# A Study of the Relationship Between Support Vector Machine and Gabriel Graph

Wan Zhang and Irwin King
{wzhang, king}@cse.cuhk.edu.hk

Department of Computer Science & Engineering
The Chinese University of Hong Kong
Shatin, NT., Hong Kong

**Abstract - In recent years, Support Vector Machine (SVM) has become a very dynamic and popular topic in the Neural Network community for its abilities to perform classification, estimation, and regression. One of the major tasks in the SVM algorithm is to locate the discriminant boundary in classification task. It is crucial to understand various approaches to this particular task. In this paper, we survey several different methods of finding the boundary from different disciplines. In particular, we examine SVM from the statistical learning theory, the Convex hull problem from the Computational Geometry's point of view, and Gabriel graph from the Computational Geometry perspective to describe their theoretical connections and practical implementation implications. Moreover, we implement these methods and demonstrate their respective results on the classification accuracy and run time complexity. Lastly, we conclude with some discussions about these three different techniques.**

## I. Introduction

Given a data set which contains the data belonging to two or more different classes, either linearly separable or nonseparable, the problem is to find the optimal separating hyperplane (Decision Boundary) to separate the data according to their class type. Support Vector Machines (SVMs) have attracted wide interest as a mean to implement structural risk minimization for the problem of classification and regression estimation introduced by Vapnik in the late seventies [8]. This led to a recent explosion of applications and deepening theoretical analyses, that has now established SVMs along with neural networks as one of the standard tools for machine learning and data mining. There are three key points needed to understand SVM: (1) maximizing margins, (2) the dual formulation, and (3) kernel functions. Most people intuitively grasp the idea that maximizing margins should help improve generalization. Kernel functions are used to solve the problems in high-dimensional space. But changing from the primal to dual formulation is typically black magic for those unfamiliar with duality theory. If the primal problem $P$ has its dual promblem, it means the solution of one problem could be recovered from that to the other. For example, Lagrangian Dual problem is formulated with the Lagrangian dual objective function which statisfies the constraints of the primal problem, and maintains any other constraints within the data set. To better understand SVMs, a geometric interpretation from the dual perspective along with a mathematically rigorous derivation of the ideas behind the geometry, can be useful.

Convex hull could be a simple intuitive geometric explanation of SVM [2]. The convex hull of a set of points is the smallest convex set which contains all the points. It is also the fundamental construction for mathematics and computational geometry. It turns out that finding the nearest points of the two convex hulls is similar to finding the separating hyperplane with maximum margin in the SVM case [2].

From [1], we know that Convex hull could be used to solve many problems, such as half space intersection, Delaunay triangulation, Voronoi diagrams, etc.

The Voronoi diagram and Delaunay triangulation are two of the possible representations for $K$-Nearest neighbor rule. The $K$-Nearest neighbor rule can be seen as the non-parametric decision rule which needs no prior knowledge of the distributions [3]. Decision rules are used in many areas such as pattern recognition and database. They are used to determine the class membership for a point based on some computational measurements for the point. Despite simplicity and good performance of $K$-Nearest neighbor rule, the traditional criticism of the method is that it needs a large storage space for the entire training data and the necessity to query the entire training set in order to make a single membership classification. As a result, there has been considerable interest

0-7803-7278-6/02/$10.00 ©2002 IEEE

in editing the training set to reduce its size.

Just like SVMs choose support vectors which are a small part of the whole training set to find the separating hyperplane, different proximity graphs (such as Delaunay triangulation and Gabriel graph) provide efficient geometric apparatus for solving the problem and find the decision boundary. The Gabriel graph of a set of points is a subgraph of Delaunay triangulation for that set, which is a dual of Voronoi diagram [3].

This paper tries to bring all these concepts to a unified application domain. We use Convex hull to be a bridge which connects SVM and Gabriel graph. Both of them could be used in the classifier and have a good performance not only on accuracy but also on time complexity. We also performed experiments to show the performance the two methods and made the discussion of their advantages and drawbacks.

In the next section, we will present the three main concepts for comparison: (1) SVM, (2) Convex hull, and (3) Gabriel graph. In Section 3, we conduct a series of experiments that will use the different data sets and demonstrate the performance of the different methods. The discussion of these results is in Section 4. Lastly, we conclude and make some final remarks in Section 5.

## II. Related Background

In this section, we plan to take a survey on three different concepts for finding the separating hyperplanes in a data set. These concepts come from different disciplines in Computer Science, ranging from Computational Geometry to statistical learning theory. We want to show the similar relationship among these different concepts arising from different disciplines.

We will start with the simplest case–linear separable data. Later we will see that the analysis for the general case, nonlinear-separable data , results in a very similar programming problem.

### A. Support Vector Machine

Given the training data, $x_i, y_i$, $i = 1, \cdots, l$, $y_i \in \{-1, 1\}$, $x_i \in \mathbf{R}^n$, suppose there exists a hyperplane which could separate the positive from the negative data set. It means that the points $x$ which lie on the hyperplane satisfy $\omega \cdot x + b = 0$, where $\omega$ is normal to the hyperplane, $|b|/\|\omega\|$ is the perpendicular distance form the hyperplane to the origin, and $\|\omega\|$ is the Euclidean norm of $\omega$. Define the margin is the sum of the distance of the separating hyperplane to the closest positive and negative points [4]. For the linearly separable case, SVM simply finds the separating hyperplane with the largest margin. It could be formulated as a set of linear constraints for all the training data points as:

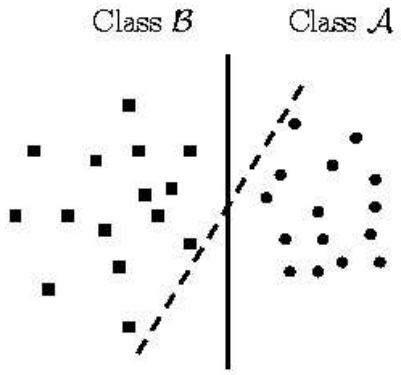


Fig. 1. Linear separating hyperplanes for the separable case

$$y_i(\omega \cdot x_i + b) - 1 \geq 0, \quad \forall i. \tag{1}$$

From Fig. 1 and Eq. (1), we can calculate the margin as $2/\|\omega\|$. Then we can find the separating hyperplane with the largest margin under the constraint in Eq. (1) by minimizing $\|\omega\|^2$ in

$$\min \frac{1}{2}\|\omega\|^2 \tag{2}$$

$$\text{subject to } y_i(x \cdot \omega + b) \geq 1, i = 1, \cdots, m.$$

Let $\alpha = \alpha_1, \alpha_2, \cdots, \alpha_m$ be the $m$ nonnegative Lagrange multipliers, one for each inequality constraints in Eq. (1), the solution to Eq. (2) equals to the solution to the constrained quadratic optimization problem using the Wolfe dual theory [4] as,

$$L_P \equiv \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j - \sum_i \alpha_i \tag{3}$$

$$\text{subject to } 0 \leq \alpha_i \leq 1, \sum_i \alpha_i y_i = 0.$$

In the soft margin (nonlinear separable case) formulation of [9], the generalized optimal separating hyperplane is regarded as the solution to Eq. (4) as follows,

$$\min(\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} \xi_i) \tag{4}$$

subject to $y_i(x \cdot \omega + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, \cdots, m, C > 0$.

Similarly, the corresponding Lagrangian formulation is defined as

$$\begin{aligned} L_P &= \frac{1}{2}\|\omega\|^2 + C\sum_i \xi_i \\ &\quad - \sum_i \alpha_i\{y_i(x_i \cdot \omega) + b) - 1 + \xi_i\} - \sum_i \mu_i \xi_i. \end{aligned} \tag{5}$$

The optimal solution to this problem satisfies the following Karush-Kuhn-Tucker (KKT) conditions [9]:

$$\alpha_i\{y_i(\omega \cdot \xi_i + b) - 1 + \xi_i\} = 0, \mu_i\xi_i = 0.$$

So its Wolfe dual problem could be defined as

$$\min \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x_i x_j - \sum_i \alpha_i \tag{6}$$

$$\text{subject to } 0 \le \alpha_i \le C, \sum_i \alpha_i y_i = 0.$$

For solving high dimension problems, SVM maps the space of covariates $X$ to a Hilbert space $\mathcal{H}$ of a higher dimension (maybe infinite), and fits an optimal linear classifier in $\mathcal{H}$.

It does so by choosing a mapping function $\phi$:$\mathbf{R}^n \to \mathcal{H}$ in such a way that $\phi(x) \cdot \phi(y) = K(x, y)$ for some known and easy to evaluate set of functions, $K$. Sufficient conditions for the existence of such a map are provided by the Mercer's theorem [9].

An example of the mapping function can be described as follows. Set $Q_{ij} = y_i y_j K(x_i, x_j)$, such that $\alpha \cdot y = 0$, $0 \le \alpha_i \le C$, $i = 1, \cdots, m$. Due to the new mapping function, the objective function is changed as follows

$$R(\alpha) = \frac{1}{2}\alpha \cdot (Q \cdot \alpha) - \alpha. \tag{7}$$

From a Computational Geometric point of view, the solution to the Convex hull problem provides a way to locate support vectors [2]. In the next section, we survey some interesting properties on the computation of Convex hull.

## B. Convex hull

The separating plane is the hyperplane which is orthogonal to the line segment and bisects the line segment which connects the nearest points of the two convex hulls of the data sets.

In Fig. 2, the convex hull of class $A(B)$ consists of all the points which could be written as convex combination of the points in $A(B)$. A convex combination of points in $A, u$, is denoted by $u$=$\sum_{i\in A}\beta_i x_i$, where $i \in A$ ,$\beta_i \ge 0$, and $\sum_{i\in A}\beta_i$=1 and the convex combination of the points in $B, v$, is denoted by $v$=$\sum_{j\in B}\beta_j x_j$, where $j \in B$, $\beta_j \ge 0$, and $\sum_{j\in B}\beta_j$=1. Assume $U$ is the convex hull of $A, V$ is the convex hull of $B$ [7].

The problem of finding two nearest points in the convex hulls could be written as follows:

$$\min \|u - v\| \tag{8}$$

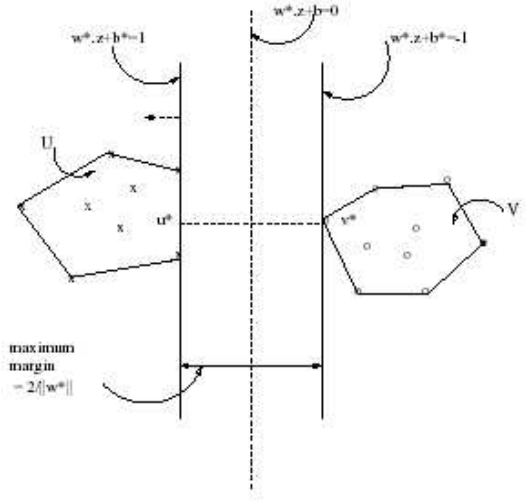


Fig. 2. Two closest points of the two convex hulls determine the separating plane

$$\text{such that } u \in U, \quad and \quad v \in V.$$

If $(\omega, b)$ is the optimal solution of Eq. (3), and $(u, v)$ is the optimal solution of Eq. (8), with the fact that the maximum margin of the two sets=$2/\|\omega\| = \|u - v\|$, and $\omega$ has the same direction with $(u - v)$, so

$$\omega = \frac{2}{\|u - v\|^2}(u - v), \quad b = \frac{\|v\|^2 - \|u\|^2}{\|u - v\|^2}.$$

We could introduce a new variable, $\sigma$, to Eq. (3) and then rewrite $\sum_i \alpha_i y_i = 0$ into two constraints

$$\sum_{i\in A}\alpha_i y_i = \sigma, \quad \sum_{i\in B}\alpha_i y_i = \sigma.$$

Now, we define $\beta_m = \frac{\alpha_m}{\sigma}, \forall m$, so Eq. (3) could be rewritten as follows

$$\min \frac{\sigma^2}{2}\sum_m\sum_k \beta_m\beta_k y_m y_k x_m x_k - 2\sigma \tag{9}$$

$$\text{such that } 0 \le \beta_i \le 1, \sum_{i\in A}\beta_i = 1, \sum_{j\in B}\beta_j = 1.$$

When $\sigma = \frac{2}{\sum_m\sum_k \beta_m\beta_k y_m y_k x_m x_k}$, it is equivalent to the following problem:

$$\min \quad \frac{1}{2}\sum_m\sum_k \beta_m\beta_k y_m y_k x_m x_k \tag{10}$$

$$\text{such that } 0 \le \beta_i \le 1, \sum_{i\in A}\beta_i = 1, \sum_{j\in B}\beta_j = 1.$$

We now define the matrix $P$ with columns: $y_1x_1, y_2x_2, \cdots, y_mx_m$. With this, we can obtain the following equation from Eq. (10) as

$$\sum_m \sum_k \beta_m \beta_k y_m y_k x_m x_k = \|P\beta\| .$$

If $\beta$ satisfies the constraints, then $P\beta = u - v$,where $u \in U$ and $v \in V$. So Eq. (3) is equivalent to Eq. (8).

Except the algebraic description for convex hull, We also could represent a convex hull with a set of facets and a set adjacency lists giving the neighbors and vertices for each facet. The boundary elements of a facet are denoted as ridges. Each ridge signifies the adjacency of two facets. In $\mathbf{R}^3$ and general position, facets are triangles and ridges are edges.

A Delaunay triangulation in $\mathbf{R}^d$ could be computed from a convex hull in $\mathbf{R}^{d+1}$. To determine the Delaunay triangulation of a set of points, we can calculate it by lifting the points to a paraboloid and computing their convex hull. The set of ridges of the lower convex hull is the Delaunay triangulation of the original points [1]. Gabriel graph can be computed by discarding edges from Delaunay triangulation. However, according to the time consumed, this approach to obtain the Gabriel graph is not a very attractive one when number of dimensions is large.

### C. Gabriel graph

In $K$-Nearest neighbor classification, we classify an object (point) in $d$-dimensional space according to the dominant class among its $k$-nearest neighbors from the training data set. It is useful if we can find some representatives from the training set to classify new point while preserving a high accuracy. Both Voronoi diagram and Gabriel graph can be used for such purpose. The general idea is found in [3].

A Voronoi diagram is a partition of special points into regions such that each region consists of points closer to one particular node than to any other nodes. For example, a set of points $P := p_1, p_2, ...$ such that for each cell corresponding to point $p_i$, the points $q$ in that cell are nearer to $p_i$ than to any other point in $P$. In other words, the points $q$ satisfies the following inequality:

$$dist(q, p_i) < dist(q, p_j), \quad p_i, p_j \in P, j \neq i.$$

Therefore, a new point in a Voronoi region must be closer to the region's node than to any other nodes. So, we can assign the new point to the class represented by the region's node. Moreover, the boundaries of the Voronoi regions separating those regions whose nodes are of different class can be used as the decision boundary of the classifier. However, it is clear that the nodes whose boundaries did not contribute to the decision boundary are redundant and can be safely deleted from the training data. Gabriel graph of a set, $S$, of points has an edge between points $p$ and $q$ in $S$ if and only if the diametral sphere of $p$ and $q$ does not contain any other points. The resulting points from the above process make up of the Gabriel edited set. We shall see that decision boundary can be constructed from those Gabriel neighbors ($p$ and $q$) such that $p$ and $q$ are of different classes (see Fig. 3).

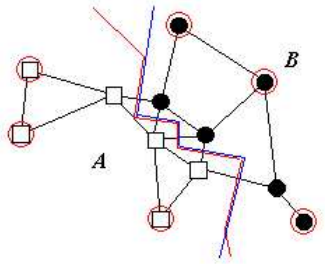


Fig. 3. Gabriel graph

The Gabriel edited set is always a subset of the Voronoi edited set because of the fact that a Gabriel graph of a set of points is a subgraph of Delaunay triangulation for that set. Thus, Gabriel editing which is the procedure of finding the Gabriel neighbors, reduces the size of the training set more than Voronoi editing. Although, the resulting Gabriel editing does not preserve the original decision boundary, the changes occur mainly outside of the zones of interest.

The Gabriel editing algorithm can be formulated as follows:

1. Compute the Gabriel graph for the training set.
2. Visit each node, marking it if all its Gabriel neighbors are of the same class as the current node.
3. Delete all marked nodes, exiting with the remaining ones as the edited training set.
   Clearly, the Gabriel graph can be computed by brute force if for every potential pair of neighbors $A$ and $B$, we just verify if any other point $X$ is contained in the diametral sphere such that $L_2(A, X) + L_2(B, X) < L_2(A, B)$ where $L_2$ is the square of the distance between the two points.

## III. Experiments and Results

### A. Time Complexity

Table I summarizes the theoretical analyses from the research result of Hush and Scovel [6] and Bhattacharya [3], where $n$ is the number of input points, $d$ is the dimension. Here, $\epsilon_n$ is obtained through an appropriate

TABLE I
TIME COMPLEXITY

| Methods | Best Case | Average Case | Worst Case |
|---|---|---|---|
| SVM | $O(n)$ | uncertain | $O(n^5 \log^n / \epsilon_n)$ |
| Gabriel graph | $O(dn^2)$ | $O(dn^3)$ | $O(dn^3)$ |

normalization of objective function, $R$, and depends on $n$. The Best Case for any algorithm is to visit each data point once. For average case analysis with SVM, it typically requires some knowledge of the distribution over problem instances. Moreover, it is not uncommon to see run time estimates of $m^2$ to $m^3$ reported from experiments with these types of algorithms [6]. But the bounds of iteration steps to find the optimal solution could not be made certain.

The Gabriel graph algorithm requires $O(n^2)$ operations to yield $O(n^2)$ pairs of Gabriel neighbors. For each such pair of points $(A, B)$, the algorithm requires $O(nd)$ operations. Hence the overall average complexity of the algorithm is $O(dn^3)$.

From Table I, we see that Gabriel graph is more stable in different cases (linear-separable or nonlinear-separable cases and different data sets) but has poor performance with data having high dimension. Even now we have not found an exact way to measure the iterations that the algorithm will take to compute the optimal solution. In other words, the result of SVM is data-sensitive, i.e., it changes with different data sets. The dimension has little effect on SVM.

The worst case for SVM is that the algorithm of SVM drives the criterion function to the optimum solution in $O(\frac{Cn^4}{\epsilon_n})$ iterations. In each step it will take $O(n \log^n)$ time to determine whether the result satisfies the constraints for the optimal solution. Although this does not happen often in the real world, it is a point to consider when implementing the algorithm.

The following empirical experiments evaluate the performance of SVM and Gabriel graph on two simple data sets: (1) Iris data set and (2) Spiral data set. First, we use Libsvm [5], an integrated software for support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). It supports multi-class classification. The basic algorithm is a simplification of both SMO by Platt and SVMLight by Joachims. It provides both C++ and Java source codes. Second, we implement the Gabriel graph [3] algorithm with Matlab. We run the two algorithms under WINNT operating system and record the result. In these experiments, the SVM parameters are set according to Table II.

TABLE II
SVM PARAMETERS SETTING

| Parameter | Spiral Data | Iris Data |
|---|---|---|
| Kernel Function | Radial Basis Function | Dot Product |
| Error Penalty($C$) | 1000 | 1000 |
| Gamma for RBF | 50 | Default |

TABLE III
EXPERIMENTS ON THE IRIS DATA AND THE SPIRAL DATA

| Method | Size of Data set | Support Vectors | Gabriel Edited Set |
|---|---|---|---|
| Iris Data | 100 | 2 | 2 |
| Spiral Data | 126 | 108 | 123 |

B. Iris Data and Spiral Data

The Iris data set consists of four measurements made on each of 150 flowers. There are three pattern classes: Virginica, Setosa, and Versicolor corresponding to three different types of Iris. In this case, the reference set consists of 150 feature vectors in 4-space each of which is assigned to one of the three above mentioned classes. In these experiments, we only choose two features of the two classes (Iris-setosa and Iris-versicolor) of the Iris data: petal length and petal width as the test data.

Table III shows the results when the Gabriel graph algorithm and SVM are applied to the Iris data. We record the number of support vectors and the size of Gabriel edited set.

In the second example, we illustrate the two methods on a two-spiral benchmark problem. The training data with two classes indicated with the different labels are in a two dimensional input space. (see Fig. 4 and Fig. 5)The excellent generalization performance is clear from the decision boundaries shown in the two methods' experiment. The results are shown in Table III.

## IV. Discussions

As described in above sections, the Convex hull could be one intuitive geometric explanation for SVM and it also can be used to solve the Voronoi diagram problem. In fact, the edited set of Voronoi diagram is the super set of the edited set of Gabriel graph. There exist some relationships between SVM and Gabriel graph.

*Observation #1* From the two experiments, we observe that the number of support vectors is always smaller than the size of the Gabriel graph edited

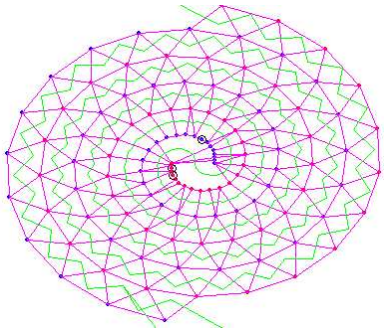

Fig. 4. The boundary found by Gabriel graph

Fig. 5. The boundary found by SVM

set. Suppose support vectors are indeed the subset of the Gabriel edited set, we could improve SVM with Gabriel graph algorithm as follows:

1. Map the data to some other higher, possibly infinite, dimension space, $\mathcal{H}$ and fit an optimal linear classifier in that space.
2. Use the Gabriel graph algorithm to reduce the size of the training data.
3. Using the SVM's optimization steps to obtain the solution to the quadratic problem and find the separating plane.

As a result of applying the steps above, the reduced training data will accelerate the convergence speed of finding the optimal quadratic solution.

*Observation #2* Moreover, the Gabriel graph's formulation is more rigid in the sense that it is not parametrizable so that there is only a single solution to Gabriel graph for a given input set. On the other hand, SVM can be parameterized to achieve different results for a given input set. For example, error penalty can be modified to suit a user's requirement. Nonetheless, the Gabriel graph algorithm provides a stable solution which can be used as a reference to evaluate the average case of some SVM algorithms.

*Observation #3* For the simple case, we can observe that SVM and Gabriel graph could solve the problem perfectly, if we select the proper parameter for SVM.

## V. Conclusion

In this paper, we have demonstrated how the SVM and Gabriel graph can be used for solving the classification problem. Moreover, we have tried to show the relationships among these two algorithms. The Gabriel graph will render a superset of the SVM results through empirical observations. We could try to improve SVM's performance in general by using the Gabriel graph's training data set reduction algorithm. In light of this, we plan to investigate the following in the future:

1. Given the same input data set, compare which algorithm is faster and more accurate in obtaining the optimal decision boundary.
2. Theoretically prove that support vectors are the subset of Gabriel edited set.

## Acknowledgement

This research is supported in part by an Earmarked Grant from the Hong Kong's University Grants Committee (UGC), CUHK #4407/99E.

## References

[1] C. Bradford Barber, David P. Dobkin, and Hannu Huhanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[2] Kristin P. Bennett and Erin J. Bredensteiner. *Duality and Geometry in SVM Classifiers*. Morgan Kaufmann, San Francisco, CA, 2000.

[3] Binay K. Bhattacharya, Ronald S. Poulsen, and Godfried T. Toussaint. *Application of Proximity Graphs to Editing Nearest Neighbor Decision Rule*. International Symposium on Information Theory, Santa Monica, 1981.

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

[5] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a Library for Support Vector Machines (Version 2.31)*. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001.

[6] Don Hush and Clint Scovel. Polynomial-time decomposition algorithms for support vector machines. Technical report, Los Alamos National Laboratory, USA, 2000.

[7] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. *A Fast Iterative Nearest Point Algorithm for Support Vector Machine Classifier Design*. Technical Report TR-ISL-99-03, Intelligent Systems Lab, Dept. of Computer Science and Automation, Indian Institute of Science, Bangalore, India, 2000.

[8] V. N. Vapnik. *Estimation of dependencies based on empirical Data*. (in Russian), Nauka, Moscow, 1979.

[9] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, Heidelberg, Germany, 1995.