A Fast and Efficient Algorithm for Multiclass Support Vector Machines Classifier

Emad A. El-Sebakhy
Assistant Professor of Computer Science
Department Mathematics, Computer Science & Statistics
State University of New York, College at Oneonta
Tel: *(607)-255-2748*, email: *eae6@cornell.edu*
Oneonta, NY 13820

Abstract

Recently, support vector machines (SVMs) have been proposed in dealing with the statistical pattern recognition within the framework of statistical learning theory and structural risk minimization. The idea of applying SVMs multiclass classifier is still an ongoing research issue, especially when we deal with the large database, the learning algorithm becomes computationally expense and NP hard problem. In this paper we propose the conjugate gradient technique as an efficient algorithm to handle the complexity and running time for the multiclass least squares SVMs classifier. We explained SVM classifier, how it can be used, and introduced the most common kernel functions in the literatures. The performance of the new technique in both simulation study and the real-world applications is proposed and compare its result with the old strategies. We test the performance of the recent SVMs classifier using the common applications: Fisher Iris data, Fraud Detection, and Bioinformatics. The performance of the new SVM classifier technique turns to be robust and efficient with respect to the parameters of the algorithm.
Keywords: Pattern Classification; Kernel function; Neural networks; Radial Basis function network; Support vector machine; Statistical learning theory; Bioinformatics.

1 The Pattern Classification Meaning

Pattern classification is defined as a mapping from the matrix of predictor variables $\mathbf{X} \in \mathbb{R}^p$ to a vector of class categories, $\mathbf{Y} \subseteq \mathbb{R}$. This means that each pattern is given in terms of the independent predictors $X_1, \ldots, X_p$, and the goal is to establish a rule that separates predictors belonging to different groups, $A_1, A_2, \ldots, A_c$. Without loss of generality, we can write the classes $\{A_1, A_2, \ldots, A_c\}$ as $\{0, 1, \ldots, c-1\}$ and $X_j = (x_{1j}, \ldots, x_{nj})^T$, for all $j = 1, 2, \ldots, p$ as the j^{th} predictor variable for more convenience.
We use $D = \{(x_1, , \ldots, x_p, y)\}$ as the given training set, and the lower case letters $x_{i1}, x_{i2}, \ldots, x_{ip}$, for $i = 1, \ldots, n$ to refer to the values of each observation of the feature variables, and $y = k - 1$ to the response variable Y (class A_k), for $k = 1, \ldots, c$. We use the symbol π_{ik} for the probability that observation i falls in the class A_k, that is,

$$\begin{aligned} &\pi_{ik} = P(y_i = k \mid \mathbf{x}_i);\ \ k = 0, 1, \ldots, c-1, \\ &\text{such that } \pi_{ik} \geq 0;\ \sum_{k=0}^{c-1} \pi_{ik} = 1;\ \forall i = 1, \ldots, n. \end{aligned} \tag{1}$$

In most recent years, dealing with the pattern classification problem in high dimension is not an easy task. The pattern recognition is critical in most human decision-making "the more relevant patterns at your

prediction, the better your decision will be", Ross (1998). The following are some of the common examples that can be formulated as a pattern classification problems:

1. *Intelligent Searching*: An internet search engine that provides the most relevant content and banner advertisements based on the users' past behavior.

2. *Fraud Detection*: Detect fraudulent credit card transactions and automatically decline the charge.

3. *Bioinformatics*: Predicting gene function from expression profiles

4. *Intelligent Searching*: An internet search engine that provides the most relevant content and banner advertisements based on the users' past behavior.

2 Literature Review

Researchers started looking for some techniques that deal with the pattern classification problem, such as the classical statistics techniques (*Disciminant Analysis*; *Logistic Regression*), Hosmer (2000); Artificial Neural Networks (*Multilayer Feedforward Neural Network*; *Radial Basis Function Networks*; *Probabilistic Network*); Machine Learning (*Support Vector Machine*; *Decision Trees*; *K-Nearest Neighbor*); and Data Mining. However, the method of classification depends on the nature of the data and the question of interest. Many developments and new materials are found in Gordon (1981, 1999), and the references therein.
Last two decades, the quick advance in information processing systems had directed both science and engineering research towards the development of intelligent systems that can develop models of natural phenomena automatically, Abbott (1997). In this respect, a wide range of machine learning techniques like decision trees (Quinlan 1986, 1992), logistic regression, discriminant analysis, artificial neural networks, radial basis function network, Bayesian Network, reinforcement learning, Watkins and Dyan (1992) and and Duda et al. (2001). An important development in these areas was proposed by Vapnik and Chervonenskis (1974), their results were extended and generalized to provide the subject currently known as statistical learning theory, which serves as a basis of support vector machine (SVM) technique, further details can be found in Vapnik (1995 and 1998).
Recently, support vector methods have been used in a variety of applications, such as, the statistical pattern recognition, estimation and time series prediction. The SVM classification methodology and theory is well documented and details can be found in a number of research papers, such as Gunn (1997); Burges (1998); Luntz et al. (1969); Burges (1998); Cristianini et al. (2000); Scholkopf et al. (1995); Raudys (2000); Burbidge et al. (2001); Jack and Nandi, 2002; Bao and Sun (2002); and Suykens et al. (2001).
In the machine learning literatures, there are several methods proposed to construct a multi-class classifier by combining several binary SVM classifiers. As it is computationally more expensive to solve multi-class problems, comparisons of these methods using large-scale problems have discussed in Hsu and Lin (2002), such as "*all-together*" methods, other methods based on binary classifications: "*one-against-all*","*one-against-one*", and "*directed acyclic graph SVM*" (DAGSVM). Their experiments indicate that the "*one-against-one*" and DAGSVM methods are more suitable for practical use than the other methods.
The structure of the paper can be summarized as: Section (1) proposes the pattern classification meaning, and its importance. In section (2), we present the background and the literature review of both pattern classification and SVM. Section (3) proposes the binary SVM classifier for both linear and non-linear separable data. In section (4) we present the new large scale conjugate gradient algorithm to handle the computational expense and running time complexity. Section (5) presents Multiclass Least squares SVMs Classifier. Section (6) gives a briefly study about the performance of the recent technique using both simulation study and some of the real-world applications in both computer science and engineering. We draw our conclusion and future work in section (7).

3 The Binary SVM Classifier

The basic idea in support vector machine is to map the inputs $\mathbf{x}$ to vectors $\varphi(\mathbf{x})$ in some high-dimensional feature space, Vapnik (1995, 1998). In this section, we present the the binary SVM classifier for both linear and non-linear separable data.

3.1 The Binary SVM Classifier for Linear Separable Data

The hyperplane in the p dimensional space is determined by the pair (ω, b), where ω is a p dimensional vector orthogonal to the hyperplane and b is the offset constant. The hyperplane $(\omega \cdot \mathbf{x}) + b$ separates the given training set if and only if

$$(\omega \cdot \mathbf{x}) + b > 0; \quad \text{if } y = 1; \quad \text{and } (\omega \cdot \mathbf{x}) + b < 0; \quad \text{if } y = -1.$$

If we additionally require that ω and b such that point closest to the hyperplane has a distance $\frac{1}{\|\omega\|}$, then we have

$$\mathbf{y}_i \left(\omega^T \mathbf{x}_i + b\right) \geq +1; \quad \text{for } i = 1, 2, \ldots, n.$$

To find the optimal separating hyperplane, we have to find the hyperplane that maximizes the minimum distance between the hyperplane and any sample of the training set in hand. Consider the example for a two-dimensional input space in Figure 3.1(a). There are many possible linear classifiers that can separate the data, but there is only one that maximizes the distance between it and the nearest data point of each class. The distance of a point $\mathbf{x}_s$ from the above hyperplane $d(\omega, b; \mathbf{x}_s)$ is defined as:

$$d(\omega, b; \mathbf{x}_s) = \frac{\mid \omega^T \mathbf{x}_s + b \mid}{\|\omega\|}.$$

Therefore, the optimal hyperplane is then given by maximizing the margin, subject to the constraints $\mathbf{y}_i \left(\omega^T \mathbf{x}_i + b\right) \geq +1$. As in Figure 3.1 (b), this margin can be computed as:

$$\rho(\omega, b) = \overset{min}{\{\mathbf{x}_i; \mathbf{y}_i = 1\}} d(\omega, b; \mathbf{x}) + \overset{min}{\{\mathbf{x}_i; \mathbf{y}_i = -1\}} d(\omega, b; \mathbf{x}) = \frac{2}{\|\omega\|}.$$

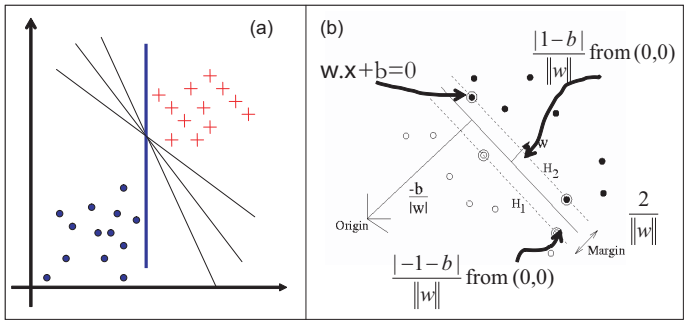

Figure 1: (a) Optimal Linear Separating Hyperplane in a 2–dimensional space, (b) A separable classification hyperplane.

Therefore, the hyperplane that optimally separate the data is the one that minimizes the function $F(\omega) =$

Top

$\frac{\omega^T \omega}{2}$, subject to the constraints: $\mathbf{y}_i \left(\omega^T \mathbf{x}_i + b\right) \geq +1$; for $i = 1, \ldots, n$, that is,

$$\begin{array}{l} \text{Min } \{\dfrac{\|\omega\|^2}{2}\}, \\ \text{subject to} \\ \mathbf{y}_i \left(\omega^T \mathbf{x}_i + b\right) \geq +1; \quad i = 1, 2, \ldots, n. \end{array} \tag{2}$$

The corresponding Lagrange function is

$$L(\omega, b, \lambda) = \frac{\|\omega\|^2}{2} + \sum_{i=1}^{n} \lambda_i \{\mathbf{y}_i \left(\omega^T \mathbf{x}_i + b\right) - 1\}, \tag{3}$$

where the λ_i are the Lagrange multipliers. The Lagrangian has to be minimized with respect to ω and b and maximized with respect to $\lambda_i \geq 0$. The Kuhn, Tucker (1961) conditions are:

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \lambda_i^0 \, \mathbf{y}_i = 0; \;\; \frac{\partial L}{\partial \omega} = 0 \Rightarrow \omega_0 = \sum_{i=1}^{n} \lambda_i^0 \, \mathbf{y}_i \, \mathbf{x}_i.$$

In the practical application, we usually require a quadratic programming optimization problem (dual problem). Since the support vectors are the vectors that have non-zero coefficients λ_0 in the expression of ω_0, we substitute for ω in the Lagrangian function, and after further simplification, then we obtain the dual form of the function to be optimized, that is, we optimize the following problem:

$$\begin{array}{l} \text{Min } \{\sum\limits_{i=1}^{n} \lambda_i - \dfrac{1}{2} \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} \lambda_i \, \lambda_j \, \mathbf{y}_i \, \mathbf{y}_j \, \mathbf{x}_i^T \, \mathbf{x}_j\}, \\ \text{subject to} \\ \sum\limits_{j=1}^{n} \lambda_i \, \mathbf{y}_i = 0; \text{ and } \lambda_i \geq 0; \quad i = 1, 2, \ldots, n. \end{array} \tag{4}$$

Once the solution has been found in the form of a vector $\lambda^0 = (\lambda_1^0, \lambda_2^0, \ldots, \lambda_n^0)$, the optimal separating hyperplane is found, the decision can be build according to the sign of the following function:

$$f(\mathbf{x}) = \text{ sign } (\omega_0^T \mathbf{x} + b_0) = \text{ sign } (\sum_{i=1}^{n} \lambda_i^0 \, \mathbf{y}_i \, \mathbf{x}_i^T \mathbf{x} + b_0).$$

3.2 The Binary SVM Classifier for Non-Linear Separable Data

If the problem is not linearly separable, then *slack* variables $\eta_i \geq 0$ are introduced, and the new constraint can be written as

$$\mathbf{y}_i \left(\omega^T \varphi(\mathbf{x}_i) + b\right) \geq 1 - \eta_i; \;\; i = 1, \ldots, n.$$

The number of *slack* variables allowed can be controlled; a penalty term $\frac{C}{\mu} \sum_{i=1}^{n} \eta_i^{\mu}$ is added to the objective function, where C, $\mu > 0$ are called the penalty coefficients, Cristiani (2000). The common values of the parameter μ are 1 and 2, giving linear and quadratic slack penalties, respectively. The binary SVM classification problem with the quadratic slack penalty becomes

$$\begin{array}{l} \text{Min } J(\omega, b, \eta_i) = \dfrac{1}{2} \, \omega^T \, \omega + \dfrac{C}{2} \sum\limits_{i=1}^{n} \eta_i^2, \\ \text{subject to} \\ \mathbf{y}_i \left(\omega^T \varphi(\mathbf{x}_i) + b\right) \geq 1 - \eta_i; \;\; \eta_i \geq 0; \;\; i = 1, 2, \ldots, n. \end{array} \tag{5}$$

Top

Suykens and Vandewalle (1999a) replaced the inequality constraints by equality constraints, a squared error term. The corresponding Lagrangian function, $L(\omega, b, \lambda, \eta, \gamma)$ for the system (5) is

$$L = \frac{\|\omega\|^2}{2} + \frac{C}{2}\sum_{i=1}^{n}\eta_i^2 - \sum_{i=1}^{n}\lambda_i\{\mathbf{y}_i\ \left(\omega^T\,\varphi(\mathbf{x}_i) + b\right) - 1 + \eta_i\} - \sum_{i=1}^{n}\gamma_i\,\eta_i, \tag{6}$$

where $0 \leq \lambda_i \leq C$ and γ_i are the associated Lagrangian multipliers. The constant C is user-defined. It is a regularizing parameter because it determines the balance between the complexity of the network, characterized by the weight vector ω and the classification error of the data. For the normalized input signals the value of C is usually much larger than one and adjusted by trials and errors, Cristiani (2000).
In practical application, the dual quadratic optimization problem to the optimization problem (5) needed to be solved. This can be achieved by computing the Kuhn and Tucker (1961) necessary conditions:

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n}\lambda_i\,\mathbf{y}_i = 0;\ \frac{\partial L}{\partial \omega} = 0 \Rightarrow \omega = \sum_{i=1}^{n}\lambda_i,\ \mathbf{y}_i\,\varphi(\mathbf{x}_i);$$
$$\frac{\partial L}{\partial \eta_i} = 0 \Rightarrow C\,\eta_i - \lambda_i = 0;\ \frac{\partial L}{\partial \lambda_i} = 0 \Rightarrow \mathbf{y}_i\ \left(\omega^T\,\varphi(\mathbf{x}_i) + b\right) - 1 + \eta_i = 0,$$

for $i = 1, 2, \ldots, n$. This system can be written in a matrix format as, Gloub and Van Loan (1989)

$$\mathbf{y}^T\,\lambda = 0;\ \omega = \mathbf{H}^T\,\lambda;$$
$$C\,\mathbf{I}\,\eta = \lambda\,\mathbf{I};\ \ \mathbf{H}\,\omega + \mathbf{Y}\,b + \mathbf{I}\,\eta = \mathbf{1},$$

By eliminating ω and η, we obtain the system

$$\left[\begin{array}{c|c} 0 & \mathbf{Y}^T \\ \hline \mathbf{Y} & \mathbf{H}\,\mathbf{H}^T + C^{-1}\,\mathbf{I} \end{array}\right]\ \left[\begin{array}{c} b \\ \hline \lambda \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{1} \end{array}\right]. \tag{7}$$

where $\mathbf{H} = [\mathbf{y}_1\,\phi(\mathbf{x}_1), \ldots, \mathbf{y}_n\,\phi(\mathbf{x}_n)]^T$, $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^T$, $\lambda = [\lambda_1, \ldots, \lambda_n]^T$, and $\mathbf{1} = [1, \ldots, 1]^T$. Thus, we substitute for ω and η in the Lagrangian function, (6) with more simplification, the dual form of the function to be optimized can be written as

$$\begin{array}{l} \text{Min } \{\mathbf{F}(\lambda) = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\,\lambda_j\,\mathbf{y}_i\,\mathbf{y}_j\,\kappa(\mathbf{x}_i, \mathbf{x}_j)\}, \\ \text{subject to} \\ \sum_{j=1}^{n}\lambda_i\,\mathbf{y}_i = 0;\ \text{and } 0 \leq \lambda_i \leq C;\ \ i = 1, 2, \ldots, n, \end{array} \tag{8}$$

where $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\,\phi(\mathbf{x}_j)$ is symmetric positive definite kernel function (satisfies the Mercer's condition, Boser et al. (1992). Once the solution has been found, the decision can be constructed according to the sign of the following function:

$$f(\mathbf{x}) = \text{ sign } (\omega^T\varphi(\mathbf{x}) + b) = \text{ sign } (\sum_{i=1}^{n}\lambda_i\,\mathbf{y}_i\,\kappa(\mathbf{x}_i, \mathbf{x}) + b). \tag{9}$$

Figure 2 shows an example of the feature map from two dimensional space to a three-dimensional one: $\varphi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1\,x_2)$.
We note that the kernel function $\kappa(\mathbf{x}, \mathbf{y})$ for the function $\varphi(\mathbf{x})$ in Figure 2 can be computed by squaring of their inner product, that is,

$$\begin{array}{rcl} \kappa(\mathbf{x}, \mathbf{y}) & = & (\varphi(\mathbf{x}), \varphi(\mathbf{y})) \\ & = & \left((x_1^2, x_2^2, \sqrt{2}x_1\,x_2), (y_1^2, y_2^2, \sqrt{2}y_1\,y_2)\right) \\ & = & \left((x_1^2\,y_1^2, x_2^2\,y_2^2, 2x_1\,x_2\,y_1\,y_2)\right) = (\mathbf{x}, \mathbf{y})^2. \end{array} \tag{10}$$

The most common kernel functions in literatures are:

Top

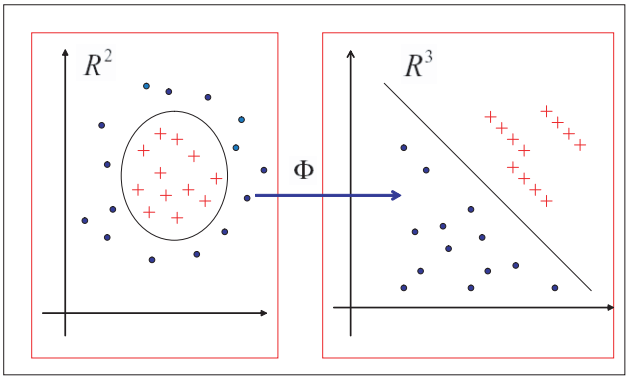

Figure 2: Optimal non-linear separating hyperplane in a 2–dimensional space

- Linear: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^T\, \mathbf{x}_j + \gamma\right)$
- Polynomial: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \left(\delta + \gamma\, \mathbf{x}_i^T\, \mathbf{x}_j\right)^q$; for $q = 1, 2, \ldots, \ldots$
- Gaussian or (Radial Basis Function): $\kappa(\mathbf{x}_i, \mathbf{x}_j) = exp\left(\dfrac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{-2\,\mathbf{\Sigma}^2}\right)$,
- Multilayer-Perceptron (Sigmoidal): $\kappa(\mathbf{x}_i, \mathbf{x}_j) = tansh\,(\gamma(\mathbf{x}_i\, \mathbf{x}_j) - \delta)$,
- Fourier Series: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \dfrac{sin(\frac{2\,n+1}{2})(\mathbf{x}_i - \mathbf{x}_j)}{sin((\frac{1}{2})(\mathbf{x}_i - \mathbf{x}_j))}$.

4 The Conjugate Gradient Algorithm

We note that for large number of observations, the matrix in (7) can not be stored, where we need an iterative solution method. A new suggestion to handle the computational complexity and running time for the system (7) using the large scale conjugate gradient algorithm, Gloub and Van Loan (1989). The conjugate gradient method for solving the system $\mathbf{AX} = \mathbf{B}$ with $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$ symmetric positive definite and $\mathbf{B} \in \mathbb{R}^n$ is given in exhibit 1. The advantage of the conjugate gradient method is that if the matrix $\mathbf{A}$ is symmetric positive definite and it can be written as $\mathbf{A} = \mathbf{I} + \tau$, where $\text{rank}(\tau) = \delta$, then the conjugate gradient algorithm in Table (1) converges in at most $\delta + 1$ steps. Now, we can write the system (7) in the form:

$$\begin{bmatrix} 0 & \mathbf{Y}^T \\ \mathbf{Y} & \Gamma \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}, \tag{11}$$

where $\beta_1 = 0$, $\beta_2 = \widetilde{1}$, $\alpha_1 == b$, $\alpha_2 = \lambda$, and $\Gamma = (\mathbf{H}\,\mathbf{H}^T + C^{-1}\, I)$. We note that the coefficient matrices in both systems (7) and (11) are not positive definite. On the other hand, we know that the conjugate gradient technique can not be applied to non-positive definite matrix. Thus, we should rewrite both coefficient matrices in (7) and (11) in a different format to be able to apply the conjugate gradient method. To get around this problem, we write each of these coefficient matrices in the systems (7) and (11) in the following equivalent systems:

$$\begin{bmatrix} \theta & 0 \\ 0 & \Gamma \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 + \Gamma^{-1}\, \mathbf{Y}\, \alpha_1 \end{bmatrix} = \begin{bmatrix} \mathbf{Y}^T\, \Gamma^{-1}\, \beta_2 - \beta_1 \\ \beta_2 \end{bmatrix}, \tag{12}$$

Top

Table 1: Conjugate Gradient Algorithm

(1.) *Begin Initialize*: $i = 0$, $x_0 = 0$, $r_0 = \mathbf{B}$
(2.) While $r_i \neq 0$, for all $i = 0, 1, \ldots, n$
  $i = i + 1$
  if $i = 1$
    $p_1 = r_0$
  else
    $\beta_i = \frac{r_{i-1}^T r_{i-1}}{r_{i-2}^T r_{i-2}}$; $p_i = r_{i-1} + \beta_i p_{i-1}$
  end
  $\lambda_i = \frac{r_{i-1}^T r_{i-1}}{p_i^T \mathbf{A} p_i}$; $x_i = x_{i-1} + \lambda_i p_i$
  $r_i = r_{i-1} - \lambda_i \mathbf{A} p_i$
(3.) end
(4.) $x = x_i$

where θ and Γ are positive definite matrices, such that $\theta = \mathbf{Y}^T \Gamma^{-1} \mathbf{Y}$. Therefore, the solution of the system (12) can be determined as follows:

1. Solve the two systems: $\Gamma u = \mathbf{Y}$, and $\Gamma v = \beta_2$ for u and v using the conjugate gradient algorithm in Table 1.

2. Determine the matrix θ from the relationship: $\theta = \mathbf{Y}^T u$.

3. Compute the desired learning parameters for the SVM classifier using equations: $b = \frac{u^T \beta_2}{\theta}$, and $\lambda = v - u\, b$.

Therefore, we conclude that the solution of each of the systems (7) and (11)is easy to compute, and there is no need to store the matrix $\mathbf{A}$. This new technique run faster, and the computational running time will be $O(n\, \delta^2)$ only. Moreover, we do not need to compute the inverse of $n \times n$ matrix, which is difficult when n is very large. The *Matlab code for least squares support vector machine for arbitrary Kernel* to compute b and λ is written as:

```
function [lambda,b]=svmls(Gamma,Y,C);
% Gamma n x m is n input data points from R^m: Assume n >> m
[n,m]=size(Gamma);
for i=1:n,
    Gamma(i,:)=Y(i)*Gamma(i,:);
end
B=inv(eye(m)/C+Gamma'*Gamma);
W=B*Gamma';
X=C*Y'-C*(Y'*Gamma)*W;
u=ones(n,1); b=(X*u)/(X*Y);
lambda=C*u-(C*Gamma)*(W*u)-b*X';
```

5 Multiclass Least squares SVMs Classifier

Dealing with multiclass pattern classification by support vector machines, both combining several binary classifiers or considering all classes at once is still an ongoing research issue, see Kulkarni et al. (2004) for further details. Figure 5 shows the most common SVM architecture. In this graph, the results are linearly

Top

combined by weights ν_i, found by solving a quadratic program and computing $\omega_i = \sum_{i=1}^{n} \nu_i \, \mathbf{x}_i$, where ν_i is the support vectors, that is $\nu_i = y_i \, \lambda_i$ in the pattern classification; and $\nu_i = \lambda_i^\star - \lambda_i$ in the regression estimation. The linear combination is fed into the decision function $\sigma(.)$ or $f(.)$. In pattern classification, $f(\mathbf{x}) = sign(\omega \, \mathbf{x} + b)$; and in regression estimation, $f(\mathbf{x}) = \omega \, \mathbf{x} + b$. We explain two SVMs multi-class techniques: The *one-against-all* technique.

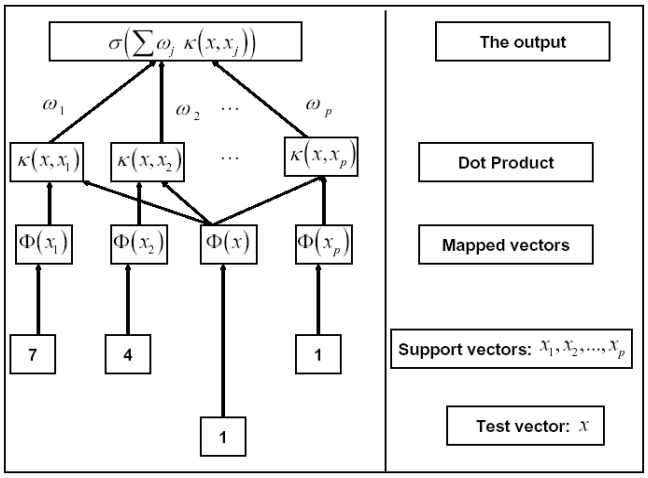

Figure 3: The Architecture of Support Vector Machines Classifier.

Consider a training set $D = \{(\mathbf{x}_i, y_i)\}$, for all $i = 1, 2, \ldots, n$ drawn from c categories; where the vector $\mathbf{x}_i \in \mathbb{R}^p$ represents the i^{th} observation of the given training set D. We use the symbol $y_i^{(k)}$ to refer to the i^{th} output unit for the pattern (class k), for all $k = 1, 2, \ldots, c$. Similarly, as in the binary case, the support vector machines uses the least square technique to handle the pattern classification problem. Therefore, the optimization problem can be written as:

$$
\begin{aligned}
&\text{Min } J^{(k)}(\omega_k, b_k, \eta_{i,k}) = \tfrac{1}{2} \sum_{k=1}^{c} \omega_k^T \, \omega_k + \tfrac{\gamma}{2} \sum_{i=1}^{n} \sum_{k=1}^{c} \eta_{i,k}^2, \\
&\text{subject to} \\
&\mathbf{y}_i^{(1)} \ \left(\omega_1^T \varphi_1(\mathbf{x}_i) + b_1\right) \geq 1 - \eta_{i,1}; \ \eta_{i,1} \geq 0; \ i = 1, 2, \ldots, n, \\
&\mathbf{y}_i^{(2)} \ \left(\omega_2^T \varphi_2(\mathbf{x}_i) + b_2\right) \geq 1 - \eta_{i,2}; \ \eta_{i,1} \geq 0; \ i = 1, 2, \ldots, n, \\
&\vdots \\
&\mathbf{y}_i^{(c)} \ \left(\omega_c^T \varphi_c(\mathbf{x}_i) + b_c\right) \geq 1 - \eta_{i,c}; \ \eta_{i,1} \geq 0; \ i = 1, 2, \ldots, n.
\end{aligned}
\tag{13}
$$

The corresponding Lagrangian function can be formulated as:

$$
L^{(k)}(\omega_k, b_k, \eta_{i,k}, \lambda_{i,k}) = J^{(k)}(\omega_k, b_k, \eta_{i,k}) - \sum_{i=1}^{n} \sum_{k=1}^{c} \lambda_{i,k} \{\mathbf{y}_i^{(k)} \ \left(\omega_k^T \, \varphi_k(\mathbf{x}_i) + b_k\right) - 1 + \eta_{i,k}\}, \tag{14}
$$

for all $k = 1, 2, \ldots, c$, where $0 \leq \lambda_{i,k} \leq \gamma$ are the associated Lagrangian multipliers. As before, in practical real application, the dual quadratic optimization problem to the optimization problem in (13) needed to be calculated. The necessary conditions of the Lagrangian function in (14)is given by:

$$
\frac{\partial L}{\partial b_k} = 0 \Rightarrow \sum_{i=1}^{n} \lambda_{i,k} \, \mathbf{y}_i^{(k)} = 0; \ \frac{\partial L}{\partial \omega_k} = 0 \Rightarrow \omega_k = \sum_{i=1}^{n} \lambda_{i,k} \, \mathbf{y}_i^{(k)} \, \varphi_k(\mathbf{x}_i);
$$

Top

$$\frac{\partial L}{\partial \eta_{i,k}} = 0 \Rightarrow \gamma\,\eta_{i,k} - \lambda_{i,k} = 0;\ \ \frac{\partial L}{\partial \lambda_{i,k}} = 0 \Rightarrow \mathbf{y}_i^{(k)}\,\left(\omega_k^T\,\varphi_k(\mathbf{x}_i) + b_k\right) - 1 + \eta_{i,k} = 0,$$

for all $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, c$. As before, by eliminating ω_k and $\eta_{i,k}$, the above optimality conditions lead to $(n+1)^2 \times (n+1)^2$ linear system of equations, that is,

$$\left[\begin{array}{c|c} 0 & \mathbf{Y}_K^T \\ \hline \mathbf{Y}_K & \Omega_K \end{array}\right]\,\left[\begin{array}{c} b_K \\ \hline \lambda_K \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \widetilde{1} \end{array}\right], \tag{15}$$

where $\mathbf{Y}_K$ and Ω_K are two diagonal matrices in the form:

$$\mathbf{Y}_K = \left[\begin{array}{cccc} \mathbf{Z}^{(1)} & 0 & \ldots & 0 \\ 0 & \mathbf{Z}^{(2)} & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & \mathbf{Z}^{(c)} \end{array}\right];\ \mathbf{Z}^{(k)} = \left[\begin{array}{c} \mathbf{y}_1^k \\ \mathbf{y}_2^k \\ \vdots \\ \mathbf{y}_n^k \end{array}\right];\ \Omega_K = \left[\begin{array}{cccc} \Theta^{(1)} & 0 & \ldots & 0 \\ 0 & \Theta^{(2)} & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & \Theta^{(c)} \end{array}\right],$$

where $\Theta^{(k)} = [\theta_{ls}^{(k)}]_{n\times n}$ matrix with $\theta_{ls}^{(k)} = \mathbf{y}_l^k\,\mathbf{y}_s^l\,\Psi_k(x_l, x_s) + \gamma^{-1}\,I$, for all $k = 1, 2, \ldots, c$, and $l, s = 1, 2, \ldots, n$. The function $\Psi_k(x_l, x_s)$ is the kernel matrix: $\Psi_k(x_l, x_s) = \varphi_k(x_l)\,\varphi_k(x_s)$, for all $k = 1, 2, \ldots, c$, and $l, s = 1, 2, \ldots, n$. The solution vectors $b_K = [b_1; b_2; \ldots, b_c]$ and $\lambda_K = [\lambda^{(1)}; \lambda^{(2)}; \ldots, \lambda^{(c)}]$; where $\lambda^{(k)} = [\lambda_{1,k}, \lambda_{2,k}, \ldots, \lambda_{n,k}]$; for all $k = 1, 2, \ldots, c$.
Similarly as we did for the binary classification case, we can use the conjugate gradient least square technique proposed in section 4 to solve the system (15). To achieve this goal, we need to modify the structure of both matrices Ω_K and $\mathbf{Y}_K$, and we should avoid storage the matrix Ω_K, due to its computational cost and complexity. In addition, the matrices in (15) needed to be reformatted, so that the two linear subsystems of equation have positive definite matrices. Therefore, we can determine the support vector machines parameters as follows:

1. Solve the two systems: $\Gamma^{(k)}\,u^{(k)} = \mathbf{Y}^{(k)}$, and $\Gamma^{(k)}\,v^{(k)} = \beta_2^{(k)}$ for u^k and $v^{(k)}$, where $k = 1, \ldots, c$ using the conjugate gradient algorithm in Table 1.

2. Determine the matrix $\theta^{(k)}$ from the relationship: $\theta^{(k)} = \left(\mathbf{Y}^{(k)}\right)^T\,u^{(k)}$.

3. Compute the desired learning parameters for the SVM classifier using equations: $b^k = \frac{\left(u^{(k)}\right)^T\,\beta_2^{(k)}}{\theta^{(k)}}$, and $\lambda^k = v^{(k)} - u^{(k)}\,b^k$ for $k = 1, \ldots, c$,

and hence we can determine the c decision hyperplanes. These c separable hyperplanes are given by the decision functions: $\omega_k^T\,\varphi(\mathbf{x}) + b^k$, for all $k = 1, 2, \ldots, c$. We can classify the object $\mathbf{x}$ to the class k if the class k has the largest value of the following decision function:

$$\text{class of } \mathbf{x} = \ \arg\ \left[\ \max_{\ k=1,2,\ldots,c}\left(\omega_k^T\,\varphi(\mathbf{x}) + b^k\right)\right].$$

6 Simulation Study and Real World Applications

In this section we illustrate the use of the support vector machine classifier with the new conjugate gradient technique, by examining its applications in both simulation study and real-world problems.

6.1 Design of The Simulation Study

We carry out a simulation study and investigate the performance of the new multiclass SVM classifier discussed in section 5 and handle all the implementation using MATLAB software. We generate both feature variables and the response variable according to the following criterion:

Table 2: Data Setups for Multivariate Feature Variables

Setup	μ_0	...	μ_{c-1}	ρ	n_0	...	n_{c-1}
1	$[0 \ \ldots \ 0]$	...	$[0 \ \ldots \ 0]$	0.5	1000	...	1000
2	$[0 \ \ldots \ 0]$	...	$[0 \ \ldots \ 0]$	-0.5	1000	...	1000
3	$[0 \ \ldots \ 0]$	...	$[0 \ \ldots \ 0]$	0.9	1000	...	1000
4	$[0 \ \ldots \ 0]$	...	$[0 \ \ldots \ 0]$	-0.9	1000	...	1000
5	$[\mu^{(0)} \ \ldots \ \mu^{(0)}]$	...	$[\mu^{(c-1)} \ \ldots \ \mu^{(c-1)}]$	0.5	1000	...	1000
6	$[\mu^{(0)} \ \ldots \ \mu^{(0)}]$	...	$[\mu^{(c-1)} \ \ldots \ \mu^{(c-1)}]$	-0.5	1000	...	1000
7	$[\mu^{(0)} \ \ldots \ \mu^{(0)}]$	...	$[\mu^{(c-1)} \ \ldots \ \mu^{(c-1)}]$	0.9	1000	...	1000
8	$[\mu^{(0)} \ \ldots \ \mu^{(0)}]$	...	$[\mu^{(c-1)} \ \ldots \ \mu^{(c-1)}]$	-0.9	1000	...	1000
9	$[\mu^{(0)} \ \ldots \ \mu^{(0)}]$	...	$[\mu^{(c-1)} \ \ldots \ \mu^{(c-1)}]$	0.5	800	...	1200
10	$[\mu^{(0)} \ \ldots \ \mu^{(0)}]$	...	$[\mu^{(c-1)} \ \ldots \ \mu^{(c-1)}]$	-0.5	800	...	1200
11	$[\mu^{(0)} \ \ldots \ \mu^{(0)}]$	...	$[\mu^{(c-1)} \ \ldots \ \mu^{(c-1)}]$	0.9	800	...	1200
12	$[\mu^{(0)} \ \ldots \ \mu^{(0)}]$	...	$[\mu^{(c-1)} \ \ldots \ \mu^{(c-1)}]$	-0.9	800	...	1200

- Suppose that we generate p feature variables $X_1, \ldots, X_p$ from a multivariate normal distribution with mean vectors μ of size $p \times 1$ and a common $p \times p$ covariance matrix, Σ, that is defined as follows:

$$\mu = \begin{pmatrix} \mu_0 \\ \mu_1 \\ \vdots \\ \mu_{c-1} \end{pmatrix}; \ \Sigma = \begin{pmatrix} 1 & \rho_{1,2} & \rho_{1,3} & \cdots & \rho_{1,p} \\ \rho_{2,1} & 1 & \rho_{2,3} & \cdots & \rho_{2,p} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ \rho_{p-1,1} & \rho_{p-1,2} & \rho_{p-1,3} & \cdots & \rho_{p-1,p} \\ \rho_{p,1} & \rho_{p,2} & \rho_{p,3} & \cdots & 1 \end{pmatrix}, \tag{16}$$

  where ρ_{j_1,j_2} is the correlation coefficient between the two variables x_{j_1} and x_{j_2}, for all $j_1, j_2 = 1, 2, \ldots, p$. We use $\rho_{j_1,j_2} = \pm 0.5$, and $\rho_{j_1,j_2} = \pm 0.9$, and we generate n_k observations from $N(\mu_k, \Sigma)$, where μ_k is the mean for the observations in group k, for $k = 0, 1, \ldots, c-1$. So, we have a total of $n = \sum_{k=0}^{c-1} n_k$ observations, which are denoted by an $n \times p$ matrix $\mathbf{X}$, the i^{th} row of which is denoted by $\mathbf{x}_i^T$. The setup details for $1000 \times c$ observations is shown in Table 2. We can choose p feature variables drawn from c classes with different correlation coefficients, that is, ρ_s, for $s = 1, 2, \ldots, \binom{p}{2}$ distinct correlation coefficients. Moreover, we can choose different variance, say σ_j, such that $\sigma_j \neq \sigma_l$, for $l, j = 1, 2, \ldots, p$.

- One way to generate the response variable (either multi-categories or binary), y according to the criterion: By setting the probability that observation i falls in a class A_k, π_{ik} as a function of $\mathbf{x}_i$, where the chosen function is other than the logistic function, but satisfying the condition $0 \leq f(x_i) \leq 1$. The form that we choose for π_i can be written as follows:

$$\pi_i = f(\mathbf{x}_i) = \frac{1}{1 + d_i}, \tag{17}$$

  where $d_i = (\mathbf{x}_i - \bar{\mathbf{x}})^T \Sigma^{-1} (\mathbf{x}_i - \bar{\mathbf{x}})$ is the squared Mahalanobis distance, $i = 1, 2, \ldots, n$.

The next step in the development and implementation of the SVM classifier is the judgment and evaluation of the quality and capability of the fitted support vector machine hyperplane. We investigate one of the following quantities:

1. Number of parameters in the model, the more parameters is the more ability to perform better. Therefore, it is important to know the number of parameters in the choice model, and check if this is the best model or not. This will lead to the calculate the following measure, for each configuration:

Top

2. Compute the cost of computations (executable time), that is the time needed to execute the classifier till obtaining the final classification model. The less computation cost is the better classifier.

3. Compute the correct classification rate (CCR) in both old and new SVM classifiers, the highest CCR value is the better performance.

To evaluate the performance of the SVM classifier in both old and new computation, we use both *internal* and *external* validation, that is summarized as follows:

- *Internal Validation*: For each method, we estimate the classification model and its parameters, and then use the obtained estimated model to measure the correct classification rate by reapplying them to the same data used during the estimation. This leads to a bias result, and a higher correct classification rate, because the classifier benefits from the data on which it is used in estimating the same model. Moreover, the performance is better than it would be in different simulated data set or in a real-world applications. To get around this problem in each simulation, we generate a training set with a very large number of observations, say $10,000$ observations to use it in building the classification model, and measure the correct classification rate. This will lead to a small bias performance, and the simulated data will be close enough to a real-world application.

- In the external validation, we first generate a data set (training set) with a very large number of observations, say $10,000$ observations to build the classification model. Next we generate another data set (validation set) from the same configuration as the training set, but with less number of observations to validate the estimated model. For instance, suppose that the new sample (validation sample) has $5,000$ observations, that is used to validate the estimated model to compute the correct classification rate of the new sample. This will lead to unbias results, and less correct classification rate, and the simulated data will be close to a real-world application problem. In general, the real-world applications have a data sat, say $D = \{(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i)\}$, we use it to estimate the pattern classification model. Next, we use the estimated model to predict $\hat{y}_i$ as a future observation, and compute the correct classification rate.

6.2 Real World ApplicationsFisher Iris Data

Most of the real-world application data are available and it can be downloaded from the machine learning database site at the University of California, Irvine, " ftp://ftp.ics.uci.edu/pub/machine-learning-databases".

1. **Fisher Iris data**: We choose this *Fisher Iris data*, (1936) due to its important a known database to be found in the pattern recognition literature. The entire data set contains four feature variables, all of them are measured in centimeters, each one of the feature vectors is drawn from 3 classes of 50 observations each, where each class refers to a type of *iris plant*. We use the symbols x_1, x_2, x_3 and x_4 to refer to the *sepal length*, *sepal width*, *petal length* and *petal width*, respectively. By looking at the scatter plot of the data in Figure 1, we observe that one class is linearly separable from the other 2; the latter are not linearly separable from each other.

   To evaluate the performance of the support vector machine classifier on the iris data set, we use 80 % of the data to estimate the classification model (internal validation) and 20 % of the data for external validation. As proposed in Turk (1990) and Cook (1986) to guarantee that we get the same proportions from each group as in the original data, we use *stratified random sampling* technique, by dividing the data set into non-overlapping samples, then we randomly hold a data of size m, such that $m = round(\frac{n}{5})$ observations with $m_k = round(m\,\frac{n_k}{n})$, where n is the number of instances in a given data and n_k is the number of instances in the group k. In the iris data, we have the training set with 120 observations, and the validation set with 30 observations, such that $m_1 = m_2 = m_3 = 10$ observations from class1, class2, and class3, respectively. We repeat the internal and external validation

Top

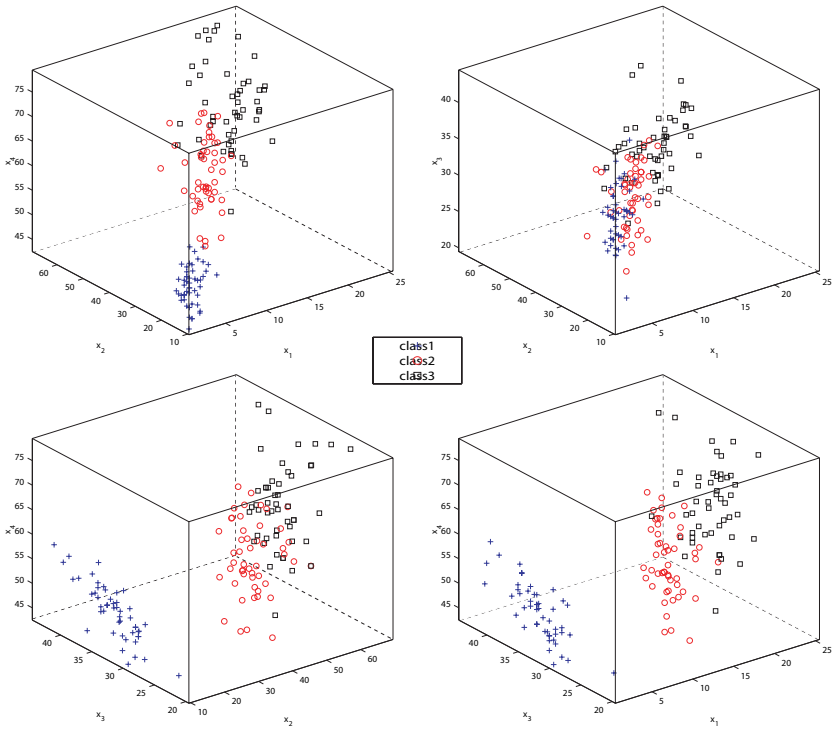

Figure 4: The three dimension plot of the Sepal Length, Sepal Width, and Petal Length.

processes for at least 100 runs, then compute the correct classification rate, its standard deviation, and the variability coefficient for these 100 runs, and compute the same output. For the four explanatory variables, and $c = 3$ category variables, then we obtain the best model with 5 parameters.

The results showed that SVMs classifier with the new computation algorithm produce a highly accurate correct classification rate, with a 98% success versus 96.67% in the case of the old computation technique. In addition, it was shown that the SVMs are significantly quicker than any other old SVM computation classifier technique in both training and running phases as it is shown in Figure 1, where the root mean square error (RMSE) reached to zero after 20 iterations only with the new technique, but it takes more iterations and more time to get better accuracy with the old computation technique, for instance, $RMSE$ reached to 10^{-2} after a 100 iterations. The estimated decision classification model is given by:

$$\text{class of } \mathbf{x} \quad = \quad \arg \left[\max \left(-0.001x_1^2 + 0.001x_2^2 + 0.01x_3^2 + 0.0057x_4^2 - 0.0011x_1x_2 \right) \right]. \qquad (18)$$

2. **Fraud Detection**: Fraud is a critical problem for most businesses, and on-line shopping services. The current explosion of the Internet and computer networks makes the issue even more difficult. Detecting fraudulent patterns in masses of records is a challenging problem, not only because identifying inconsistency may be looking for a spine in a haystack, but mostly because fraudulent activities are all dissimilar and can change dramatically over time, Cahill et al. (2002). Most available intrusion detection systems are based on methods that compare the network activity to a set of known attack signatures provided by human experts. Such systems would therefore fail in detecting a new, yet unknown, type of attack. Support vector machine method as one of the machine learning techniques help address this challenge and allow the development of efficient, more adaptive and autonomous intrusion detection systems.

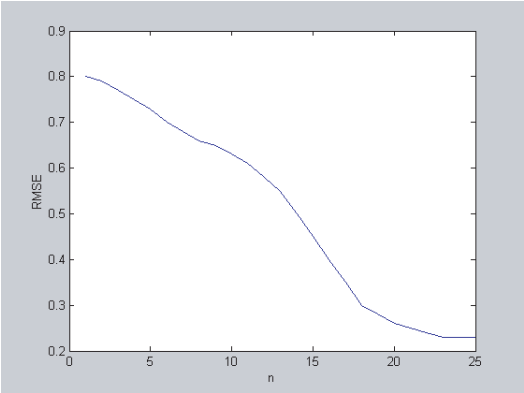

Figure 5: RMS Error versus the number of iterations n.

Mukkamala et al. (2002) are applying support vector machines to the data set used for the knowledge discovery, where this contains 38 varieties of intrusion attack within 4,900,000 data instances simulated in a military local-area network environment. Each data record is composed of 42 features describing a network connection (e.g. duration, protocol type, port, super user access, guest login, etc.) and has been flagged bad for intrusions or attacks or good for normal connections. The approach followed is the "supervised" learning approach for which the support vector machine system is first trained with a subset of the data, including the good and bad label. Once the machine is trained, its efficiency at predicting if a connection is fraudulent or not is assessed against the rest of the training set. Figure 6 shows that the support vector machines can detect network intrusions in order to protect a computer system from fraudulent users. Generally, we observe that the shows that the support vector machines

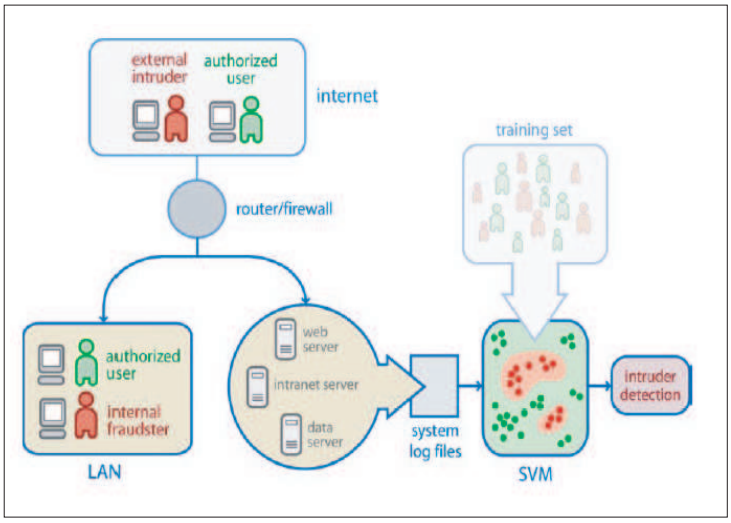

Figure 6: Intrusion Detection on Computer Networks

can detect network intrusions in order to protect a computer system from fraudulent users.. These outputs make SVMs adequate classifiers in such situations where the intrusion detection has to be

Top

performed in retraining needs to be carried out frequently and swiftly. We conclude that SVMs can be successfully applied to fraud detection in many different types of activity, such as, master cards, and bank card operations and computer network security. These SVM techniques perform better than both classical and modern fraud detection classifiers, as they are more suited to the detection of new unknown types of attack. In addition, as it is shown in Figure 1, we observe that the shows that the support vector machines can detect network intrusions in order to protect a computer system from fraudulent users.

3. **Bioinformatics**: The Gene expression analysis performed by SVMs is discussed in Eisen et al. (1998). Gene expression data from DNA microarray hybridization experiments was used. The gold standard applied is MIPS yeast genome database, see MYGD (1999) for more details. Support vector machines with different similarity metrics as well as other machine learning approaches are used in the analysis including Fishers linear discriminant, and two decision tree classifiers. SVMs showed the best performance in accurately classifying genes into functional categories with respect to the other methods. Among different types of SVM, the radial basis function support vector machine performed the best for this task. Previous work in Eisen et al. (1998) had used a normalized dot product as similarity metric. Apart from large amounts of RNA expression data, other sources of information about the genes can be used including the presence of transcription factor binding sites in the promoter region or sequence features of the translated protein.

   The method shows potential to manage with the more complex problem of reconstructing complete regulatory pathways within the cell. Detection of remote protein homologies by SVMs is discussed in Jaakkola et al. (2000). The discriminative method is built on top of a generative model, for instance, hidden Markov models (HMMs), built from multiple sequences and which provides appropriate features for the identification of structural relationships. Accordingly, the protein sequences are mapped to points of an Euclidian feature space with fixed dimension. The method provides with a significant improvement in relation to previous approaches to classification of protein domains based on remote protein homologies including approaches based on training of HMM parameters, Duan et al. (2003) based on neural discrimination, Dubehak et al. (1997).

7 Conclusions

We conclude that the performance of the new SVM classifier technique turns to be robust and efficient with respect to the parameters of the algorithm. The conjugate large scale algorithm involves solving a linear system of equations instead of quadratic programming. Moreover, the conjugate gradient algorithm helps us to minimize the complexity running time, and it becomes less expense. We do not need to store the entire data, and hence the computational running time will be of $O(n)$ only, which is more useful in the case of multi-category problems.
Areas of application research discussed included bioinformatics, fraud detection, and text categorization. These applications proved that SVMs can be successfully applied to fraud detection in many different types of activity, such as, master cards, and bank card operations and computer network security. These SVM techniques perform better than both classical and modern fraud detection classifiers, as they are more suited to the detection of new unknown types of attack.

References

[1] Abbott M. B.,(1997), "Engine 2000: research in to the next generation of computational hydraulic modelling". *Proc., 27th Congr. International Association for Hydraulic Research, Delft, The Netherlands*, 2, 859–864.

[2] Bao L. and Sun Z., (2002),"Identifying genes related to drug anticancer mechanisms using support vector machine". *FEBS Letters*, 521(1–3), 109–114.

[3] Boser B. E., Guyon I., and Vapnik V., (1992), "A training algorithm for optimal margin classifiers". *Proceedings of the Fourth Workshop on Computational Learning Theory*, ACM Press, San Mateo, CA, pp. 144–152.

[4] Bredensteiner E. J., Bennett K. P., (1999),"Multicategory Classification by Support Vector Machines", *Computational Optimization and Applications* (12) 53–79.

[5] Burbidge R., Trotter M., Buxton B. and Holden S., (2001), "Drug design by machine learning: Support vector machines for pharmaceutical data analysis". *Computers and Chemistry*, 26, 5–14.

[6] Burges C. J. C., (1998), "A tutorial on support vector machines for pattern recognition". *Data Mining and Knowledge Discovery*, 2(2), 121–167.

[7] Cahill M.H., Lambert D., Pinheiro J.C. and Sun D.X., (2002), *Detecting Fraud in the Real World*, in Handbook of Massive Datasets, Kluwer Academic Publishers.

[8] Cook, R. L., (1986), "Compute the third vertex of the sub-triangle". *Stochastic sampling in computer graphics*. ACM Transactions on Graphics 5, 1 51–72.

[9] Cristianini N. and Shawe-Taylor J., (2000), *An introduction to Support Vector Machines*. Cambridge: Cambridge University Press.

[10] Duan K., Keerthi S. S., Poo A. N., (2003), "Evaluation of simple performance measures for tuning SVM hyperparameters", *Neurocomputing* (51) 41-59.

[11] Dubchak I., Muchnik I., Kim S., (1997), "Protein folding class predictor for SCOP: approach based on global descriptors". *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, Halkidiki, Greece, pp. 104-108.

[12] Duda O. Richard , Hart E. Peter and Stork G. David , (2001), *Pattern Classification*, Wiley Interscience, Second Edition, ISBN 0-471-05669-3, John Wiley & Sons, Inc, New York.

[13] Eisen M., Spellman P., Brown P., Botstein D., (1998), "Cluster analysis and display of genome-wide expression patterns", *Proc. Natl. Acad. Sci.*, USA 95, 14863-14868.

[14] Fisher R. A., (1936), "The Use of multiple Measurements in taxonomic Problems", *Ann. Eugenics*, **7**, 179–188.

[15] Goloub G. h., Van Loan C. F., (1989), *Matrix computations*, Baltimore MD: John Hopkins University Press.

[16] Gordon, A. D. ,(1981), *Classification: Methods for the Explanatory Analysis of Multivariate Data*, Chapman & Hall, London.

[17] Gordon, A. D., (1999), *Classification*, 2nd Edition, Chapman & Hall/ CRC.

[18] Gunn S., (1998), *Support vector machines for classification and regression*. Image, Speech and Intelligent Systems Tech. Rep., University of Southampton, U.K.

[19] Hosmer D. L. and Lemeshow S., (2000), *Applied Logistic Regression*, second edition. John Wiley & Sons, New York.

[20] Hsu C. and Lin C., (2002), "A comparison of methods for multi-class support vector machines". *IEEE Transactions on Neural Networks*, 13:415–425.

[21] Jaakkola T., Diekhans M., and Haussler D., (2000), "A discriminative framework for detecting remote protein homologies", *Journal of Computational Biology*, 7 (12), 95-114.

[22] Jack L. B. and Nandi, A. K., (2002), "Fault detection using support vector machines and artificial neural networks, augmented by genetic algorithms". *Mechanical Systems and Signal Processing*, 16(2–3), 373–390.

[23] Kuhn, H. W., and Tucker, A. W., (1961), "Nonlinear Programming," in proceedings of the second Berkeley Symposium on Mathematical Statistics and Probability, *J. Neyman (editor), University of California Press, Berkeley and Los Angeles, California*, pages 481–492.

[24] Luntz A. and Brailovsky V., (1969), *On estimation of characters obtained in statistical procedure of recognition.* Technicheskaya Kibernetica, 3 (in Russian).

[25] Mukkamala S., Janoski G.I. and Sung A. H., (2002), "Intrusion Detection Using Neural Networks and Support Vector Machines", To appear in IEEE IJCNN.

[26] MYGD, (1999), "Munich information center for protein sequences yeast gene database", http://www.mips.biochem.mpg.de/proj/yeast.

[27] Quinlan J. R., (1986), "Induction of decision trees". *Machine Learning*, 1, 81–106.

[28] Quinlan J. R., (1992), *C4.5: program for machine learning*, Morgan Kaufmann, San Mateo, California, USA.

[29] Raudys S., (2000), "How good are support vector machines?" *Neural Networks*, 13, 17–19.

[30] Ross P. E., (1998), "Flash of Genius", *Forbes*, 98 – 104.

[31] Schéolkopf B., Burges C., Vapnik V., (1995), "Extracting Support Data for a Given Task", in: U.M. Fayyad, R. Uthurusamy (Eds.), *Proceedings, First International Conference on Knowledge Discovery and Data Mining*, pp. 252–257.

[32] Suykens J. A. K. and Vandewalle J., (1999a), "Least squares support vector machine classifiers," *Neural Process. Lett.* 9, 293–300.

[33] Suykens J.A.K. and Vandewalle J., (1999b), "Training multilayer perceptron classifiers based on a modi.ed support vector method". *IEEE Trans. Neural Networks* 10(4), 907–912.

[34] Suykens J. A. K., Vandewalle J. and Moore, B. D., (2001), "Optimal control by least squares support vector machines". *Neural Networks*, 14(1), 23-35.

[35] Turk, G., (1990), "Generating Random Points in Triangles. In Graphics Gems", *Academic Press*, New York, 1990, pp. 24–28.

[36] Vapnik V. and Chervonenkis A., (1974), *Theory of pattern recognition*, Nauka, Moscow.

[37] Vapnik V., (1995), *The nature of statistical learning theory*, Springer, New York.

[38] Vapnik V., (1998), *Statistical learning theory*, John Wiley, New York.

[39] Watkins C. J. C. H. and Dayan P., (1992), "Q-learning", *Machine Learning*, 8, 279–292.