} \hat{\mu}_{g,1} = \frac{1}{m(m-1)} \sum_{k=1}^{m} T(k) \frac{1}{m} \left(\sum_{i=1}^{m} K(x_k, x_i) - \frac{1}{m} \sum_{i=1}^{m} \sum_{l=1}^{m} K(x_i, x_l) \right) =$$

$$= \frac{1}{m(m-1)} \sum_{k=1}^{m} T(k) \frac{1}{m} \left(\sum_{i=1}^{m} \left(K(x_i, x_k) - \frac{1}{m} \sum_{l=1}^{m} K(x_i, x_l) \right) \right) =$$

$$= \frac{1}{m^2(m-1)} \sum_{k=1}^{m} (T(k))^2,$$

Similarly, we get

$$\hat{\mu}_{g,2}^T \hat{\Sigma}_{g,2} \hat{\mu}_{g,2} = \frac{1}{(N-m)^2(N-m-1)} \sum_{k=1}^{N-m} (S(k))^2 \text{ , where}$$

$$S(k) = \sum_{i=1}^{N-m} \left(K(x_{m+i}, x_{m+k}) - \frac{1}{N-m} \sum_{p=1}^{N-m} K(x_{m+i}, x_{m+p}) \right).$$

Note that the weight coefficient ρ_1 can be also expressed as,

$$\rho_1 = \frac{\left\| \hat{\mu}_{g,2} \right\|^4}{\frac{\hat{\mu}_{g,2}^T \hat{\Sigma}_{g,2} \hat{\mu}_{g,2}}{\hat{\mu}_{g,1}^T \hat{\Sigma}_{g,1} \hat{\mu}_{g,1}} \left\| \hat{\mu}_{g,1} \right\|^4 + \left\| \hat{\mu}_{g,2} \right\|^4},$$

that is when $n = 1$, ρ_1 is the Fisher coefficient,

$$\rho_1 = \frac{\hat{\mu}_{g,2}{}^4}{\frac{\hat{\mu}_{g,2}^2 \sigma_{g,2}^2}{\hat{\mu}_{g,1}^2 \sigma_{g,1}^2} \hat{\mu}_{g,1}{}^4 + \hat{\mu}_{g,2}{}^4} = \frac{\hat{\mu}_{g,2}{}^2}{\frac{\sigma_{g,2}^2}{\sigma_{g,1}^2} \hat{\mu}_{g,1}{}^2 + \hat{\mu}_{g,2}{}^2} = \frac{\frac{\hat{\mu}_{g,2}{}^2}{\sigma_{g,2}^2}}{\frac{\hat{\mu}_{g,1}{}^2}{\sigma_{g,1}^2} + \frac{\hat{\mu}_{g,2}{}^2}{\sigma_{g,2}^2}}$$

Being given the lack of theoretical arguments to justify conclusions about the performance of the proposed method, an experimental analysis should be developed instead. Our tests aimed to establish on experimental basis conclusions concerning:

1. The dependency of the number of iterations on the parameter γ, required to obtain significant accuracy. The comparative analysis involved the standard SMO method (Platt, 1998), the gradient ascent in the initial space and the modified

gradient ascent algorithm in the feature space using the weight coefficient ρ_1 , both methods being applied in the feature space.

2. The performance analysis of the resulted classifier expressed in terms of the recognition rates.

The evaluation of distances between two normal n-dimensional classes $N(\mu_i, \Sigma_i), i = 1,2$ is performed in terms of the Mahalanobis distance,

$$d(h_1, h_2) = (\mu_1 - \mu_2)^T (\Sigma_1 + \Sigma_2)^{-1} (\mu_1 - \mu_2).$$

In case of two Bernoullian samples $S^{(1)}, S^{(2)}$ of sizes N_1, N_2 coming from these classes, the distance between samples is evaluated two ways, the former being the sample Mahalanobis distance,

$\hat{d}_{N_1,N_2}(h_1, h_2) = (\hat{\mu}_1 - \hat{\mu}_2)^T (\hat{\Sigma}_1 + \hat{\Sigma}_2)^{-1} (\hat{\mu}_1 - \hat{\mu}_2)$, where $\hat{\mu}_i, \hat{\Sigma}_i$ are the sample mean and sample covariance matrix respectively, and the latter being the minimum value of the distances between pairs of examples coming from different classes,

$$\tilde{d}_{N_1,N_2}(h_1, h_2) = \min_{\substack{x \in S^{(1)} \\ y \in S^{(2)}}} \|x - y\|$$

The variability of a sample coming from a certain class can be numerically expressed many ways. We considered the indicator given by the mean distances between the feature vectors representing examples coming from that class to quantitatively express the variability within the sample. If C is the subset of S_N containing the examples coming from one of the labeled classes, then the measure of the variability of C is,

$$\text{var}(C) = \frac{1}{2|C|^2} \sum_{x_i, x_j \in C} \|g(x_i) - g(x_j)\| \qquad (19)$$

that can be expressed in terms of the kernel function as (State, Cocianu, Mircea, 2012),

$$\text{var}(C) = \frac{1}{2|C|^2} \sum_{x_i, x_j \in C} \sqrt{K(x_i, x_i) - 2K(x_i, x_j) + K(x_j, x_j)} \qquad (20)$$

Let C_+, C_- be the subsets of S_N containing the examples labeled by 1 and -1

respectively. We express the overall variability of S_N by ,

$$ind = \frac{\min\{\text{var}(C_+), \text{var}(C_-)\}}{\max\{\text{var}(C_+), \text{var}(C_-)\}} \qquad (21)$$

Let $b_{\max}=\begin{cases} b_+, & \text{var}(C_+)\geq \text{var}(C_-) \\ b_-, & \text{otherwise} \end{cases}$, $b_{\min}=\begin{cases} b_-, & \text{var}(C_+)\geq \text{var}(C_-) \\ b_+, & \text{otherwise} \end{cases}$.

Based on the idea that the classification of new samples in the class maximum variability should be "encouraged", we take the bias $\hat{b}$ as (State, Cocianu, Mircea, 2012)

$$\hat{b}=\begin{cases} ind * b_{\max}+(1-ind)* b_{\min}, ind\leq 0.5 \\ (1-ind)* b_{\max}+ind * b_{\min}, ind>0.5 \end{cases} \quad (22)$$

5. Experimental analysis of the performance of the learning scheme based on the weight ρ_1 in case of Gaussian kernel

The tests were performed on simulated data coming from two-dimensional normal distributions, the feature extractor *g* corresponding to the RBF kernel $K(x,x')=\exp\{-\gamma\|x-x'\|^2\}$, $\gamma>0$, aiming to iteratively solve the QP-problem (7). In our tests the generated data for the learning phase were linearly separable.

The evaluation of the empirical recognition rates of the resulted classifier was performed on non-linearly separable new test data randomly generated from the same classes.

The tests were performed according to the scheme presented in the following. The classes correspond to two dimensional Gaussian repartitions $N(\mu_i,\Sigma_i), i=1,2$, where

$\mu_1=(2 \quad 0.5)^T$, $\mu_2=(1 \quad -0.75)^T$, $\Sigma_1=\begin{bmatrix} 0.063 & -0.0007 \\ -0.0007 & 0.1646 \end{bmatrix}$

$\Sigma_2=\begin{bmatrix} 0.0666 & 0.017 \\ 0.017 & 0.1063 \end{bmatrix}$ The sample of learning data consisted of 60 and 70 examples coming from these classes respectively. The Mahalanobis distance between the classes is $d(h_1,h_2)=12.3844$, the empirical Mahalanobis distance computed for the learning data is $\hat{d}_{m,N-m}(h_1,h_2)=12.334$, and the minimum distance between examples coming from different classes is $\tilde{d}_{m,N-m}(h_1,h_2)=0.3363$. The learning data and the hyperplane computed according to the proposed method are presented in Figure 1.

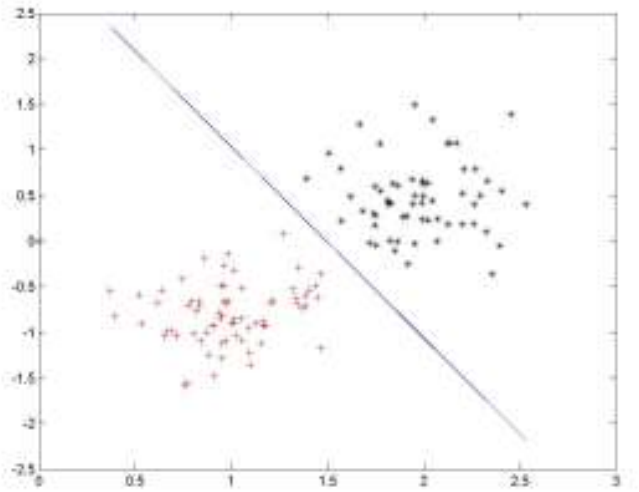

Figure 1. The learning data and the hyperplane computed according to the proposed method

The solution of the QP problem (7) was adaptively learned both ways, using the standard SMO algorithm and the proposed variant of gradient ascent method for different values of the parameter γ, and some of the results being reported in Table 2.

Table 2. Comparative analysis of the proposed method to the standard SMO

γ	The number of iterations		Recognition rates for the first class		Recognition rates for the second class	
	Standard SMO algorithm	The proposed variant of gradient ascent method	Standard SMO algorithm	The proposed variant of gradient ascent method	Standard SMO algorithm	The proposed variant of gradient ascent method
0.15	159	194	0.9996	0.9872	0.9134	0.9912
0.3	175	103	0.9992	0.9878	0.9254	0.9932
0.5	216	65	0.9986	0.9886	0.9448	0.9938
0.8	242	46	0.999	0.9892	0.9652	0.9938
1	274	41	0.9984	0.987	0.9744	0.993
1.2	314	37	0.998	0.99	0.9806	0.9956
1.4	328	35	0.9944	0.9894	0.9836	0.993

The variation of the required number of iterations on the parameter γ in case the solution of the QP problem (7) is computed according to the standard SMO algorithm and the proposed method respectively is shown in Figure 3.

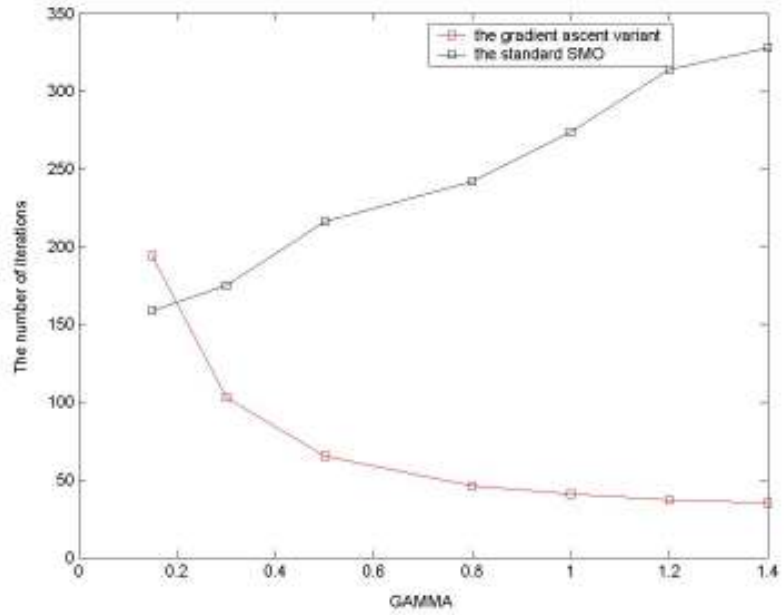

Figure 3. The dependency of the number of iterations on the parameter γ

The performance of the resulted classifier was empirically evaluated on new non-linearly separable test data containing 5000 examples coming from each class, where the bias b is given by (8) and (22) respectively. The results are represented in Figure 4 and Figure 5. In Figure 4, the values of the recognition rates are shown for different values of the parameter γ and the two choices of the bias b, where the hyperplane was computed according to the proposed method. In Figure 5 similar results are shown in case the hyperplane was computed by the standard SMO algorithm.

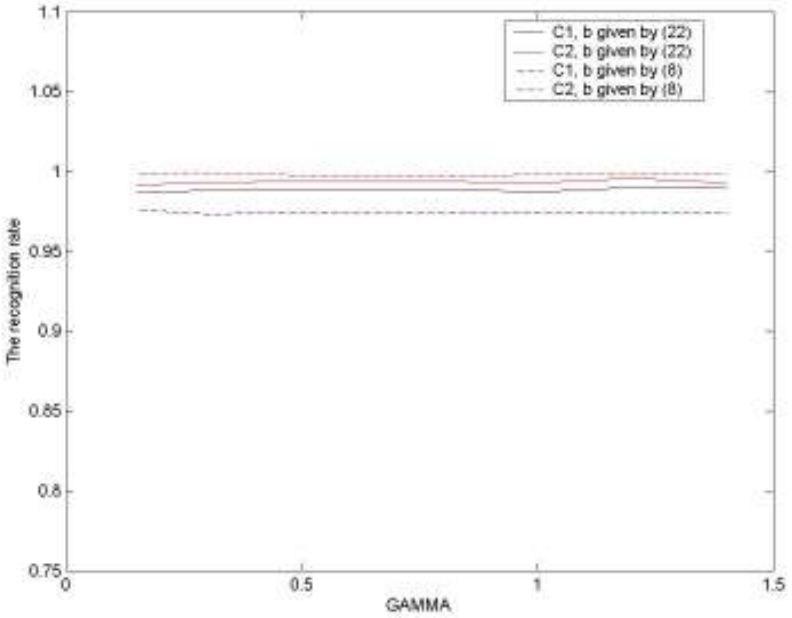

Figure 4. The recognition rates for the classifier computed according to the proposed method

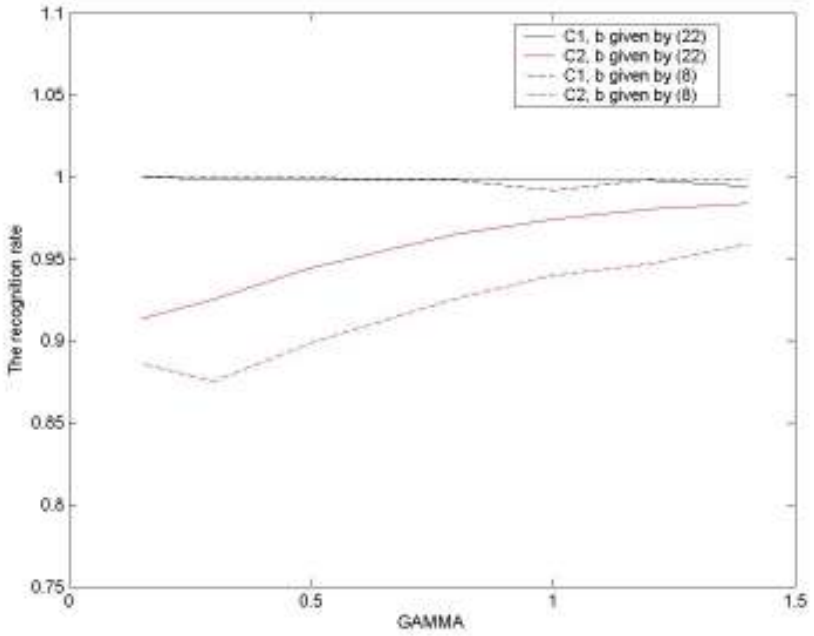

Figure 5. The recognition rates for the classifier computed by the standard SMO

The tests entailed a series of conclusions that are briefly presented in the following.

a) Concerning the dependency of the number of iterations on the parameter γ in case of the standard SMO method as compared to the proposed method, and comparable accuracy in terms of the values of the objective function, we conclude:

1. Significant improvement is obtained in case of the proposed method for values of $\gamma \geq 0.5$, while the standard SMO seems to perform better for smaller values of γ.
2. The proposed method computes an approximation of the solution of (7) significantly faster than the standard SMO algorithm, at comparable accuracy expressed in terms of the values of the objective function.

b) Concerning the performance of the resulted classifier expressed in terms of the recognition rates, we conclude

1. The recognition rates are comparable for both classifiers. It seems that in case of the proposed method the values of the recognition rates corresponding to the classes are closer as compared to the classifier computed by standard SMO algorithm, where the value of the recognition rate corresponding to one class is significantly larger than the second one. A possible explanation of this effect could be related to the fact that the test data coming from the classes are of different variability.
2. The performance of the classifier computed by the proposed method is not influenced by the values of γ, while in case of the standard SMO it seems that better recognition rate is obtained for larger values of γ.

6. Conclusions

The research reported in the paper proposes a modified gradient ascent method for solving the dual QP problem of nonlinear SVM. The proposed method uses the weight parameter ρ_1 to tune the direction of the search displacement to the particular training sequence and the computation rule of the bias based on class variability indices. The method is not mathematically founded, therefore conclusions on its performance can be derived on experimental basis only. The tests pointed out good convergence properties and, moreover, the proposed modified variants proved higher convergence rates as compared to the standard SMO algorithm. Also, the performance of the resulted classifier proves better as compared to the standard choice of the bias.

REFERENCES

[1] **Abe, S.(2010),** ***Support Vector Machines for Pattern Classification***, *Springer;*

[2] **Cawley, G.C., Talbot, N.L.C. (2002),** ***Improved Sparse Least Squares Support Vector Machines***. *Neurocomputing* 48 (1-4);

[3] **Cocianu, C ., State, L., Vlamos, P.(2011),** ***A New Method for Learning the Support Vector Machines****;* Proceedings of the 6th International Conference on Software and Data Technology, ICSOFT 2011, INSTICC Press, pp. 365-370;

[4] **Keerthi, S.S., Shevade S.K. (2003),** ***SMO Algorithm for Least Squares SVM Formulations****. Neural Compt.* 15 (2);

[5] **Joachims, T. (1998),** ***Making Large-Scale SVM Learning Practical***, in Schőlkopf, B., Burges, C.J., Smola, A.J. eds. *Advances in Kernel Methods – Support Vector Learning*, Cambridge, MA, MIT Press;

[6] **Laskov, P. (2002),** ***An Improved Decomposition Algorithm for Regression Support Vector Machines***. *Mach. Learning*, 46;

[7] **Lin C.F., Wang S.D. (2002),** ***Fuzzy Support Vector Machines***; *Transactions on Neural Networks,* Vol. 13, pp. 464-471;

[8] **Platt, J. (1998),** ***Fast Training of Support Vector Machines Using Sequential Minimal Optimization***. In: *Advances in Kernel Methods – Support Vector Learning*, Cambridge, MA, MIT Press. (Schőlkopf, B., Burges, C.J., Smola, A.J. eds.);

[9] **Smola, A.J., Schőlkopf, B., Müller, K.-R. (1998),** ***General Cost Functions for Support Vector Regression***. In: *Proc. of the Ninth Australian Conf. in Neural Networks*, Brisbane, Australia;

[10] **State, L.,Cocianu, C., Mircea, M.(2012),** ***Heuristic Attempts to Improve the Generalization Capacities in Learning SVMs*** ; Proceedings of 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2012), August, Kyoto, Japan, IEEE Computer Society Press, in press;

[11] **State, L., Cocianu, C., Vlamos, P.(2008),** ***A New Unsupervized Learning Scheme to Classify Data of Relative Small Volume***; Economic Computation and Economic Cybernetics Studies and Research, *ASE Publishing*, ISSN 0424-267X;

[12] **Steinwart I., Hush D., Scovel C. (2011),** ***Training SVMs Without Offset***; *Journal of Machine Learning Research* 12 pp. 141-202;

[13] **Suykens, J.A.K., De Brabanter J., Lukas, L. (2002),** ***Weighted Least Squares Support Vector Machines: Robustness and Sparse Approximation***. In: *Neurocomputing special issue;*

[14] **Shawe-Taylor, J., Cristianini, N. (2000),** ***Support Vector Machines and Other Kernel-based Learning Methods****;* Cambridge University Press;

[15] **Tsujinishi D, Abe S.(2003),** ***Fuzzy Least Squares Support Vector Machines for Multiclass Problems****; Neural Networks*; pp. 785-792 ;

[16] **Vapnik, V. (1998),** ***Statistical Learning Theory***. John Wiley, N.Y.;

[17] **J. Zhang (2011),** ***Optimization of Kernel Function Parameters SVM Based on the GA*** .Advance Materials Research, Jan. pp. 4124-4128, doi: 10.4028/www.scientific.net/AMR.433-440.4124.