Support vector machines for classification.

Rong Fan

SUPPORT VECTOR MACHINES FOR CLASSIFICATION

A Thesis Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

University of Nebraska at Omaha

by

Rong Fan

July 23, 2001

THESIS ACCEPTIONCE

Acceptance for the faculty of the Graduate College, University of Nebraska, in partial fulfillment of the Requirements for the degree (Master of Science), University of Nebraska at Omaha.

Committee

[illegible] 7/23/01

[illegible] 7/23/01

[illegible] 7/23/01

Chairperson [illegible]

Date 7/23/01

SUPPORT VECTOR MACHINES FOR CLASSIFICATION

Rong Fan, MS

University of Nebraska, 2001

Advisor: Professor Zhu, Qiuming

Statistical Learning Theory now plays a more active role: after the general analysis of learning processes, the research in the area of synthesis of optimal algorithms was started. These studies, however, do not belong to history. They are a subject of today's research activities.

Vladimir Vapnik (1995)

As the research of many different learning and function estimation problem develops, a new technique, called a Support Vector Machine (SVM), has been introduced to solve those problems.

In this thesis, we describe a new SVM learning machine that is relatively simple, and easy to implement. A Support Vector Machine (SVM) is a very specific kind of learning machine, trained by the use of kernel functions. The idea behind it is that most data points are ignored, only a small amount of data points, called support vectors, are involved to determine the hyperplane, which separates the data points into different classes. There could be several hyperplanes, but only one optimal hyperplane that separates two class data sets with the largest distance exists.

The learning approach to training SVMs in this thesis comes from perceptron theory, which does a finite number of calculations for adjusting the coefficients and learning functions. The optimal hyperplane is obtained by means of SVMs trained by above learning approach, and then applies for the classification of data sets.

When considering a linear classifier, the optimal hyperplane is obtained by maximizing the distance between the margins of the training sets. For a non-linear classifier, we map the input space into the expanded space, in which the parameters of the input set are linear, by choosing an appropriate kernel function. With respect to the non-separable case, the problem has led to the technique of the "soft-margin", a procedure aimed at extending the large margin algorithms to the noisy areas by permitting a trade-off between accuracy and margin.

Key words: Support Vector Machine, hyperplane, learning machine, classification, optimal solution, margin of solution, training sets

Acknowledgements

Let me say how grateful I am to all of my committee members:
Dr. Qiuming Zhu, Dr Zhengxin, Dr. Hassan Farhat, Dr. John Konvalina, they supported me and taught me, I am so proud to be a student of them, without them, I wouldn't finish my entire graduate program successfully. Particular thank my advisor Dr. Qiuming Zhu, who not just gave me support, also gave me tremendous courage to learn, to challenge myself. Without his encouragement, patient guidance, and constant support, I wouldn't have done my thesis. He also taught me how to use our knowledge and talent to serve the world.

I want to thank my family for their love, for trusting me and supporting me unconditionally. I am very thankful to all of my friends for the great understanding and help.

Again, thank all of my supervisory committee members for their valuable suggestions to my thesis.

Contents

Contents

List of Figures

Chapter 1: Introduction

1.1 Basic Definitions

A *classifier* is a hyperplane that separates data into different categories or classes. A *trainable classifier* is a classifier that may make its performance better in response to information it receives and tasks it takes. *Training* is the process by which the parameters of a function are adjusted in response to categories or classes. A *training procedure* is a training algorithm that implements the training process.

Learning is the process of a system's performance from one level to another. Learning is often related to feedback, and provides a method by which people can manage their technologies. Also learning means that certain machines may adapt automatically to changing environments. A *Learning machine* can be thought of as a set of functions implementing an induction principle, or an algorithmic procedure for implementing the induction principle on the given set of functions.

Recall that we defined a classifier as a hyperplane that separates data into categories. These data are often structured as vectors in *feature space*. Every vector in this space is called a *feature vector*. The fact that each vector in a given vector space can be expressed as a unique linear combination of the vectors of a specific basis suggests that we write the elements of an arbitrary vector space as a coordinate vector whose components are the coefficients in the linear combination (Hugh. G. Campbell. 1997).

The feature vectors in a given class occupy a region in feature space that we call a *class region*. When the class regions don't overlap, the classes are said to be *separable*, and to have the property of *separability*. If, for every class region, a hyperplane can be placed so that it separates that region from all other class regions, the classes are said to be *linearly separable*. More details are discussed in chapter 4. Classes that are not linearly separable are called *nonseparable classes*.

The *training set* is the set of feature vectors that are used as input data during the training procedure. The set of *discriminant functions* $\{g_i(x)\}$ determine the decision hyperplanes:

$$\Re_j = \{x \mid g_j(x) \geq g_i(x) \quad \forall i\}.$$

Where $g_j(x)$ is of the form

$$g_j(x) = w_j^T x + w_{j0},$$

Where w_j^T denotes the transpose of a *weight vector* w_j, when $g_j(x)$ is a *linear discriminant function* as above the classifier is referred to as a *linear classifier*. (Jack Sklansky. Gustav N. Wassel. 1998)

1.2 Prior Knowledge in Support Vector Machines

A Support Vector Machine (SVM), as a kind of learning machine, is a new approach to pattern classification. It consists of a very specific set of algorithms, characterized by the use of kernel functions. As it makes sure to give good performance, therefore it has been applied to various tasks. The foundations of Support Vector Machines have been developed by Vapnik (1995) and are getting popularity as their many attractive features, and promising empirical performance. The term "SVM" is typically used to describe classification with support vector methods. The term "support vector regression" is used to describe the regression process using support vector methods. In this thesis we prefer it to refer to both classification and regression methods.

The approach of SVM has recently been introduced as a new technique for solving various function estimation problems, including the pattern recognition and regression estimation problems, i.e, to problems of finding the indicator function $y = f(x)$ given by its measurements y_i at some vectors x_i,

$$(x_1, y_1), \ldots, (x_\ell, y_\ell).$$

It can be applied for pattern recognition (i.e. to estimate indicator functions), or for regression (to estimate real-valued functions), and for solving linear operator equations. (Vladimir Vapnik, 1997)

In pattern classification, very good results have been reported in (Edgar Osuna, Robert Freund, and Federico Girosi, 1997). It also has been applied to a speaker identification task (M.Schmidt, 1996) and reported that SVM gives slightly better performance than the modified Gaussian system on the difficult Switchboard task.

When SVM is used for two-group classification problems, it attempts to separate points belonging to two given datasets in n-dimensional real space R' by a nonlinear surface defined by a kernel function. The nonlinear surface in R' is typically linear in its feature space (O.L.Managasarian, 1998). Therefore, it can be represented as a linear function in a higher dimensional space, and the original points of the two sets can be mapped into this dimensional space. Further the two sets can be linearly separated in this dimensional space by choosing a suitable kernel function. Generally speaking, for a given learning task, with a given finite amount of training data, the best generalization performance will be achieved by mapping the input vectors into a high-dimensional feature space through some nonlinear mapping, chosen by the appropriate kernel functions. In this space, an optimal separating hyperplane is constructed. The main idea behind the techniques is to separate the classes with a surface that maximizes the margin between them.

1.3 The Research of This Thesis

The purpose of this thesis is to study how to classify the separable or non-separable datasets by using Support Vector Machines (SVMs). A technical analysis will be touched and discussed on what SVM solution is optimal and how to define the decision function on the real data sets. We will also describe how support vector training can be practically implemented, and discuss in detail the learning technique which is used to construct SVM solutions, by which the linear or nonlinear hyperplane is constructed for the data sets.

In this thesis we will study the following:

- Using the convex hull algorithm to determine whether the two data sets are linearly separable or non-linear. A set of vectors χ can be convex if a straight-line segment joining any pair of vectors in χ is entirely contained in χ. The convex hull of χ, which we denote by $\zeta\ \chi$, is the intersection of all convex sets containing χ. The two convex hulls are intersected if and only if there exist points (a point), which are inside both of convex hulls.

- Choosing an appropriate mapping function such that it transforms the input space into a new space $\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x})$. Let the training set X be partitioned into two subsets X_1, X_2. In some situations the set X_1, X_2 are not linearly separable in X space, but are linearly separable in a $\xi-$space, where ξ is a function of X and the dimensionality of ξ is larger than X. Suppose that d is the dimensionality of X, and suppose that $\xi = \Phi(x)$, where $\Phi(x) = \begin{bmatrix} \Phi_1(x) \\ \vdots \\ \Phi_r(x) \end{bmatrix}$, $r \geq d$. Then a separating hyperplane may be linear in ξ –space, but not in X –space.

- Training the data sets to obtain the SVMs by choosing a suitable learning function for linear and non-linear situations in the new space. Assume a perceptron with a fixed function Φ and adjustable coefficients α_i, the sum $\sum \alpha_\varphi \varphi(X)$ for X belonging to **F⁺** is positive, and for X belonging to **F⁻** is negative. How can we make this procedure as simple as possible? The first idea coming out is feedback. Since the sum is too small (in **F⁺**), we increase the coefficients, but if the sum is too large (in **F⁻**), we decrease the coefficients. We adjust the coefficients in a reasonable manner, so that the feedback effect is directed properly.

- Measuring the empirical risk error for nonseparable datasets. For the nonseparable case, there is no feasible solution. But we can introduce the separable case to the nonseparable case by introducing positive slack variables ξ_i, $i = 1, \ldots, \ell$ in the constrains of the separable hyperplane functions. Therefore, for an error occurring, the corresponding ξ_i must exceed unity, so $\sum_i \xi_i$ is an upper bound on the number of training errors.

- Constructing the decision function $y = \alpha * z + b$ (z is a vector of the feature space Z, α is a weight vector, and b is a threshold value) through the SVMs to build an optimal hyperplane.

- Expanding such a two-dimensional function (in the pattern recognition case: such as hyperplane) in a high-dimensional space. The derivation of linear decision surfaces in an expanded feature space can be a means for deriving nonlinear decision surfaces in the original feature space. We are interested in principal components of variables, or features that are nonlinearly related to the input variables. To this end, we are computing dot products in the feature space by means of kernel functions in the input space.

- Demonstrating the results of the empirical data in Microsoft Visual C++ environment.

Roadmap of this thesis is followings:

Chapter 1, basic concepts and some prior knowledge of SVMs are introduced. Chapter 2 gives a brief introduction to the SVM principle. In chapter 3, we will discuss the theoretical aspects of hyperplanes, which makes a separation of two different data sets with multiple dimensions. It is presented in several cases, such as, linearly separation, nonlinearly separation, and nonseparation. The algorithmic details are dealt with in chapter 4, which represents 5 algorithms for solving the classification problems of different separating cases. Finally, we show some results of the implementation of the algorithms and analyze the final results.

Chapter 2: Support Vector Machines

2.1 Overview of the problems

Since SVMs are a new type of universal learning machine, they are used widely. For example, for the pattern recognition case, SVMs have been used for isolated handwritten digit recognition (C.Cortes and V.Vapnik, 1995. B.Schőlkopf, C.Burges, and V.Vapnik, 1995. B.Schőlkopf, C.Burges, and V.Vapnik, 1996. B.Schőlkopf, K. Sung, C.Burges, F. Gurisu, P.Niyogi, T. Poggio, and V.Vapnik, 1997), object recognition (V. Blanz, B.Schőlkopf, H.Bűlthiff, C.Burges, V.Vapnik, and T.Vetter, 1996), speaker identification (M.Schmidt, 1996), charmed quark detection, face detection in images (Edgar Osuna, Robert Freund, and Federico Girosi, 1997), and text categorization (T.Joachims, 1997). For the regression estimation case, SVMs have been compared on benchmark time series prediction tests (K. –R. Műller, A. Smola, G. Rätsch, B.Schőlkopf, J. Kohlmorgen, and V.Vapnik, 1997. Edgar Osuna, Robert Freund, and Federico Girosi, 1997), the Boston housing problem (H. Drucker, C. J. C. Birges, L. Kaufman, A. Smola, and V. Vapnik, 1997), and (on artificial data) on the operator inversion problem (V.Vapnik, S. Golowich, and A. Smola, 1996), In most of these cases, the SVM generalization performance either matches or is significantly better than those competing methods.

Although SVMs have good generalization performance, they can be slow in the test phase because of the range of the data, a problem about decomposing procedure addressed in (C.J.Burges, P.Knirsch, and R.Haratsch, 1996. Edgar Osuna, and Federico Girosi, 1998). Recent work has generalized the basic ideas, shown connections to regularization theory, and shown how SVM ideas can be incorporated in a wide range of other algorithms. The problem that drove the initial development of SVMs occurs in several guises- the bias variance tradeoff, capacity control (B.E.Boser, I.M.Guyon, and V.Vapnik, 1992), and overfitting, (D.C.Montgomery and E.A.Peck, 1992).

The Support Vector Machine has been thought of as a capable learning machine for human-computer interfaces. Because it has the ability to handle difficult speech recognition tasks and give nice performance. Based on the principle of Structural Risk Minimization, SVMs have advantages over other classifiers. For a particular problem, the most generalized classifier to be found by embedding the capacity control in the training algorithm.

The algorithm of SVM is elegant as it is simplified to a limited data set, and can be separated linearly in the expanded space. Theoretically the training is guaranteed to converge to a global optimality. This ability to select the training data that defines the classification boundary could have many applications other than in pattern classification. Also the SVM algorithm can be thought of as an alternative training technique for Polynomial, Radial Basis Function and Multi-Layer Perceptron classifiers, in which the weight of the network are found by solving a Quadratic Programming (QP) problem with linear inequality and equality constraints, rather than by solving a non-convex, unconstrained minimization problem, as in standard neural network training techniques. Since the number of variables in the QP problem is equal to the number of data points, when the data set is large, this optimization problem becomes very challenging. It is because the quadratic form is completely dense and the memory requirements grow with the square of the number of data points. In (Edgar E. Osuna, Robert Freund and Federico Girosi, 1997), a decomposition algorithm that guarantees global optimality has been presented, and can be used to train SVMs over very large data sets. The main idea behind the decomposition is the iterative solution of sub-problems and the evaluation of optimality conditions that are used both to generate improved iterative values, and also establish the stopping criteria for the algorithm.

The face detection problem as an application of SVM for solving a pattern classification problem is introduced in (Edgar E. Osuna, Robert Freund and Federico Girosi, 1997). The problem has many important practical applications, and received a lot of attention in recent years. While being a totally new approach, SVM also offers a fresh view to a few conventional classifiers, namely Neural Networks and Gaussian Radial Basis Function (RBF) classifier. So SVM can be incorporated with other model-based approaches, which capture temporal information. This combination could result in a high–performance speech recognition and other intelligent systems.

2.2 Cortes and Vapnik (VC) Dimension

The VC dimension is a well-known scalar value that can be evaluated for any set of functions accessible to a learning machine.

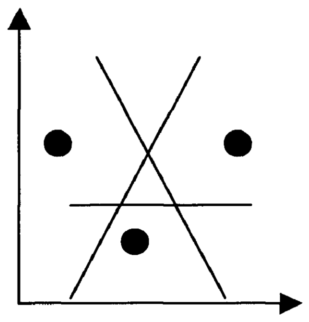


Fig.1. VC-Dimension illustration.

The VC dimension is defined as:
There exists a set of points x_n such that these points can be separated in all 2^n possible configurations, and that no set x_m exists where $m > n$ satisfying this property.

The fig.1 illustrates how three points in the plane can be separated by the set of linear indicator functions. In this case the VC dimension is equal to the number of free parameters. Of course, in general that is not always the case, e.g. the function Asin(bx) has an infinite VC dimension.

2.3 Linearly separable case

If the set of points is linearly separable, i.e. the figure 2, two different data sets, class 1 and class 2, each of them respects to a convex hull, are separated by a linear surface. The goal of the SVM is to find, among the canonical hyperplans that correctly classify the data, the one with minimum norm, or equivalently minimum $\|w\|^2$, because keeping this norm small will also keep the VC-dimension small. It is interesting to see that minimizing $\|w\|^2$ (in this case of linear separable) is equivalent to finding the separating hyperplane for which the distance between the two convex hulls, measured along a line perpendicular to the hyperplane is maximized.

Let S = $\{x_1,\ldots,x_\ell\}$ be a set of ℓ points $x_i \in \mathbf{R}^d$. each x_i is given a label $y_i \in \{-1,+1\}$. S is linearly separable if for some $w \in \mathbf{R}^d$ and $b \in \mathbf{R}$, such that $y_i(w \cdot x_i + b) \geq 1$, for $i = 1,2,\ldots,\ell$. The hyperplane $w \cdot x + b = 0$ is a separating hyperplane. The signed distance d_i of x_i from the separating hyperplane is

$$d_i = w \cdot x_i + \frac{|b|}{\|w\|}$$

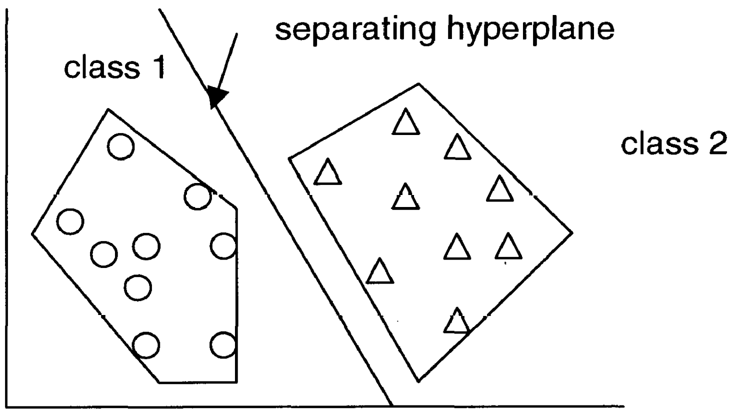


Fig.2. Linearly separable class

A general two-class pattern classification problem is defined as follows:

- Giving a dataset: $(x_1, y_1), \ldots, (x_i, y_i)$ where x_i, for $i = 1, \ldots, \ell$, is a feature vector of length *d* and $y_i = \{-1, +1\}$ is the class label for data point x_i.
- Find a classifier with the decision function $f(x)$, such that $y = f(x)$, where y is the class label for x.
- The performance of the classifier is measured in terms of classification error which is defined in the follow:

$$E(y, f(x)) = \begin{cases} 0, & \text{if} \quad y = f(x). \\ 1, & \text{otherwis.} \end{cases}$$

Let us look at the linear support vector machine. It is based on the idea of hyperplane classifier, or linearly separability.

To learn a linear separating hyperplane classifier:

$$f(x) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

This hyperplane needs to have the maximum separating margin, namely, to find this hyperplane H: $y = \mathbf{w} \cdot \mathbf{x} + b = 0$ and two hyperplanes parallel to it and with equal distances to it.

$$H_1:\ y = \mathbf{w} \cdot \mathbf{x} + b = +1$$

$$H_2:\ y = \mathbf{w} \cdot \mathbf{x} + b = -1$$

With the condition that there are no data points between H_1 and H_2.

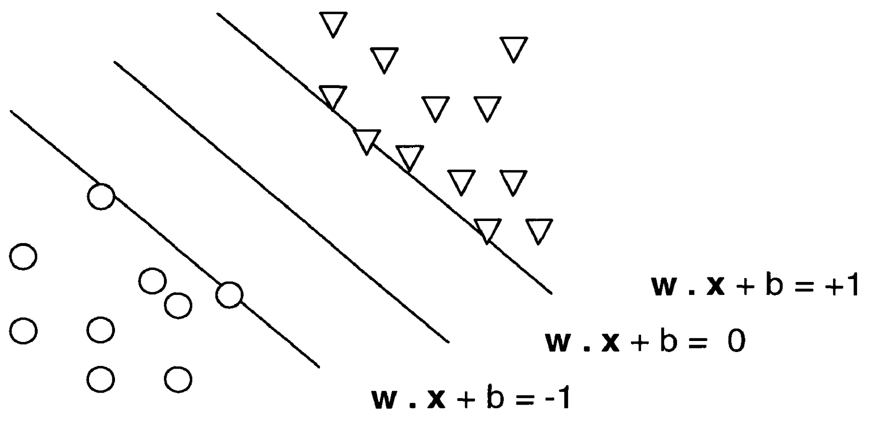


Fig.3.optimal separating hyperplane.

For any separating plane H and the corresponding H_1 and H_2, we can always "normalize" the coefficients vector w so that H_1 will be y = **w . x** + b = +1, and H_2 will be y = **w . x** + b = -1. So there will be some positive data points on H_1 and some negative data points on H_2. These data points are called *support vectors* because only they participate in the definition of the separating hyperplane, and other data points can be removed around as long as they do not cross the planes H_1 and H_2. In other words, there are three hyperplanes,

H: y = **w . x** + b = 0, there is no any data point on this hyperplane;

H_1*:* y = **w . x** + b = +1, only positive data points lie on this hyperplane;

H_2*:* y = **w . x** + b = -1, only negative data points lie on this hyperplane. There are no any data points between H_1 and H_2.

The figure 3 shows the three hyperplanes, the middle one is the optimal hyperplane with minimum norm, and apparently there is no any point between H_1 and H_2. Consequently the hyperplane is determined by a small subset of the data set, namely SVs. The other points could be removed from the data set and recalculating the hyperplane would produce the same answer. Hence SVM can be used to summarize the information contained in a data set.

Suppose we have some hyperplanes that separate the positive data points from the negative data points (a "separating hyperplane"). The points x which lie on the H satisfy **w** • **x** + b = 0, where w is normal to the hyperplane, $\frac{|b|}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin, and ||w|| is the

Euclidean norm of w. Let $d_+(d_-)$ be the shortest distance from the separating hyperplane to the closest positive (negative) data points. Define the "margin" of a separating hyperplane to be $(d_+ + d_-)$. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin. This can be formulated as follows:

All the training data satisfy the following constraints:

$$\mathbf{x}_i . \mathbf{w} + b \geq +1 \quad \text{for } y_i = +1 \qquad (1)$$

$$\mathbf{x}_i . \mathbf{w} + b \leq -1 \quad \text{for } y_i = -1 \qquad (2)$$

These can be combined into one set of inequalities:

$$y_i (\mathbf{x}_i . \mathbf{w} + b) - 1 \geq 0 \quad \forall i \qquad (3)$$

Now consider the points for which the Eq. (1) holds (requiring that there exists such a point that is equivalent to choosing a scale for w and b). These points lie on the hyperplane H_1: $y = \mathbf{w} \cdot \mathbf{x} + b = +1$ with normal w and perpendicular distance from the origin $\frac{|1-b|}{\|w\|}$. Similarly, the points for which the Eq. (2) holds lie on the hyperplane H_2: $y = \mathbf{w} \cdot \mathbf{x} + b = -1$ with normal again w, and perpendicular distance from the origin $\frac{|1-b|}{\|w\|}$. Hence $d_+ = d_- = \frac{1}{\|w\|}$ and the margin is simply $\frac{2}{\|w\|}$. Note that H_1 and H_2 are parallel (they have the same normal) and that no training points fall between them. Thus we can find the pair of hyperplane which gives the maximum margin by minimizing $||w||^2$, subject to constraints (3). The expected solution for a typical two-dimensional case looks like the figure 4, those marked points are support vectors.

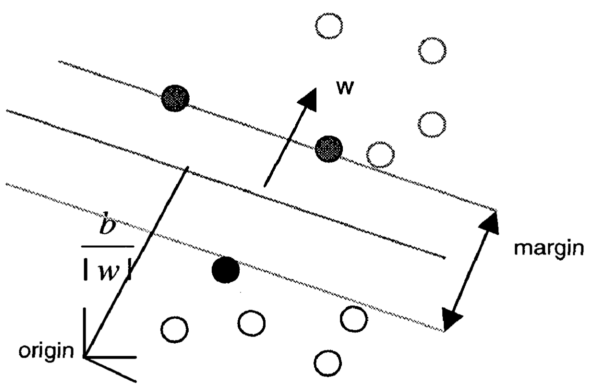


Fig.4. Linear separating hyperplane for separable case.

Consider the above example again, there are many possible linear classifiers that can separate the data, but there is only one that maximizes the margin (maximizes the distance between it and the nearest data point of each class). This linear classifier is termed the *optimal separating hyperplane*. Obviously, we expect this boundary to generalize well as opposed to the other possible boundaries.

2.4 The optimal separating

Let us consider the problem of separating the set of training vectors belonging to two separate classes,

$$(x_1, y_1),\ldots,(x_\ell, y_\ell), \quad x_\ell \in \mathrm{R}^n, \quad y_\ell \in \{+1, -1\}, \tag{4}$$

with a hyperplane

$$(w \cdot x) + b = 0. \tag{5}$$

The set of vectors is said to be optimally separated by the hyperplane if it is separated without error and the distance between the closest vectors to the hyperplane is maximal (Steven Gunn, 1998). Such as figure 5, there is no data crossing the hyperplane, and sitting in the set in which the data does not belong to, also maintaining the maximal distance between the two sets.

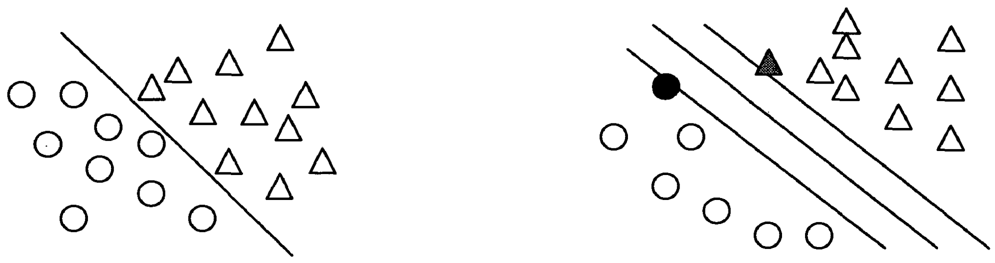


Fig.5. Optimal separating hyperplane without error.

The fact that the optimal separating hyperplane is determined by the SVs is most remarkable. We know, the number of support vectors is usually relatively small, using a few points to replace the whole data set and the same hyperplane is exactly what we want. Because the support vectors summarize all the information contained in the data set. There are two classifiers, one is hard classifier, and another is soft classifier.

The hard classifier is then,

$$f(x) = \mathrm{sgn}(w \cdot x + b).$$

The soft classifier may be used to linearly interpolate the margin,

$$f(x) = h(w \cdot x + b) \quad \text{where } h(x) = \begin{cases} -1 & x < -1 \\ x & -1 \le x < 1 \\ 1 & x = 1 \end{cases}$$

Chapter 3. Support Vector Algorithm

3.1 Introduction

The Support Vector Machine (SVM), as a generalized linear classifier with a maximum-margin fitting criterion, provides regularization that helps the classifier generalizing better. The classifier tends to ignore many of the features. For example, a classification boundary is explored by SVM to allow the largest possible margin of errors. The goal is to minimize the expectation of the out of point error of a learning machine. We set up a hierarchy of function spaces and choose the space with smallest complexity that can attain the desired training error. SVMs use linear hyperplanes as the hierarchy of functions in which "learning" takes place. In order to extend the method to the nonlinear domain, we use the similar method in some non-linear transformed space where certain linear hyperplanes will correspond to a non-linear function in the original space. It turns out that the expected test error is related to the expected number of 'Support Vectors' (certain particularly informative members of the data). So we can get the benefits of non-linearity without much overfitting. The virtue of this algorithm is that it performs well even with a few training data points, as can often be the case. It is expected that such a method would perform well even if the data were noisy.

The SVM algorithm is divided into four steps. In the first step, a structure of decision function is generated which is sufficiently simple to admit the formulation of a bound on their VC-dimension. Based on this result, the optimal margin algorithm minimizes the VC-dimension for this class of decision functions in the second step. This algorithm is then generalized in two steps in order to obtain SV machines: non-separable classification problem are dealt with in the third steps, and nonlinear decision functions, retaining the VC-dimension bound are described in fourth step.

3.2 Existing Approaches

The extension from the binary two-class problem to K classes is an important question for the support vector machine approach. (Ulrich H. –G. Kerβel, 1997) Investigated pairwise classification as an alternative to the often used approach "one class versus all others". The idea of pairwise classification fits perfectly to the borderline-based adoption of the support vector machine. Not just the high-dimensional classifier deserves attention, also the simple linear pairwise support vector machine showed good recognition results with extremely low classification requirements. For the digit recognition (Ulrich H. –G. Kerβel, 1997) the expense compares to a multi-reference Euclidean-distance classifier having $(K - 1)/2 = 4.5$ reference vectors per class. The linear pairwise support vector machine provides an algorithm to adapt for a given sample set a piecewise linear classifier with at most $K * (K - 1)/2$ hyperplanes, which are optimal placed in respect to the error rate. The pairwise support vector machine with its strong discrimination possibilities between class pairs and the ease of combination of these votes allows quite interesting classifier designs, such as using different classifiers depending on the difficulty to separate the given classes or such as pre-classification by a linear classifier and separation between the top votes by a more elaborate method. Support Vector Machines using "ANOVA Decomposition" (Mark O. Stitson, Alex Gammerman, Vladimir Vapnik, Volodya Vovk, Chris Watkins, Jason Weston, 1997). Kernels (SVMD) are another way of imposing a structure on multi-dimensional kernels that are generated as the tensor product of one-dimensional kernels. A SVM using ANOVA decomposition kernels gave a better result than a SVM using other kernels on the Boston housing data and performed more reliably. The low variance observed also indicates a slightly more stable method than with the other kernels. SVM using ANOVA decomposition kernels also yields results that are better than results with other regression methods on the same data. ANOVA decomposition is applicable to many other kernels such as Fourier expansions, Hermite polynomials and Radial Basis Functions.

Support Vector Machine algorithms have been successfully applied to elimination problems. (K. P. Bennett, J. A. Blue, 1997) examined how three key ideas from SVMs can be extended to two-class decision trees (DTS). The key ideas are formulating the problems using structural risk minimization, solving the dual problem, and using nonlinear transformation of the input space to construct nonlinear discriminants. By making simple changes in a nonlinear transformation of the input space, the decisions in the tree can be linear discriminants, polynomials, radial basis functions, neural networks, or any combination of the above. The primary question is: given the underlying structure of a DT, the number and types of decisions, and the classification of the leaves, what is the optimal DT?

The first key idea of the SVM's algorithm is the formulation of the problem using the structural risk minimization principle (SRM). By SRM, the classifier must both reduce the expected classification error and the confidence interval to ensure good generalization. The more details are discussed in chapter 4. Geometrically, this corresponds to pushing apart two parallel planes, one supporting class A and one supporting class B; the wider the margin, the smaller the VC-dimension of the resulting classifier. By widening the classification margin, the confidence interval for the classification error is reduced.

The second key idea from SVMs is the use of the dual problem. An equivalent dual problem can be constructed that can be efficiently solved even for problems with very high input dimension. A Lagrangian dual variable α_i is defined for every constraint in the primal problem. Any point x_i with $\alpha_i > 0$ is a support vector. A global mathematical programming technique is applied to train the support vectors.

The third key idea is using convolutions to construct nonlinear discriminamts. A nonlinear function, Φ: $R^n \rightarrow R^N$ where N>> n, maps the input vectors x to a new space and then constructs a linear discriminant in the new space. Figure 6 illustrates a two-layer mapping, to reach a leaf, a point must satisfy two linear inequalities (assume A^1, B^1 , A^2, B^2 are classified Correctly). Each layer linearly matches to its ancient, and the data in the bottom layer are eventually separated correctly.

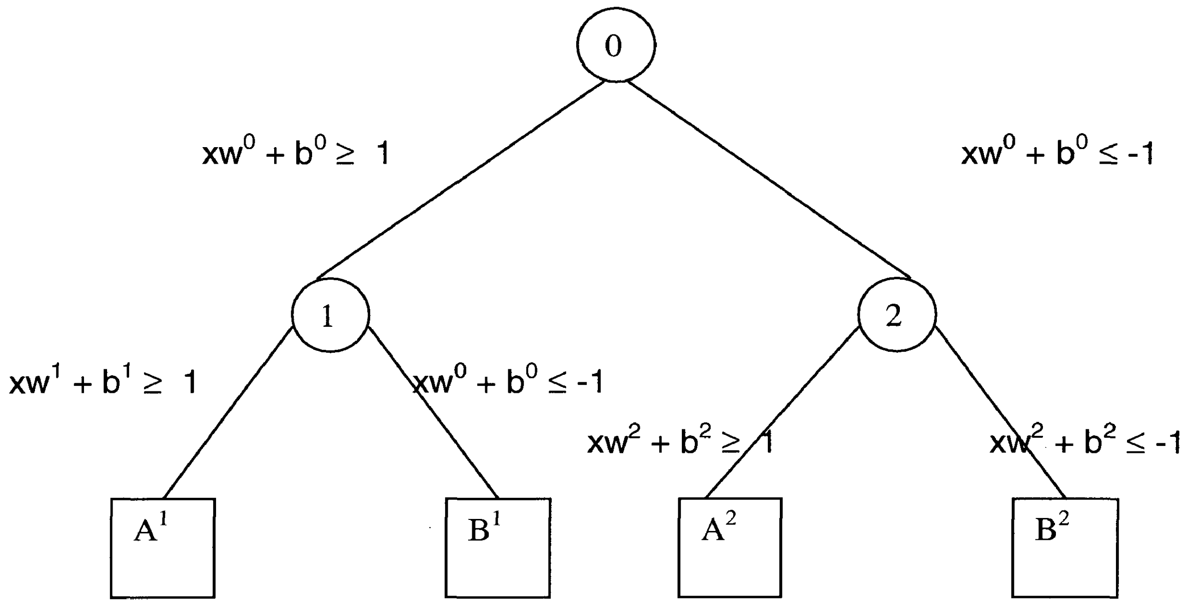


Fig.6. two-layer mapping.

3.3 Structure on the Set of Hyperplanes

Since the SVM algorithm is based on a structure defined on the set of separating hyperplanes. First note that given a dot product space F and a set of pattern vectors $z_1,\ldots,z_\ell \in F$, any hyperplane can be written as $\{z \in F : (w \cdot z) + b = 0\}$.

In this formulation, there still have the freedom to multiply w and b with the same nonzero constant. However the hyperplane corresponds to a canonical pair $(w,b) \in F \times R$ if additionally requiring

$$\min |(w \cdot z_i) + b| = 1,$$

such that the point closest to the hyperplane has a distance of $\frac{1}{\| w \|}$. The margin between the two classes, measured perpendicular to the hyperplane, is at least $\frac{2}{\| w \|}$. The possibility of introducing a structure on the set of hyperplanes is based on the following result:

Let R be the radius of the smallest ball $B_R(\alpha) = \{z \in F: \|z-\alpha\| < R\}$ $(\alpha \in F)$ containing the points $z_1, \ldots, z_r$, and let

$$f_{w,b} = sgn\ ((w \cdot z_i) + b)$$

be canonical hyperplane decision functions defined on these points. Then the set $\{f_{w,b}: \|w\| \leq A\}$ has a VC-dimension h satisfying:

$$h < R^2 A^2 + 1.$$

Due to $\|w\|<A$, we can get VC-dimensions which are much smaller than N_F, where N_F is the dimension of F. This enables us to work in very high dimensional spaces, because that the risk bound dose not explicitly upon N_F, but on the VC-dimension. So the hyperplane decision functions should be constructed such that they maximize the margin, and at the same time separate the training data with as few exceptions as possible.

3.4 Optimal Margin Hyperplane

The exploretion of optimal margin hyperplane is defined as the follows:

As we discussed in the previous chapter, a set of points $(z_1, y_1), \ldots, (z_\ell, y_\ell)$, $z_i \in F$, $y_i \in \{\pm 1\}$, defines a decision function

$$f_{w,b}(z_i) = \mathrm{sng}((w \cdot z) + b)$$

with property

$$f_{w,b}(z_i) = y_i , \quad i = 1, \ldots, l. \tag{1}$$

if this function exists (the non-separable case shall be dealt with in the next step), it implies

$$y_i \cdot ((w \cdot z_i) + b) \geq 1, \quad i = 1, \ldots, l. \tag{2}$$

the separating hyperplane can be found by minimizing

$$\tau(w) = \frac{1}{2} \|w\|^2 \tag{3}$$

Subject to (2).

Thus there are two reasons for introducing a Lagrangian formulation of the problem. The first is that the constraints (3) will be replaced by constraints on the Lagrange multipliers themselves, which will be much easier to handle. The second is that in this reformulation of the problem, the training data will only appear in the form of dot products between vectors. So we introduce positive Lagrange multipliers α_i, $i = 1,\ldots,\ell$, one for each of the inequality constraints (2). This gives Lagrangian:

$$L(w,b,\alpha) = \frac{1}{2} \| w \|^2 - \sum_{i=1}^{\ell} \alpha_i \left(y_i \left((w \cdot z) + b \right) - 1 \right) \tag{4}$$

With multipliers $\alpha_i \geq 0$. The Lagrangian L has to be maximized with respect to α_i and minimized with respect to w and b under the condition at the saddle point, the derivatives of L with respect to the primal variables must vanish. This is

$$\frac{\partial L(w,b,\alpha)}{\partial b} = 0 \ ,$$

and

$$\frac{\partial L(w,b,\alpha)}{\partial w} = 0 \tag{5}$$

leads to

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0 \tag{6}$$

and

$$w = \sum_{i=1}^{\ell} \alpha_i y_i z_i \tag{7}$$

The solution vector thus has an expansion in terms of training points. Although the solution w is unique, the coefficients α_i need not be.

According to the Kuhn-Tucker theorem of optimization theory (Hans Paul Künzi, Wilhelm Krelle, Werner Oettli, 1966), at the saddle point only those Lagrangain multipliers α_i can be nonzero which correspond to constraints (2) such that

$$\alpha_i[y_i \cdot ((w \cdot z)+b)-1]=0\,, \quad i=1,\ldots,\ell \qquad (8)$$

the patterns z_i for which $\alpha_i > 0$ are the Support Vectors.

According to (8), the Support Vectors lie exactly on the margin. All remaining points of the training set are irrelevant: their constraint (2) is satisfied automatically, and they do not appear in the expansion (7). Substituting the conditions (6) and (7) into the Lagrangian (4), we derive the dual form of the optimization problem:

maximizing

$$W(\alpha)=\sum_{i=1}^{\ell}\alpha_i-\frac{1}{2}\sum_{i=1}^{\ell}\alpha_i\alpha_j y_i y_j (z_i \cdot z_j) \qquad (9)$$

Subject to the constraints

$$\alpha_i \geq 0\,, \quad i=1,\ldots,\ell \qquad (10)$$

and

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0 \tag{11}$$

then we obtain an expression which can be evaluated in terms of dot products between the pattern to be classified and the Support Vectors,

$$f(z) = \operatorname{sgn}\left(\sum \alpha_i y_i (z_i \cdot z) + b\right). \tag{12}$$

It is interesting to note that the solution has a simple physical interpretation. If we assume that each Support Vectors z_i exerts a perpendicular force of size α_i and sign y_i on a solid plane sheet lying along the hyperplane $w \cdot z_i + b = 0$, then the solution satisfies the requirements of mechanical stability. The constraint (11) translates into the forces on the sheet summing to zero; and (7) implies that the torques $z_i \times \frac{\alpha_i y_i w}{\| w \|}$ also sum to zero. This mechanical analogy illustrates the physical meaning of the term Support Vector.

3.5 Linearly Nonseparable Hyperplane

In practice, a separating hyperplane often does not exist. To allow for the possibility of examples violating (2), Cortes and Vapnik(1995) introduce slack variables

$$\xi_i \geq 0, \qquad i = 1,\ldots,\ell. \tag{13}$$

and a function

$$N(\xi) = \sum_{i=1}^{\ell} \xi_i$$

using relaxed separation constraints

$$y_i((w \cdot x_i) + b) \geq 1 - \xi_i, \quad i = 1,\ldots,\ell. \tag{14}$$

adjust (9) constraints (10)

$$0 \leq \alpha_i \leq \gamma, \quad i = 1, \ldots, \ell. \tag{15}$$

Minimize the above function subject to constraints

$$y_i((w \cdot x_i) + b) \geq 1 - \xi_i, \qquad i = 1,\ldots,\ell,$$

and

$$(w \cdot w) \leq c_n.$$

The method of solution of this quadratic optimization problem is in large sense equivalent to the method used in the separable case: to find the coefficients of the optimal hyperplane

$$w = \sum_{i=1}^{\ell} \alpha_i y_i x_i ,$$

We have to find the parameters α_i, $i = 1,\dots,\ell$ one that maximize the same quadratic form as in the separable case

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$$

under a little bit different constraints

$$0 \le \alpha_i \le C, \ i = 1,\dots,\ell ,$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 .$$

As in the separable case, only some of the coefficients α_i, $i = 1,\dots,\ell$ differ from zero. They determine the support vectors. C is a given value by user.

3.6 Nonlinear Separable Hyperplane

To allow for much more general decision surfaces, we can first nonlinearly transform a set of input vectors $x_1,\dots,x_\ell$ into a high-dimensional feature space by a map Φ: $x_i \mapsto z_i$ and then do a linear separation there. Note that in all of the above, we made no assumptions on the dimensionality of F. we only required F to be equipped with a dot product. The pattern z_i that we talked about in the previous sections thus need not coincide with the input patterns. They can equally well be the results of mapping the original input patterns x_i into a high-dimensional feature space.

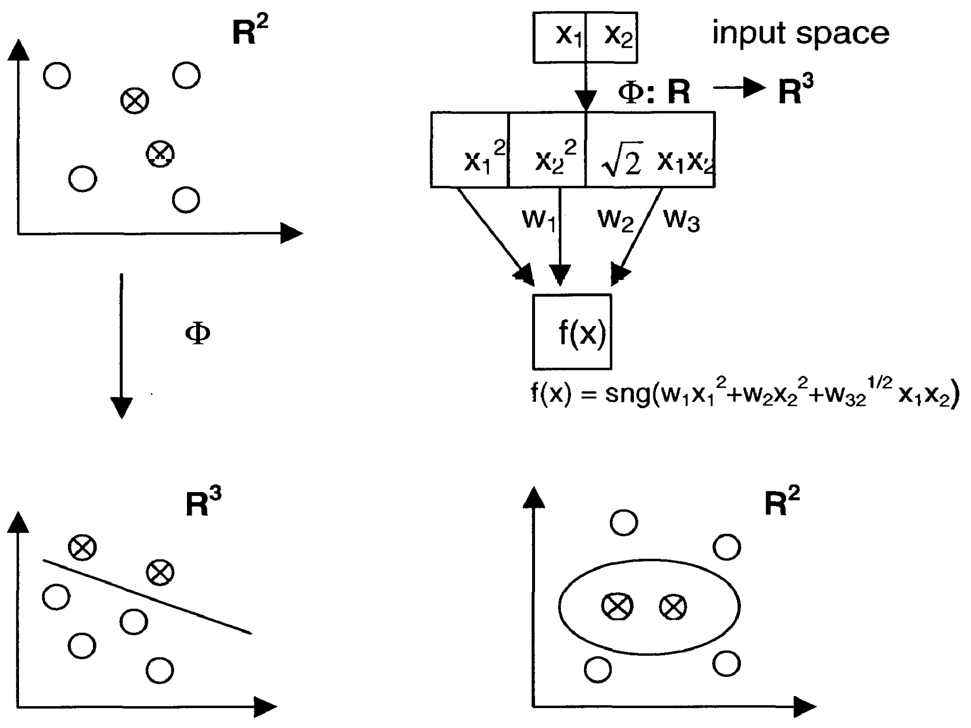
Maximizing the target function (9) and evaluating the decision function (12) then requires the computation of dot products $(\Phi(x)\cdot\Phi(x_i))$ in a high-dimensional space. Under Mercer's conditions (R. Courant and D. Hilbert, 1953) $K(x,y) = \sum_i \Phi(x)_i\Phi(y)_i$, these expensive calculations can be reduced significantly by using a suitable function K such that

$$(\Phi(x)\cdot\Phi(x_i)) = k(x,x_i) \qquad (16)$$

leading to decision functions of the form

$$f(x) = \mathrm{sgn}(\Sigma^{l}_{i=1}\alpha_i y_i \cdot k(x,x_i) + b). \qquad (17)$$

As the result, everything that has been said about the linear case also applies to nonlinear cases obtained by using a suitable kernel k instead of the dot product. Figure 7 shows a procedure that maps the input data (top left) nonlinearly (via Φ) into a higher-dimensional feature space F (here: R^3), and constructs a hyperplane there (bottom left), an SV machine (top right) corresponds a nonlinear decision surface in input space (here: R^2, bottom right).



R^2
Φ
x_1 x_2
input space
Φ: R → R^3
x_1^2 x_2^2 $\sqrt{2}\ x_1x_2$
w_1 w_2 w_3
f(x)
$f(x) = sng(w_1x_1^2+w_2x_2^2+w_{32}^{1/2}x_1x_2)$
R^3
R^2

Fig.7. Nonlinearly transorm R to R^3.

By using different kernel functions, the SV algorithm can construct a variety of learning machines, some of them coincide with classical architectures:

Polynomial classifiers degree d:

$$k(x,x_i) = (x \cdot x_i)^d$$

Radial basis function classifiers:

$$k(x,x_i) = \exp(-\|x-x_i\|^2/c)$$

Neural networks:

$$k(x,x_i) = \tanh(k\cdot(x\cdot x_i) + \Theta)$$

Figure 8 shows a architecture of SVMs. The kernel function k is chosen a prior; it determines the type of classifier (e.g. polynomial classifier, radial basis function classifier, or neural network). All other parameters (number of hidden units, weights, threshold b) are found during training by solving a quadratic programming problem. The first layer weights x_i are a subset of the training set (the Support Vectors); the second layer weights $\lambda_l = y_i\alpha_i$ are computed from the Lagrangain multipliers.

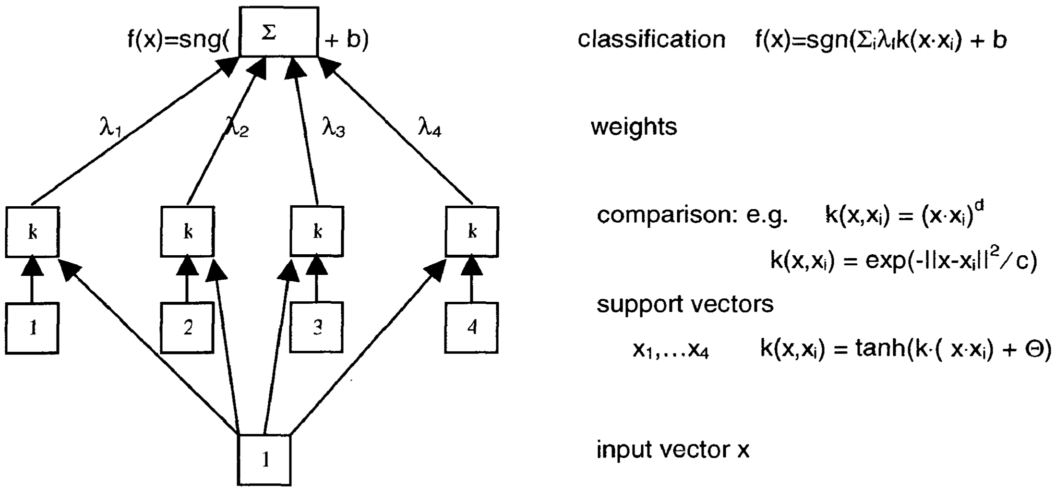

f(x)=sng(Σ + b)
λ1
λ2
λ3
λ4
k
1
2
3
4
1
classification f(x)=sgn(Σ_iλ_ik(x·x_i) + b
weights
comparison: e.g. k(x,x_i) = (x·x_i)^d
k(x,x_i) = exp(-||x-x_i||^2/c)
support vectors
x_1,...x_4
k(x,x_i) = tanh(k·(x·x_i) + Θ)
input vector x

Fig.8. Architecture of SV machines.

To find the decision function (17), maximize (9)

$$W(\alpha)=\sum_{i=1}^{\ell}\alpha_i-\frac{1}{2}\sum_{i=1}^{\ell}\alpha_i\alpha_j y_i y_j k(z_i\cdot z_j) \qquad (18)$$

Subject to the constraints (15) and (11). Since k is required to satisfy Mercer's condition, it corresponds to a dot product in another space (16), thus $K_{ij}:=(y_i y_j k(x_i\cdot x_j))_{ij}$ is a positive matrix, providing us with a problem that can be solved efficiently. To compute the threshold b, for Support Vector x_j for which

$\xi_I = 0$,

we have

$$\sum_{i=1}^{\ell}\alpha_i y_i k(x_i\cdot x_j)+b=y_i$$

Thus, the threshold can for instance be obtained by averaging

$$b=y_i-\sum_{i=1}^{\ell}\alpha_i y_i k(x_i\cdot x_j)$$

over all Support Vectors x_j.with $\alpha_j>0$.

Chapter 4. Approaches Contributed in This Research

4.1. Sequential Minimal Optimization (SMO) Approach for training Support Vector Machines

The method for controlling the generalization ability of learning machines is employed in this research to construct an inductive principle for minimizing the risk error using a small sample of training data.

If the ratio ℓ / h(ratio of the number of training data to the VC dimension of functions of a learning machine) is small, say ℓ / h < 20, then the size ℓ is considered to be small. To make the sample size to be small, we will make use of the bounds for the generalization ability of learning machines with sets of totally bounded non-negative functions,

$$R(\alpha_\ell) \leq R_{emp}(\alpha_\ell) + \frac{B\varepsilon}{2}\left(1+\sqrt{1+\frac{4R_{emp}(\alpha_\ell)}{B\varepsilon}}\right), \tag{4.1}$$

and the bounds for the generalization ability of learning machines with sets of unbounded functions

$$R(\alpha_\ell) \leq \frac{R_{emp}(\alpha_\ell)}{\left(1-a(p)\tau\sqrt{\varepsilon}\right)_+}, \tag{4.2}$$

where

$$a(p) = \sqrt[p]{\frac{1}{2}\left(\frac{p-1}{p-2}\right)^{p-1}},$$

and

$$\varepsilon = 2\frac{\ln \mathrm{N} - \ln \eta}{\ell}.$$

Suppose that the set of functions $Q(z,\alpha_i)$, $1,\ldots,\ell$, contains ℓ elements, and

$$\varepsilon = 4\frac{h(\ln\frac{2\ell}{h}+1) - \ln(\eta/4)}{\ell}$$

if the set of functions $Q(z,\alpha_i)$, $\alpha \in \Lambda$ contains an infinite number of elements and has a finite VC dimension h. Each bound is valid with probability of at least $1-\eta$.

4.1.1 Empirical Risk Minimization (ERM)

For the pattern recognition problem, we use the sample to find the function from the set of admissible functions that minimizes the probability of error. This is exactly what we want. The reason that ERM uses empirical data better is because it does not depend on a priori information, and there are clear ways to implement it.

Assume we have a learning machine with adjustable parameters ξ. Given a classification task, the machine will adjust its parameters ξ to learn the mapping $x \mapsto y$. It will bring about a possible mapping $x \mapsto f(x,\xi)$, which corresponds to this particular learning machine. The performance of this machine can be measured by the expectation of test error,

$$R(\xi) = \int E(y, f(x,\xi))dF(x,y)$$

This is called expected risk or actual risk. It requires at least an estimate of $p(x, y)$, which is not available for most classification tasks. Therefore, we have to be content with the empirical risk measure, which is defined in the following.

This is just a measure of the mean error over the available training data,

$$R_{emp}(\xi) = \frac{1}{\ell}\sum_{i=1}^{\ell} E(y, f(x,\xi))$$

most training algorithms for learning machines employ ERM, for instance, minimize the empirical error using Maximum Likelihood estimation for the parameters ξ. However, These conventional training algorithms do not consider the capacity of the learning machine and this can cause over fitting, such as, using a learning machine with too much capacity for a particular problem.

The Empirical Risk Minimization (ERM) principle can be used to deal with a large sample size. It can be modified by considering the inequalities (4.1) or (4.2). When ℓ / h is large, ε is small. Therefore, the second addend on the right-hand side of inequality (4.1) becomes small. The actual risk is then close to the value of the empirical risk. In this case, the smaller value of the empirical risk is , the smaller value pf the expected risk is.

However, sometimes even ℓ / h is small, but a small $R_{emp}(\alpha_\ell)$ does not guarantee a small value of the actual risk. In this case, to minimize the actual risk $R(\alpha\ )$ we have to minimize the right-hand side of inequality (4.1) or (4.2) simultaneously over both terms. Note, however, that the first term in inequality (4.1) depends on a specific function of the set of functions, while the second term depends on the VC dimension of the whole set of functions. To minimize the right-hand side of the bound of risk (4.1) or (4.2), simultaneously over both terms, we have to make the VC dimension a controlling variable.

4.1.2 Structural Risk Minimization

The goal of Structural Risk Minimization (SRM) [Vapnik, 1995], in contrast with ERM, is to find the learning machine that gives a good trade-off between low empirical risk and small capacity. Two major problems in achieving this goal are involved.

- A measure of the capacity of a particular learning machine or at least an upper bound of the measure is required. For achieving the goal, we need an algorithm to select the desired learning machine. We may divide the entire class of machines into nested subsets with decreasing capacity, One for each subsets. Then we can learn a series of machines by using the ERM principle. Finally the machine that gives the best trade-off can be selected. Of curse, this can be a very difficult task.

- An alternative is to define a learning machine with variable capacity and a corresponding training algorithm that minimizes both the empirical error and capacity of that machine.

The following general principle, which is called SRM inductive principle, is intended to minimize the risk function with respect to both terms, the empirical risk, and the confidence interval (Vapnik and Chervonenkis, 1974).

Let the set S of functions $Q(z,\alpha_i)$, $\alpha \in \Lambda$, be provided with a structure consisting of nested subsets of functions $S_k = \{Q(z,\alpha), \alpha \in \Lambda_k\}$, such that

$$S_1 \subset S_2 \subset \ldots \subset S_n \ldots, \qquad (4.3)$$

As shown in figure 9.

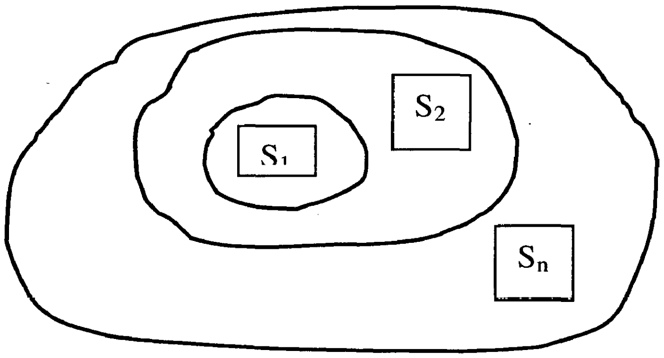


Figure 9. A structure on the set of functions is determined by the nested subsets of functions.

Where the elements of the structure satisfy the following two properties:

1. The VC dimension h_k of each set S_k of functions is finite, and $h_1 \le h_2, \ldots, \le h_n, \ldots$.

2. Any element S_k of the structure contains:

   Either a set of totally bounded functions,

$$0 \le Q(z,\alpha) \le B_{k,} \alpha \in \Lambda_k ,$$

   Or a set of functions satisfying the inequality

$$\sup_{\alpha \in \Lambda_k} \frac{\left(\int Q^p(z,\alpha) dF(z)\right)^{\frac{1}{p}}}{\int Q(z,\alpha) dF(z)} \le \tau_k, \quad p > 2 \tag{4.4}$$

   for some pair (p, τ_k).

We call this structure an admissible structure. For a given set of points $z_1, \ldots, z_\ell$ the SRM principle chooses the function $Q(z, \alpha_\ell^k)$, which minimizes the empirical risk in the subset S_k for which the guaranteed risk is minimal.

A trade-off between the quality of the approximation of the given data and the complexity of the approximating function are defined by the SRM principle. When the subset index n increases, the minima of the empirical risks decrease, however, the term corresponding to the confidence interval increases. The SRM principle takes both factors into account by choosing the subset S_n for which minimizing the empirical risk gives the best bound on the actual risk.

4.2. Convex Hull Approach

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior. Let's denote the convex hull of Q by $CH(Q)$. Intuitively, we can think of each point in Q as being a nail sticking out from a board. The convex hull is then the shape formed by a tight rubber band that surrounds all the nails. Figure 10 shows a set of points and its convex hull.

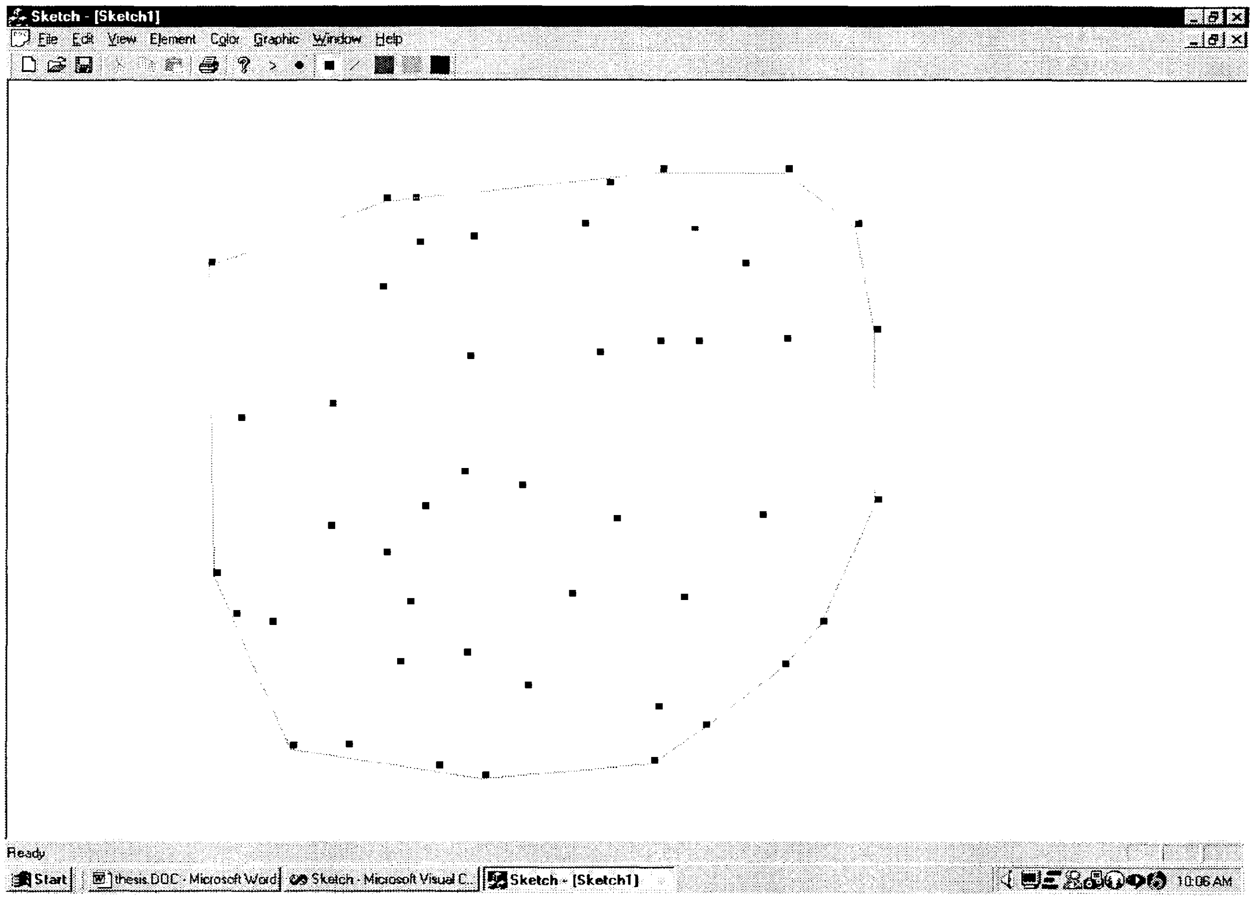

Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
Ready
Start
thesis.DOC - Microsoft Word
Sketch - Microsoft Visual C...
Sketch - [Sketch1]
10:06 AM

Figure 10. A set of points Q with its convex hull $CH(Q)$

In this thesis, we will present the algorithm, Graham's scan (Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rovest, 1989), which computes the convex hull of a set of n points.

We use Graham's scan to solve the convex hull problem by maintaining a queue Q of points. Each point of the input set Q is pushed once onto the queue, and the points that are not vertices of $CH(Q)$ are eventually deleted from the queue, when the algorithm terminates, queue Q contains exactly the vertices of $CH(Q)$, in counterclockwise order of their appearance on the boundary. Figure 11a-h shows the procedure.

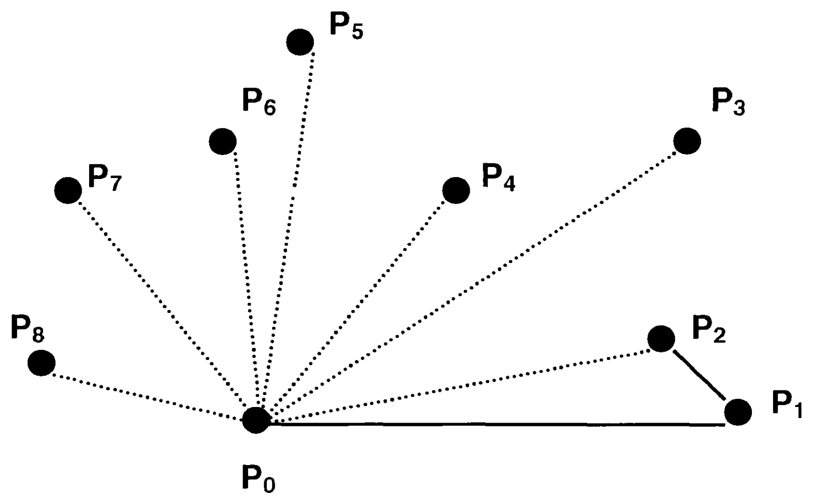


Figure 11.a. Sorting the set of points according to the polar angle in counterclockwise order relative to p_0.

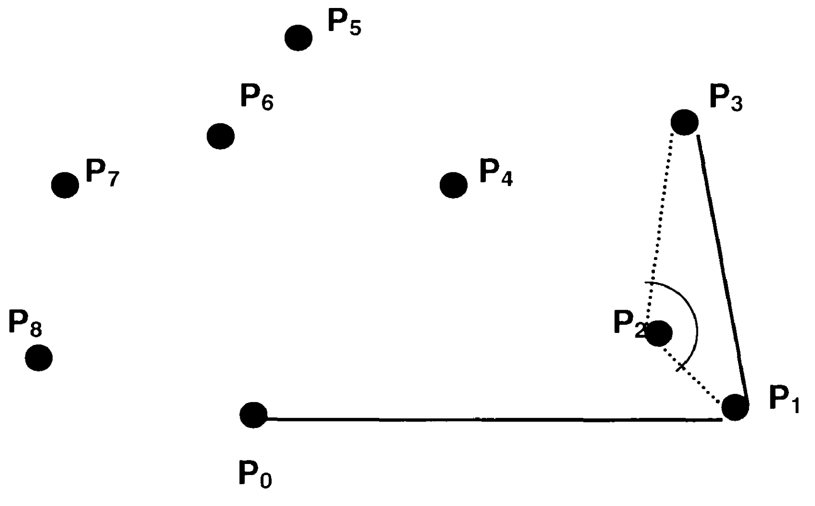


Figure 11.b. When the point makes a nonleft turn, then the point will be deleted from queue Q.

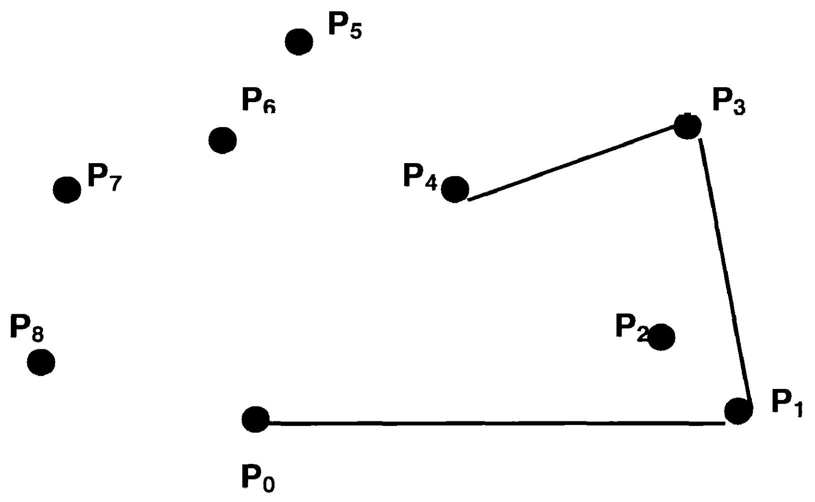


Figure 11.c. When the point makes a left turn, then the point will be kept in the queue Q.

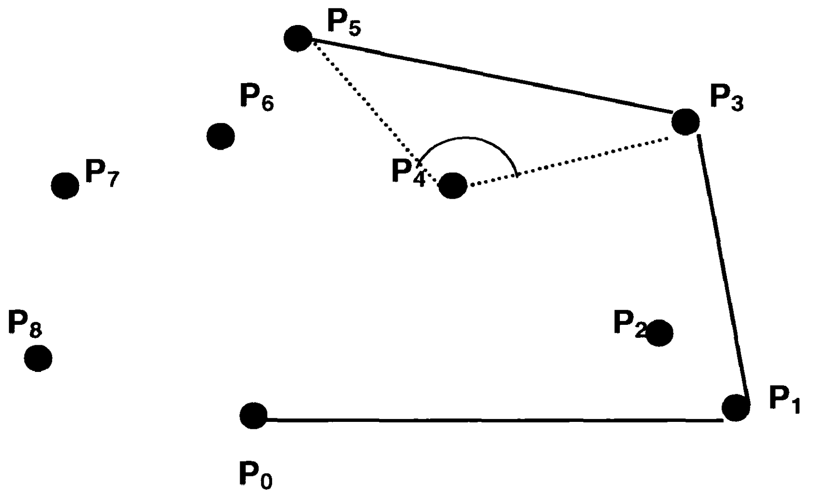


Figure 11.d. When the point makes a nonleft turn, then the point will be deleted from queue Q.

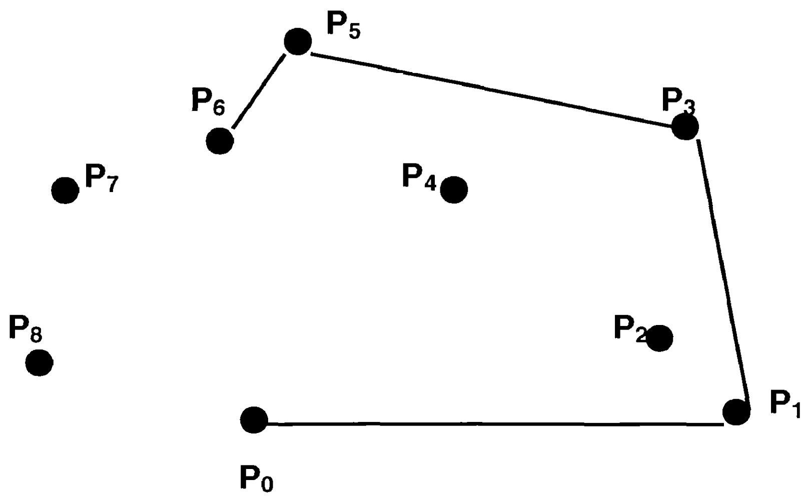


Figure 11.e. When the point makes a left turn, then the point will be kept in the queue Q.

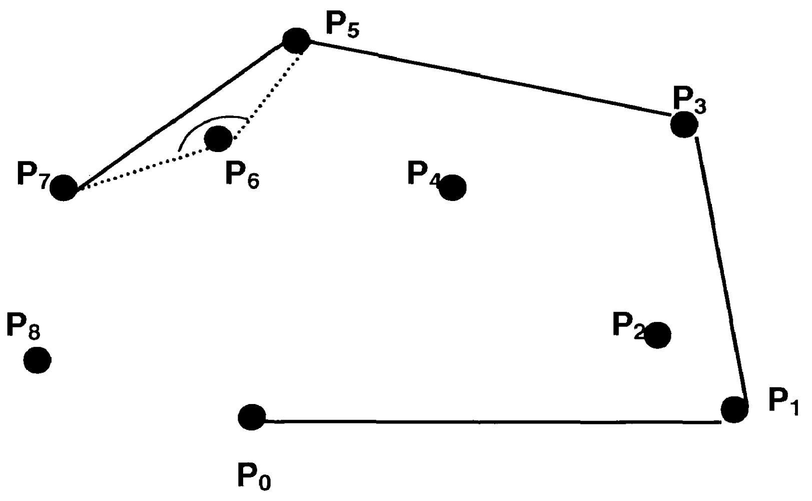


Figure 11.f. When the point makes a nonleft turn, then the point will be deleted from queue Q.

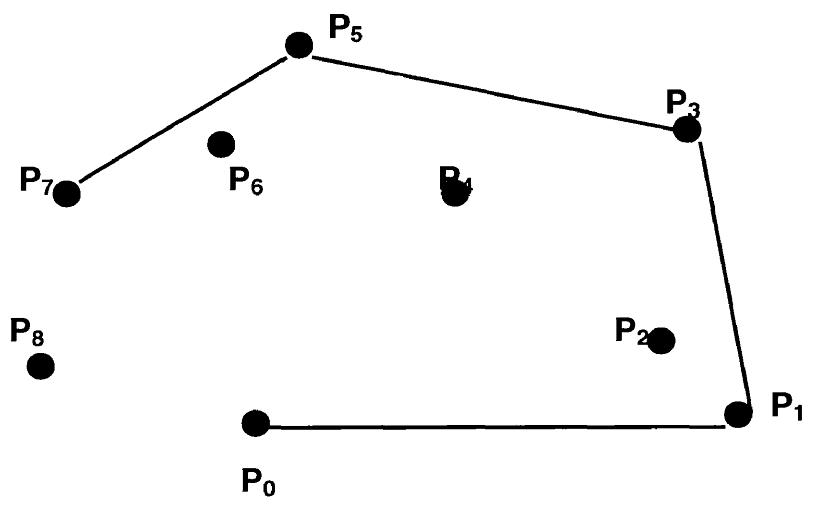


Figure 11.g. When the point makes a left turn, then the point will be kept in the queue Q.

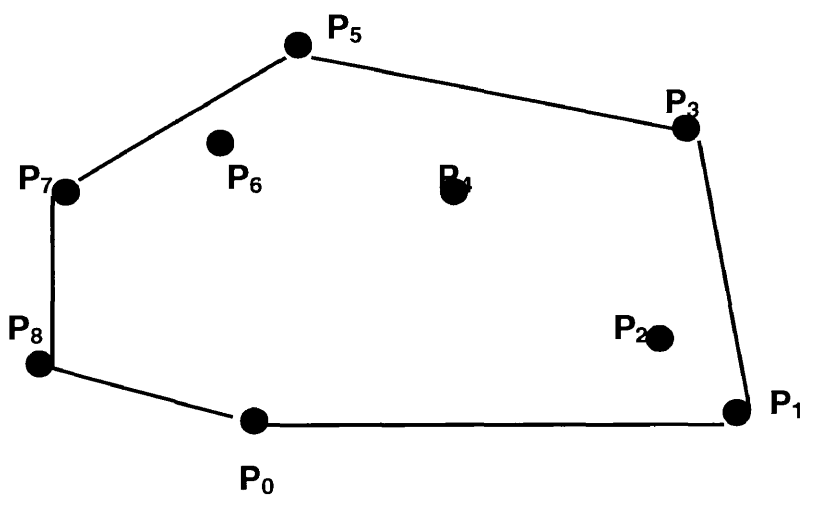


Figure 11.h. convex hull consists of the points that are kept in the queue Q.

4.3 Convex Hulls Intersection Approach

If two convex hulls are intersected, there must be a point (or points) that exists inside both of convex hulls. As shown in the figure 12, the two convex hulls overlap.

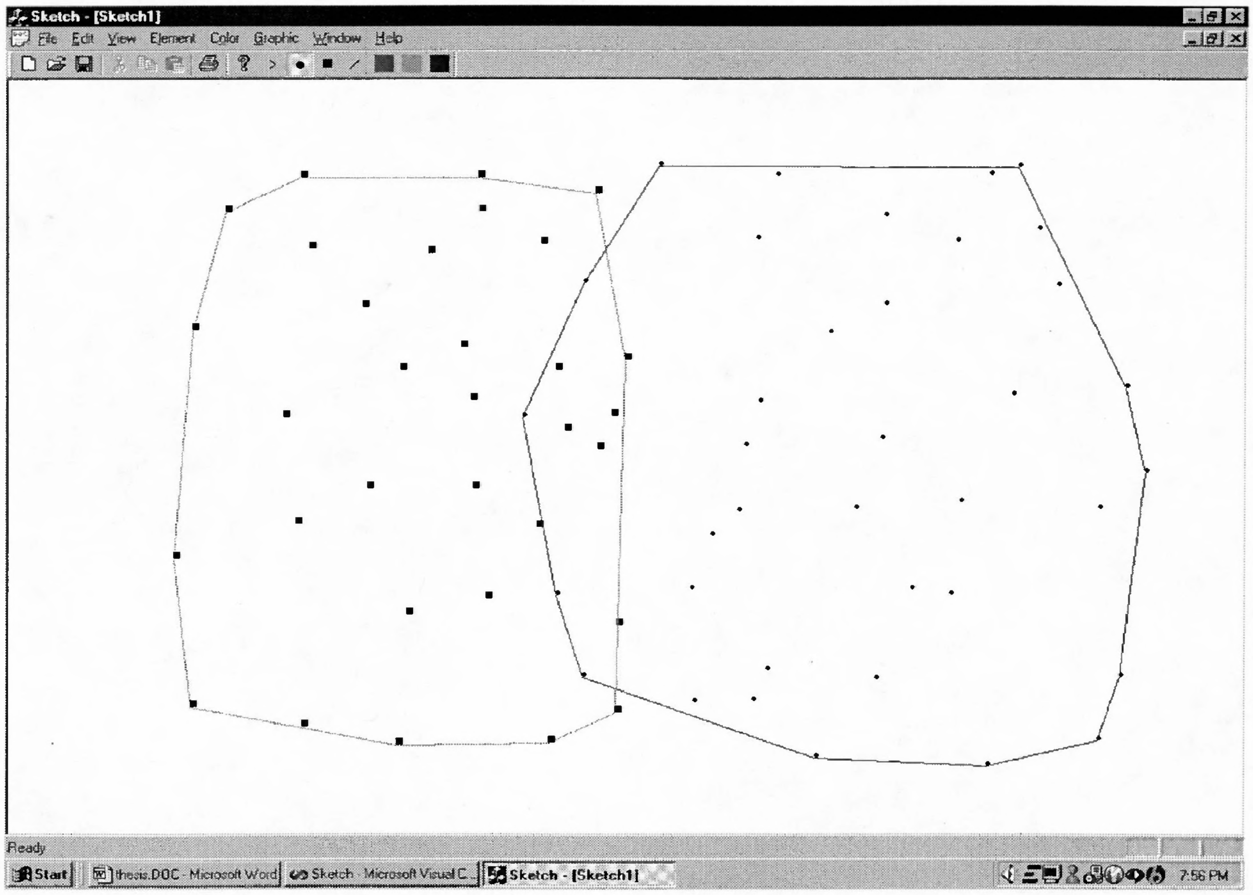

Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
Ready
Start
thesis.DOC - Microsoft Word
Sketch - Microsoft Visual C...
Sketch - [Sketch1]
7:56 PM

Figure 12. Two convex hulls are intersected.

An instance of the point-in-convex hull problem need be solved: Given a convex hull (dataset) P and a query point q which belongs to another convex hull (dataset), is $q \in P$?

We will use the LeftOn test method for each edge of the given convex hull to determine whether or not a query point is inside the given convex hull.

A directed line is determined by two points given in a particular order (p_0, p_1). If a query point q is to the left of the line determined by (p_0, p_1), then the triple (p_0, p_1, q) forms a counterclockwise circuit. In another words, q is to the left of (p_0, p_1) if and only if the area of the counterclockwise triangle, $A(p_0, p_1, q)$ is positive. See the figure 13 that both triangles have positive areas. Therefore we may implement the Left predicate by returning a true (positive) or false (negative) to Area (). If the point q is always to the left of the each edge of the given convex hull, then the point q must be inside the given convex hull, meanwhile, it also demonstrates that the two convex hulls are intersected each other.

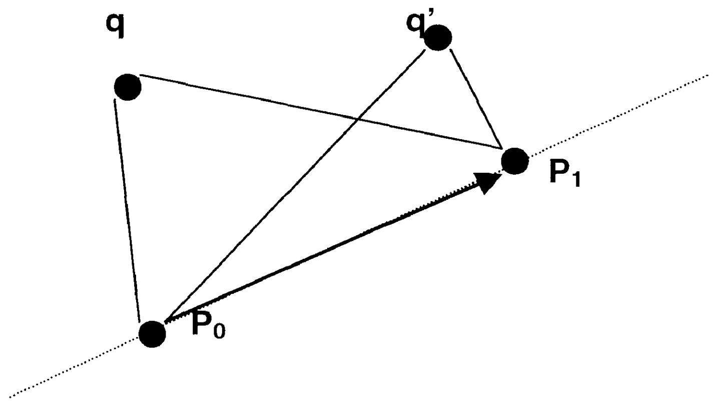


Figure 13. q is left to p_0p_1 iff $\Delta p_0 p_1 q$ has positive area; also $\Delta p_0 p_1 q'$ has positive area.

4.4 Learning Approach

To implement the SVM inductive principle in learning approach, we have to minimize the risk in a given set of functions by controlling two factors: the value of the empirical risk and the value of the confidence interval. In this section, we describe learning approach for classification.

The generalization ability of learning machines is to control the generalization ability during learning processing. According to this theory, to guarantee a high level of generalization ability of the learning process, therefore the bound (4.1) can be rewritten in the following form

$$R(\alpha_\ell^k) \leq R_{emp}(\alpha_\ell^k) + \Phi\left(\frac{\ell}{h_k}\right), \tag{4.5}$$

where the first term is the empirical risk, and the second term is the confidence interval.

During the design of the learning machine we define a set of admissible functions with some VC dimension h^*. For a given amount ℓ of training data, the value h^* determines the confidence interval $\Phi(\frac{\ell}{h^*})$ for the machine. Choosing an appropriate element of the structure is therefore a problem of designing the machine for a specific amount of data. During the learning process the machine minimizes the first term of the bound of (4.5). If for a given amount of training data we design too complex a machine, the confidence interval $\Phi(\frac{\ell}{h^*})$ will be large. In this case even if we could minimize the empirical risk down to zero the number of errors on the test set could still be large. This phenomenon is called over fitting.

In order to avoid over fitting we have to construct machines with small VC dimension. On the other hand, for the set of functions with a small VC dimension, it will be hard to approximate the training data. To achieve a small approximation error and at same time maintain a small confidence interval, we have to choose the structure of the learning machine to reflect a prior knowledge about the problem at hand. Thus, to solve the problem at hand by these types of machines, we first have to find the appropriate structure of the learning machine, and second find the function in this machine that minimizes the number of errors on the training data. This approach, which minimizes the right-hand side of inequality (4.5) can be describe as follows:

- Keep the confidence interval fixed (by choosing an appropriate construction of machine) and minimize the empirical risk.
- Keep the value of the empirical risk fixed (say equal to zero) and minimize the confidence interval.

Let's consider the problem of minimizing the empirical risk on the set of linear indicator functions

$$f(x,w) = \operatorname{sgn}\{(w \cdot x)\}, \quad w \in R^n, \qquad (4.6)$$

Where $(w \cdot x)$ denotes an inner product between vectors w and x. Let

$$(x_1, y_1), \ldots, (x_\ell, y_\ell)$$

be a training set, where x_j is a vector, and $y_j \in \{1,-1\}$, $j = 1,\ldots,\ell$.

Our goal is to find the weight vectors of parameters w_0, which minimize the empirical risk function

$$R_{emp}(w) = \frac{1}{\ell}\sum_{j=1}^{\ell}\left(y_i - f(x_j, w)\right)^2. \qquad (4.7)$$

If the training set is separable without error (i.e. the empirical risk can become zero) then there exists a finite step procedure that allows us to find such a vector w_0, we use the perceptron learning method to train the data set.

In the nonseparable case, we can not apply regular gradient based on procedures to find a local minimum of functional (4.7), since for this functional the gradient is either equal to zero or undefined. Therefore, the idea was proposed to approximate the indicator functions (4.6) by so-called sigmoid functions

$$\bar{f}(x,w) = S\{(w \cdot x)\}, \qquad (4.8)$$

Where $S(u)$ is a smooth monotonic function such that

$$S(-\infty) = -1,$$
$$S(+\infty) = 1.$$

The idea is to use the sigmoid approximation at the stage of estimating the coefficients, and use the threshold functions for the last neuron at the stage of recognition.

4.5 Efficient Implementation

The implementation will be written in C++, using Microsoft's Visual C++ compiler. The algorithm will be tested on experimental data sets and real data sets.

4.5.1 Convex Hull Algorithm

The procedure Convex Hull takes a set Q of points as input, where $|Q| \geq 3$. It calls the functions Head(Q), which returns the point on head of queue Q without changing Q, and Next-to-Top(Q), which returns the point one entry below the head of the queue Q without changing Q. The queue Q returned by Convex Hull contains, from head to tail, exactly the vertices of $CH(Q)$ in counterclockwise order.

Convcx Hull Algorithm (Q)

1. Let p_0 be the point in Q with the minimum $y-coordinate$, or the leftmost such point in case of a tie.
2. Let $\langle p_1, p_2, \ldots, p_m \rangle$ be the remaining points in Q, calculated the polar angle in counterclockwise order around p_0 by using cross product, and sorted by polar angle in counterclockwise order around p_0. (If more than point has the same angle, remove all but the one that is farthest from p_0)

```
for  $i \leftarrow 2$ to $n$
        angle[$i$] $\leftarrow$ $\mathbf{p_0 \bullet p_1}^{1)}$ / $|p_0p_i| \bullet |p_1p_i|$
        min $\leftarrow$ find the minimal angle
        insert min into the $Q$
```

3. Find the points on the convex hull based on the principle that the angle formed by points Next-to-Top, and p_i makes a nonleft turn should be delete from the queue Q.

```
position ← (pDoc → GetlistHeadPosition());
for $i \leftarrow 3$ to $n$
    top ← (pDoc → GetNext());
    topnext ← (pDoc → GetNext());
    t ← (**pi – topnext)*(top-topnext)**;
    if t < 0
        then delete topnext;
return $Q$.
```

4.5.2 Intersection Algorithm

If a point is inside a convex hull, then each area consisting of the point and each edge of the convex hull is always positive, namely the flag equals to 1, otherwise, there exists any area that is negative, it means that the point is outside the convex hull.

```
flag ← 0
for $i \leftarrow 1$ to $n$
    area ← ($p_1$[x] – $p_0$[x]) $*$ (q[y] - $p_0$[y])
            - (q[x] - $p_0$[x]) $*$ ($p_1$[y] – $p_0$[y])
    if area < 0
        then  $i \leftarrow i+1$
    else
        flag ← 1
return  flag
```

4.5.3 Learning Algorithm

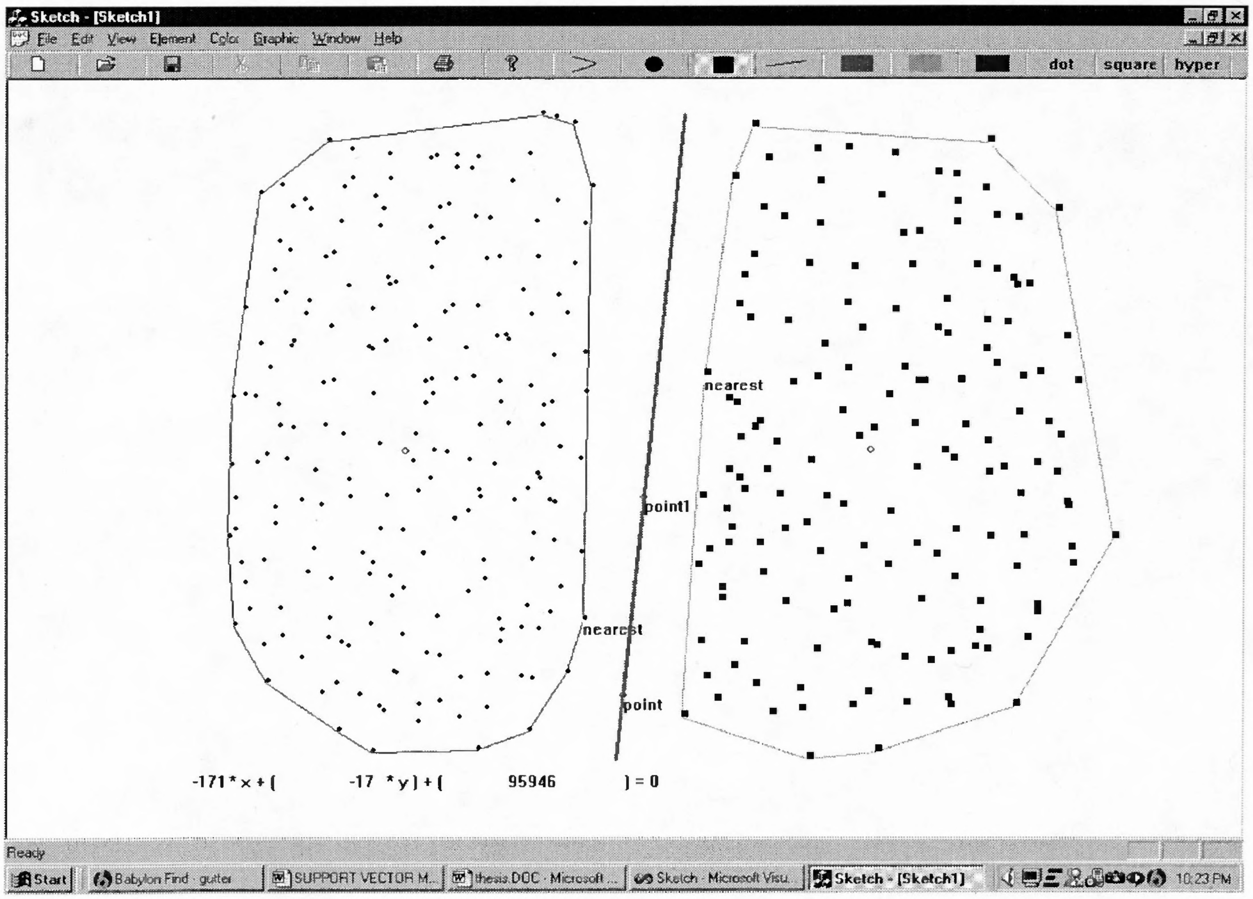
Let us suppose that a perceptron with fixed Φ and adjustable coefficients. When a figure X is presented the sum $\Sigma\alpha_{\varphi}\varphi(X)$ is computed. If X belongs to **F**$^{+}$ and this sum is positive; else if X belongs to **F**$^{-}$ and this sum is negative. Assume that $\Sigma\alpha_\varphi\varphi(X)$ comes out negative for an X in **F**$^+$. In general some φ's give zero values for $\varphi(X)$, and their coefficients clearly cannot be blamed for the bad result. In fact, changing these coefficients might do harm in relation to other X's and dose no good in relation to the current X. Thus we should increase α_φ only if $\varphi(X) = 1$.

```
$\mathbf{w} \rightarrow 0$
$X \in \mathbf{F}^+ \cup \mathbf{F}^-$
for $i \leftarrow 0$ to $n$
    if $X \in \mathbf{F}^+$
      if $\mathbf{w} \bullet \Phi(X) \leq 0$
          then $\mathbf{w} \leftarrow \mathbf{w} + \Phi(X)$
    else
      if $\mathbf{w} \bullet \Phi(X) \geq 0$
          then $\mathbf{w} \leftarrow \mathbf{w} - \Phi(X)$
```

Chapter5. Expected Result

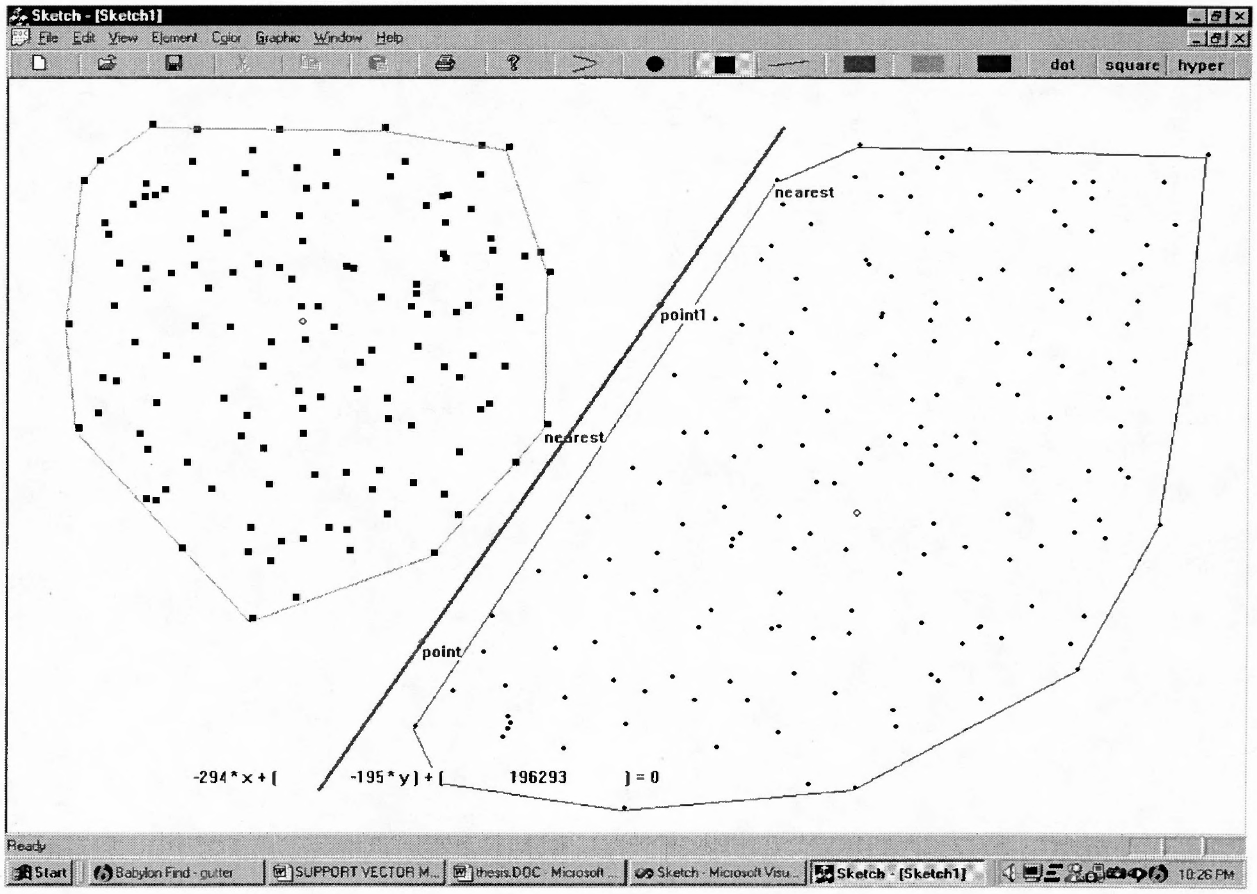
This thesis should contribute to the following problems:

- Processing the given data set to get the support vectors of the data set. Those support vectors determine the optimal hyperplane, perform the training of support vector machines. Figure 14a-c illustrates three training procedure and classification results.



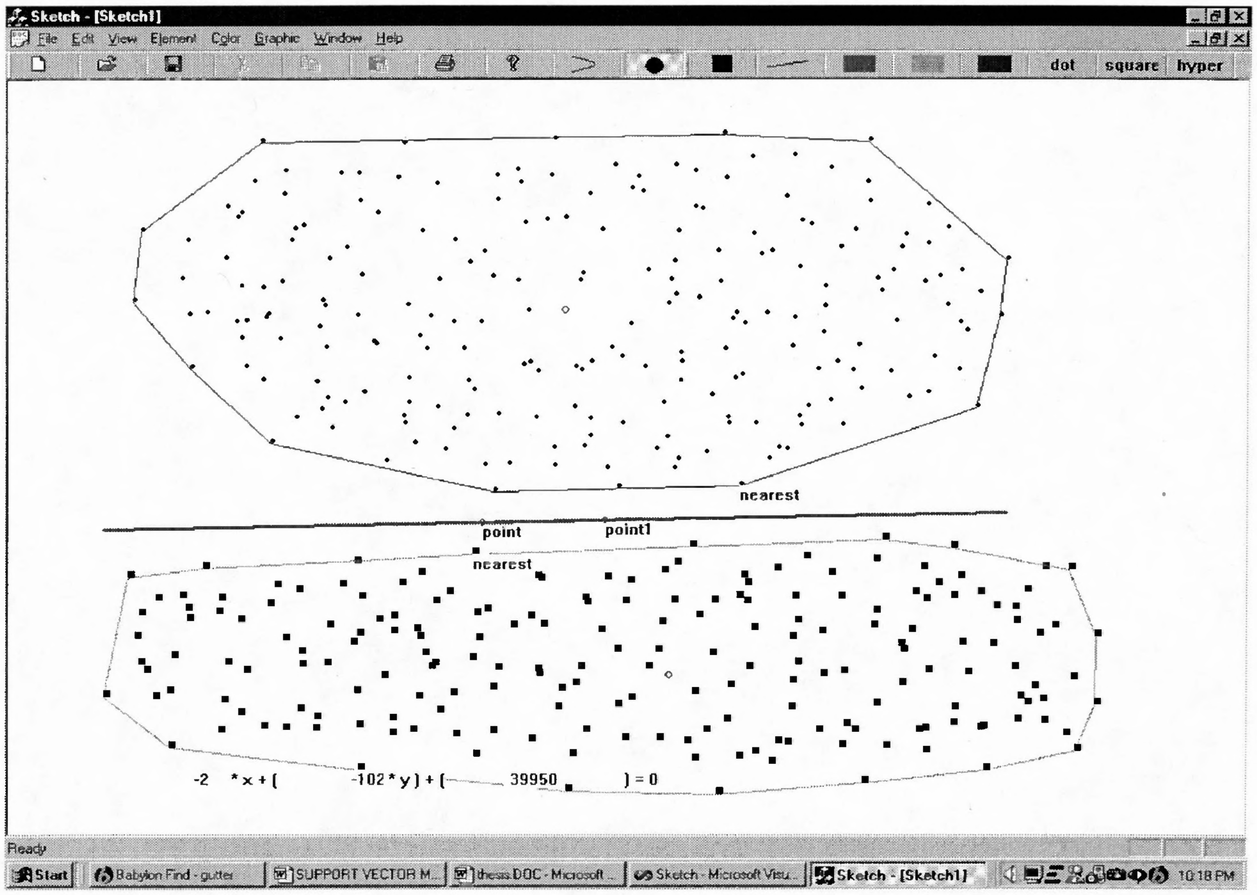
```
-171*x+(     -17  *y)+(     95946      )=0
```

Figure 14a. Training and classifying process.



-294 * x + (-195 * y) + (196293) = 0

Figure 14b. Training and classifying process.



-2 *x+(-102*y)+(39950)=0

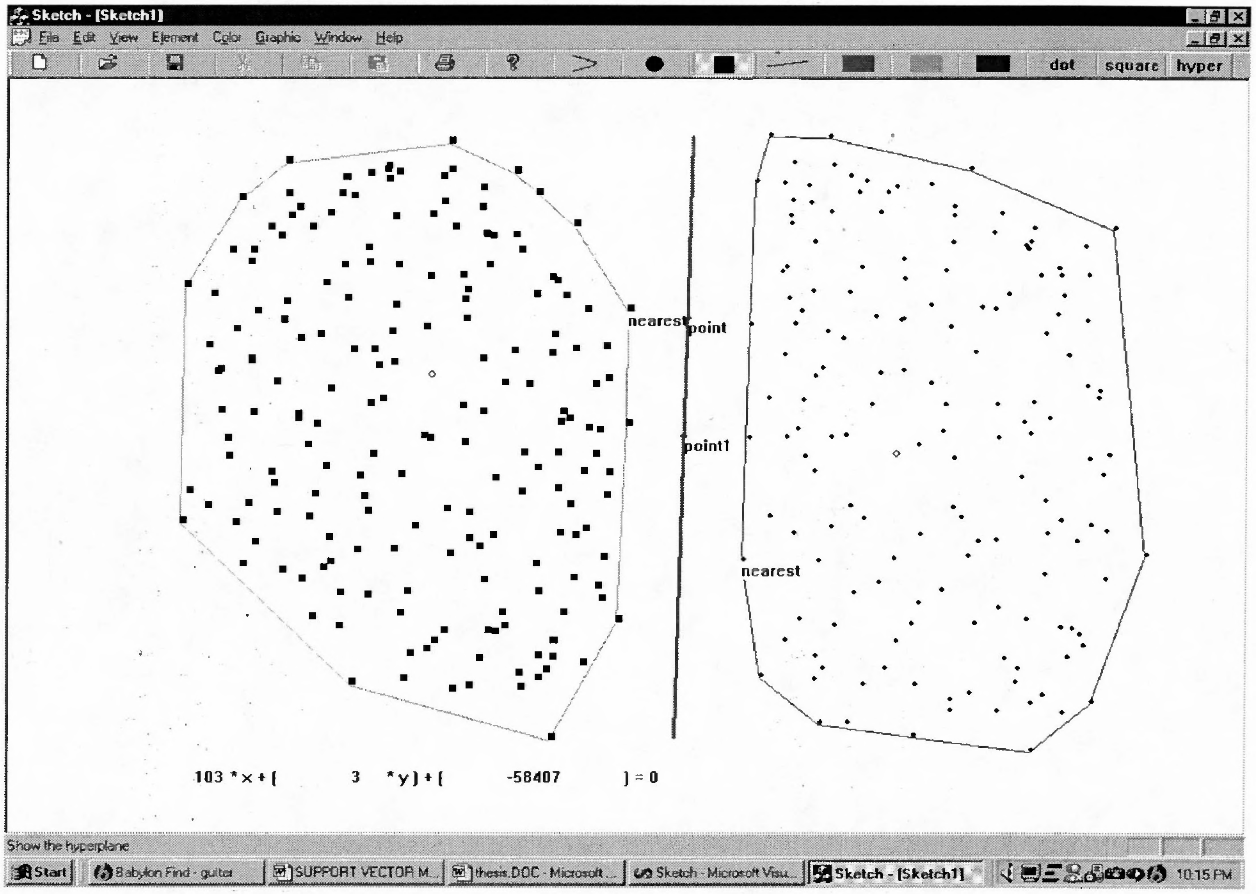
Figure 14c. Training and classifying process.

- Classifying the data in high-dimension.

  Separating the given high-dimension data sets into two sets.

- Interpolating data into the classified data set.

  When new data coming, interpolate it into the correct set according to the hard classifier or soft classifier to. This is a process of regression. The basic idea is to map the data X into a high- dimensional feature space *F* via a nonlinear mapping, and to do linear regression in this space. As shown in figure 15a-c.



```
103 *x+(      3    *y)+(      -58407        )=0
```

Show the hyperplane

Figure 15a. When a new data comes in, interpolate it into right class, if the data is a support vector, then the hyperplan also is modified at same time.

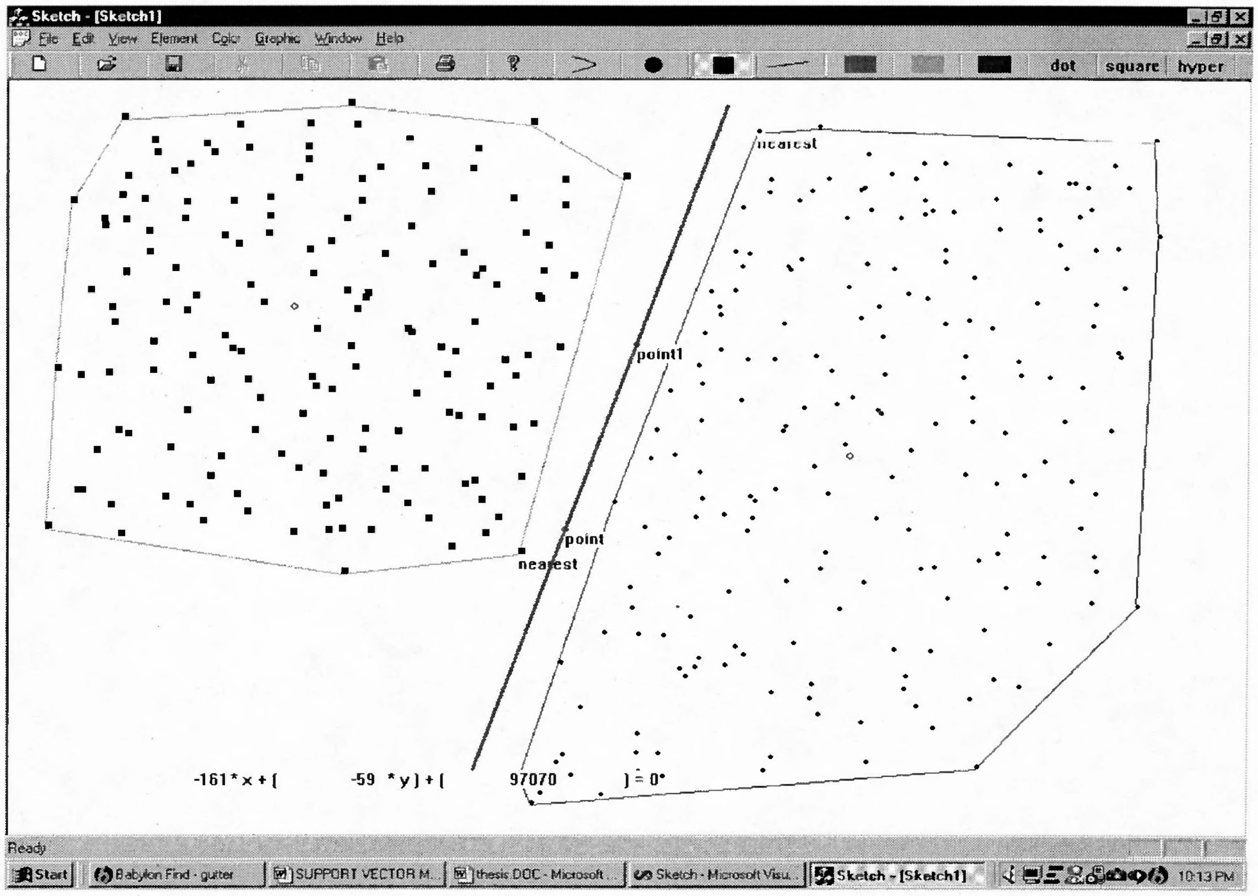
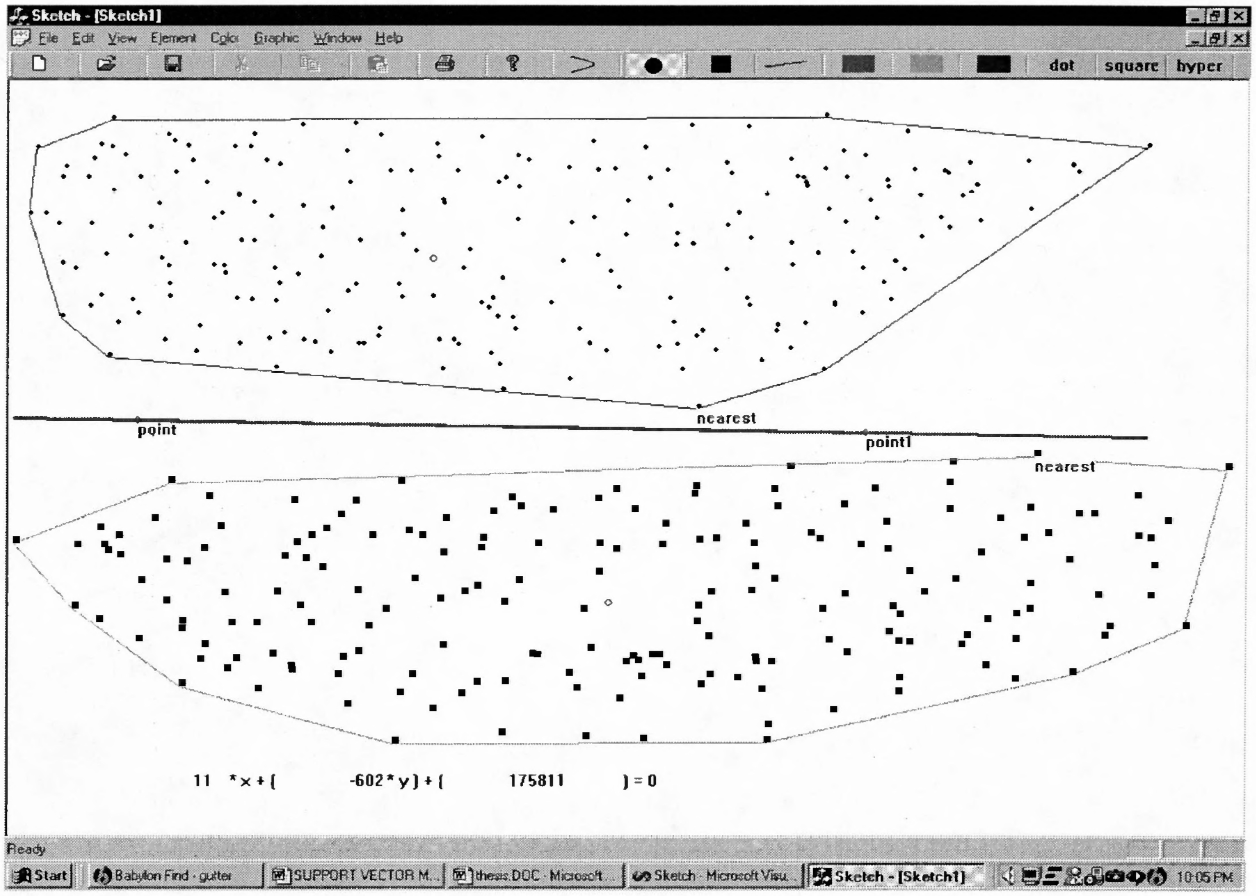

Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
nearest
point1
point
nearest
-161 * x + (-59 * y) + (97070) = 0
Ready
Start
Babylon Find - gutter
SUPPORT VECTOR M...
thesis.DOC - Microsoft...
Sketch - Microsoft Visu...
Sketch - [Sketch1]
10:13 PM

Figure 15b. When a new data comes in, interpolate it into right class, if the data is a support vector, then the hyperplan also is modified at same time.



11 *x+(-602*y)+(175811)=0

Figure 15c. When a new data comes in, interpolate it into right class, if the data is a support vector, then the hyperplan also is modified at same time.

Chapter 6. Conclusion

We first described the technique of hyperplane that separates the data sets into two different categories through training SVs. There exist several cases in training processing, such as, linear, non-linear, and non-separable cases, actually, non-linear, and non-separable cases are the expansion of linear case by computing dot products in feature space by means of kernel functions in original space.

6.1 New contributions of this thesis

(1). A efficient and fast convex hull algorithm for defining the convex hulls for each data set. Based on the convex hull, we only consider the data that sit on the bound of the convex hull. this algorithm tremendous deduces test time because all the data inside the convex hull can be ignored, particular to the linear case.

(2). We developed the Perceptron algorithm to train non-linear case data, which is kind of time consuming at first, but in the long run, as long as we define the hyperplane, we can use the Support Vector Machine to do the pattern recognition, which only involves a very small part of data sets, namely, Support Vectors.

(3). In this thesis, we successfully implemented the above algorithm using acceptable amounts of computer time and memory in MFC to obtain a visual result.

6.2 Possible future research

Even though the SRM inductive principle dose give us a good performance, but there are still some further researches needed to study, such as, a technique for choosing the suitable kernel function and additional capacity control, and development of kernels with invariance.

Another related research is how to decompose the large scale efficiently, as known, the memory for the large scale data set increases in exponential, we have to break the large set into several small sets by using approximately method, however, the approximation can never be as good as the original, so how to construct a efficient approximately method becomes a challenge.

We believe that SVM is very useful to the problem of classification, and to information retrieval at large. We hope that this thesis will encourage some to explore SVMs for further.

Bibliography

[1]. O.L.Managasarian. Generalized Support Vector Machines. Computer Sciences Department University of Wisconsin. Mathematical Programming Technical Report 98-14, October 1998

[2]. CHRISTOPHER J.C BURGES. A Tutorial on Support Vector Machines for Pattern Recognition.

[3]. Steven Gunn. Support Vector Machines for Classification and Regression. ISIS Technical Report. 14 May 1998.

[4]. Edgar E. Osuna, Robert Freund and Federico Girosi. Support Vector Machines: Training and Application. A.I.Memo No. 1602. C.B.C.L Paper No.144. March, 1997.

[5]. Vladimir Vapnik. Three Remarks on the Support Vector Method of Function Estimation. AT&T Labs Research. http://www.research.att.com/info/vlad.

[6]. Ulrich H. –G. Kerβel. Pairwise Classification and Support Vector Machines. Daimler-Benz AG, Research and Technolgy. 1997.

[7] Mark O. Stitson, Alex Gammerman, Vladimir Vapnik, Volodya Vovk, Chris Watkins, Jason Weston. Support Vector Regression with ANOVA Decomposition Kernels

[8]. K. P. Bennett, J. A. Blue. A Support Vector Machine Approach to Decision trees. Mathematical Sciences Department Rensselaer Polytechinc Institute. National Science Foundation Grant 949427.

[9]. V. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag, New York, 1995.

[10]. http://www.umiacs.umd.edu/users/yab/SVMForPatternRecognition/svm.html

[11]. C.Cortes and V.Vapnik. Support Vector network. Machine Learning, 0:273-297, 1995.

[12]. B.Schőlkopf, C.Burges, and V.Vapnik. Extracting support data for a given task. In U. M. Fayyad and R. Uthurusamy, editors, Processings, First International Conference on Knowledge Discovery & Data Mining. AAAI Press, Menlo Park, CA, 1995.

[13]. B.Schőlkopf, C.Burges, and V.Vapnik. Incorporating invariances in support vector learning machines. In C. von der Malsburg, W. von Seelen, J. C. Vorbrűggen, and B. Sendhoff, editors, Artificial Neural Networks – ICANN'96, pages 47-52, Berlin, 1996. Spring Lecture Notes in Computer Science, Vol. 1112.

[14]. B.Schőlkopf, K. Sung, C.Burges, F. Gurisu, P.Niyogi, T. Poggio, and V.Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. IEEE Trans. Sign. Processing, 45:2758-2765, 1997.

[15]. V. Blanz, B.Schőlkopf, H.Bűlthiff, C.Burges, V.Vapnik, and T.Vetter. Comparison of view-based object recognition algorithms using realistic 3D modes. In C.von der Malsburg, W. von Seelen, J. C. Vorbrűggen, and B. Sendhoff, editors, Artificial Neural Networks – ICANN'96, pages 251-256, Berlin, 1996. Spring Lecture Notes in Computer Science, Vol. 1112.

[16]. M.Schmidt. Identifying speaker with support vector networks. In Interface '96 Proceedings, Sydney, 1996.

[17]. Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In IEEE Conference on Computer Vision and Pattern Recognition, pages 130-136, 1997.

[18]. T.Joachims. Text categorization with support vector machines. Technical report, LS VIII Number 23, University of Dortmund, 1997.

[19]. K. –R. Műller, A. Smola, G. Rätsch, B.Schőlkopf, J. Kohlmorgen, and V.Vapnik. Predicting time series with support vector machines. In Proceedings, International Conference on Artificial Neural Networks, page999. Springer Lecture Notes in Computer Science, 1997.

[20]. Edgar Osuna, Robert Freund, and Federico Girosi. An improved training algorithm for support vector machines. In proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing, Eds. J.Principe, L.Giles, N. Wilson, page 276-285, Amelia Island, FL, 1997.

[21]. H. Drucker, C. J. C. Birges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. Advances in Neural Information Processing Systems, 9:155-161, 1997.

[22]. V.Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. Advances in Neural Information Processing Systems, 9:281-287, 1996.

[23]. J. Weston, A. Gammerman, M. O. Stitson, V. Vapnik, V. Vovk, and C. Watkins. Density estimation using support vector machines. Technical report, Royal Holloway College, Report number CSD-TR-97-23, 1997.

[24]. M. O. Stitson, A. Gammerman, V. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector anova decomposition. Technical report, Royal Holloway College, Report number CSD-TR-97-22, 1997.

[25]. C.J.Burges, P.Knirsch, and R.Haratsch. Surpport Vector web page: http://svm.research.bell-labs.com. Technical report, Lucent Technologies, 1996.

[26]. Edgar Osuna, and Federico Girosi. Reducing the run-time complexity of support vector machines. In International Conference on Pattern Recognition 1998.

[27]. B.E.Boser, I.M.Guyon, and V.Vapnik. A training algorithm for optimal margin classifiers. In Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, 1992. ACM.

[28]. D.C.Montgomery and E.A.Peck. Introduction to Linear Regresson Analysis. John Wiley and Sons, inc., 2nd edition, 1992.

[29]. Jack Sklansky. Gustav N. Wassel. Pattern Classifiers and Trainable Machines. Springer-Verlag New York Heidelberg Berlin. 1998. Page 2-3.

[30] Hugh. G. Campbell. Linear Algebra with Applications. Virginia Polytechnic Institute and State University. 1997. Page 193.

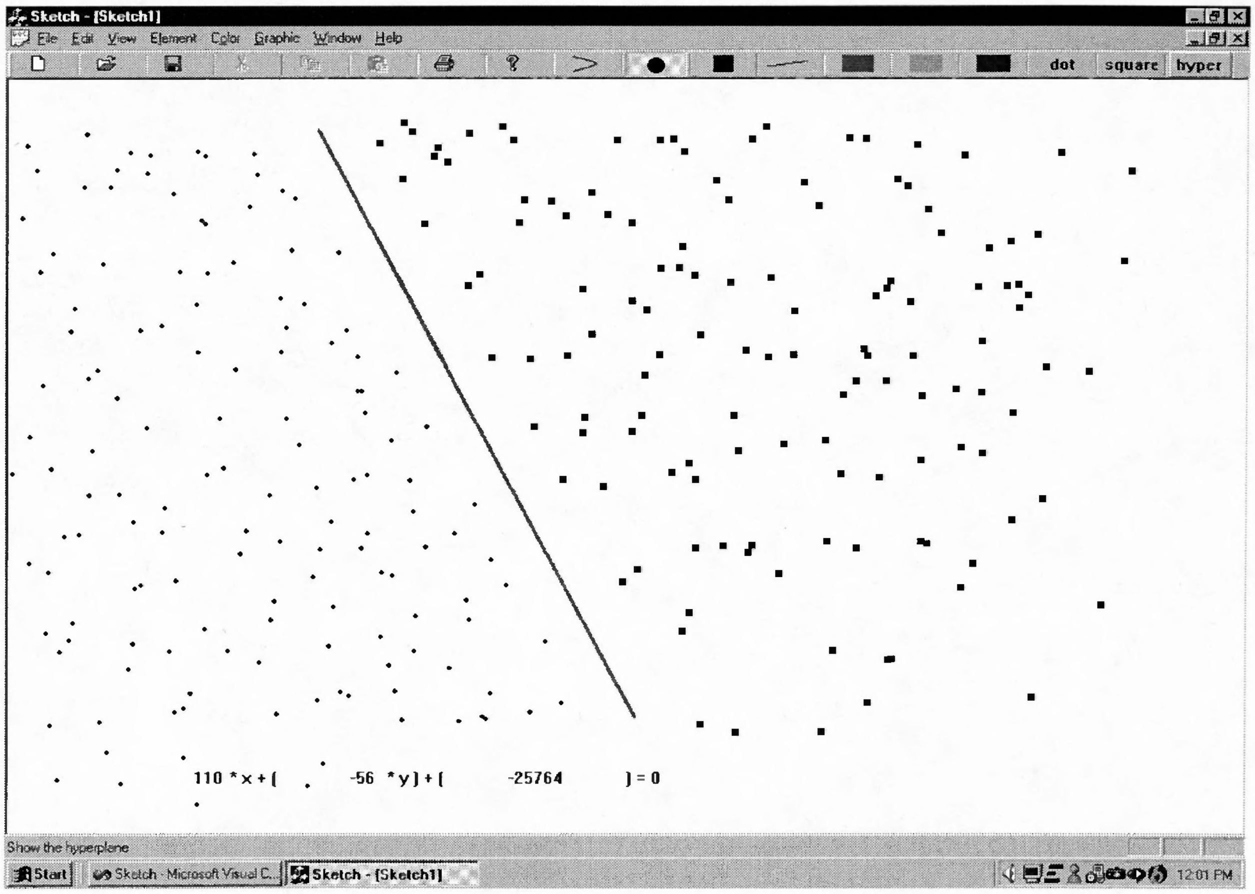
[31]. R. Courant and D. Hilbert. Methods of Mathematical Physics, volume1. Interscience Publishers, Inc, New York, 1953.

[32]. Hans Paul Künzi, Wilhelm Krelle, Werner Oettli. Nonlinear programming. Waltham, Mass., Blaisdell Pub. Co. [1966].

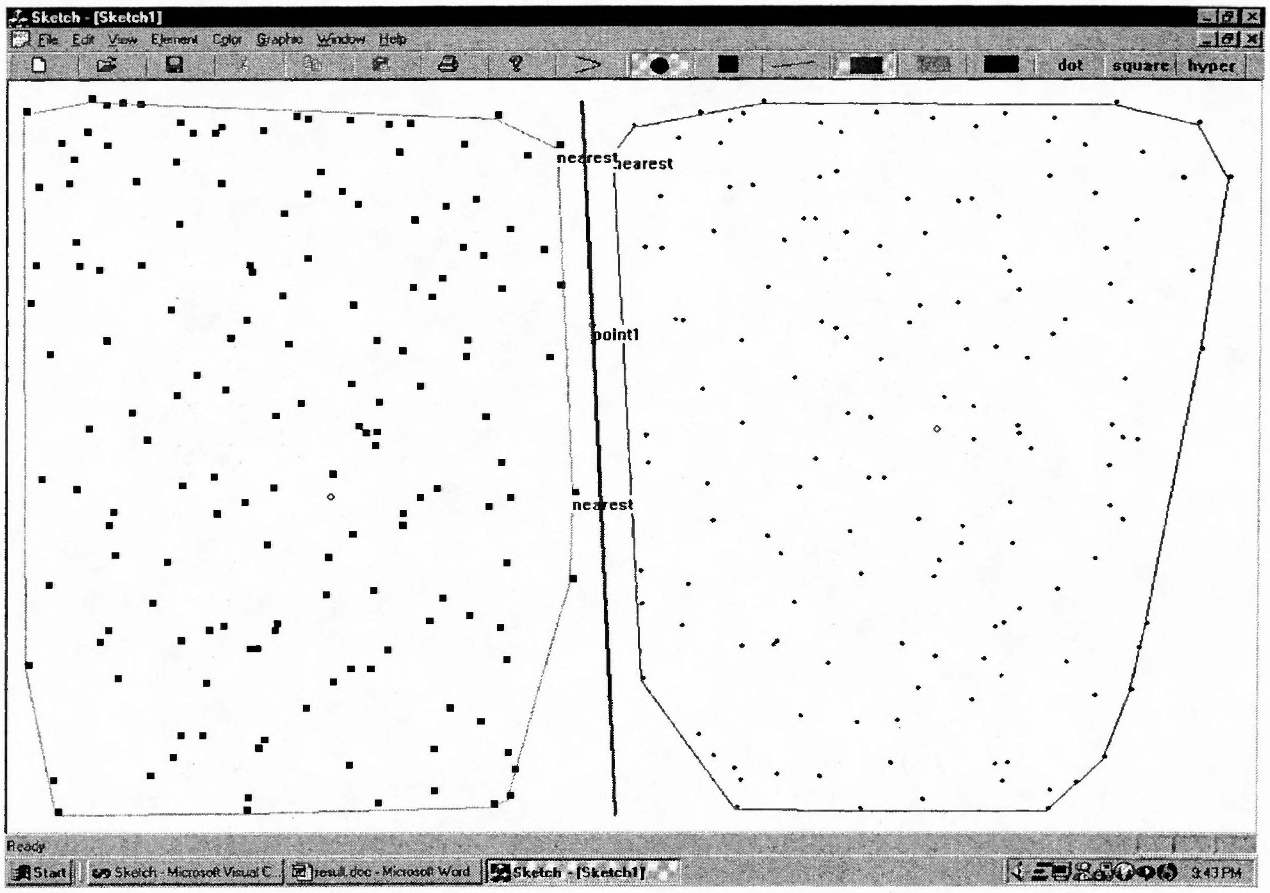
[33]. Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rovest. Introduction to Algorithms. The MIT Press, 1989.

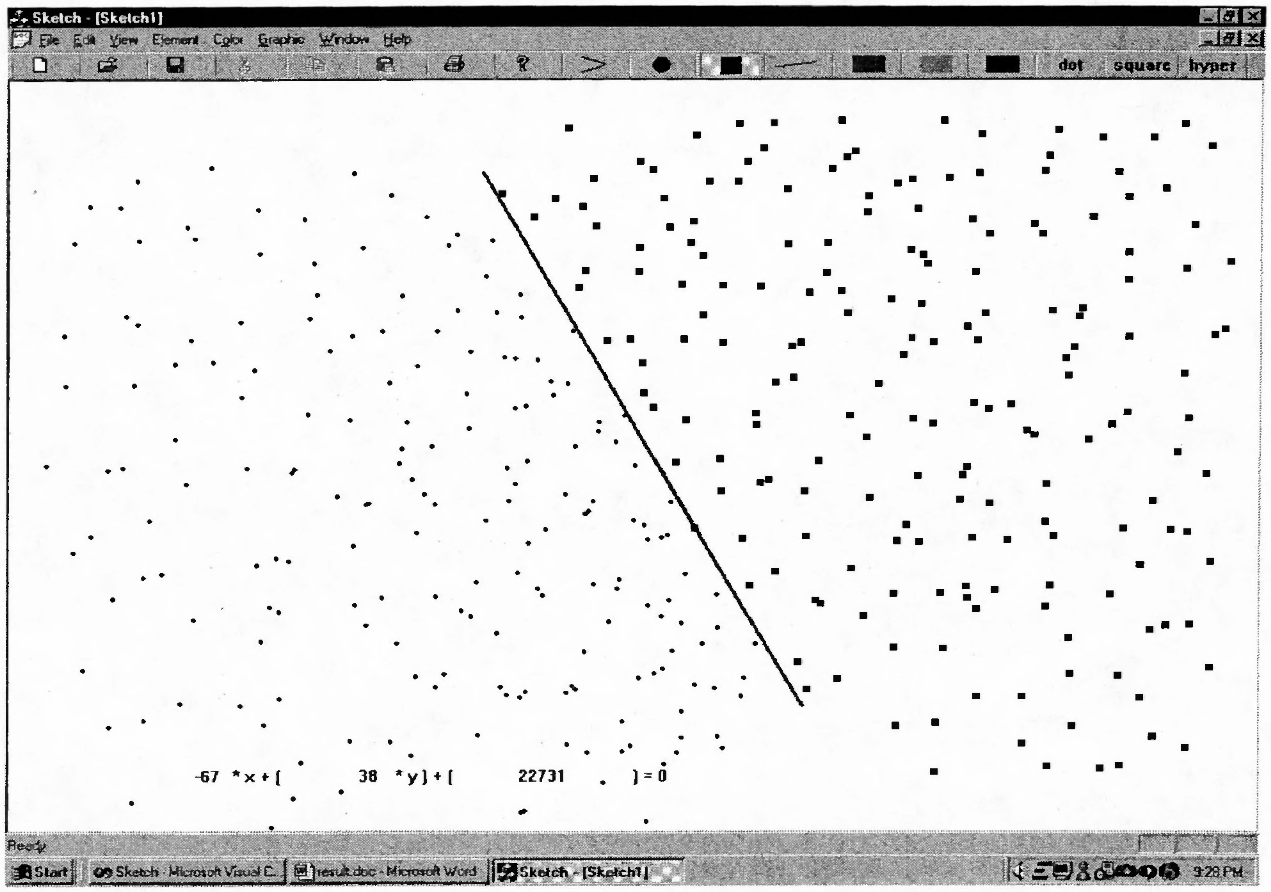
Appendix

In this section, we collected some results, which reference to different scenarios, such as linear cases and non-linear cases. In each case, we also presented the decision function corresponding to the classification.

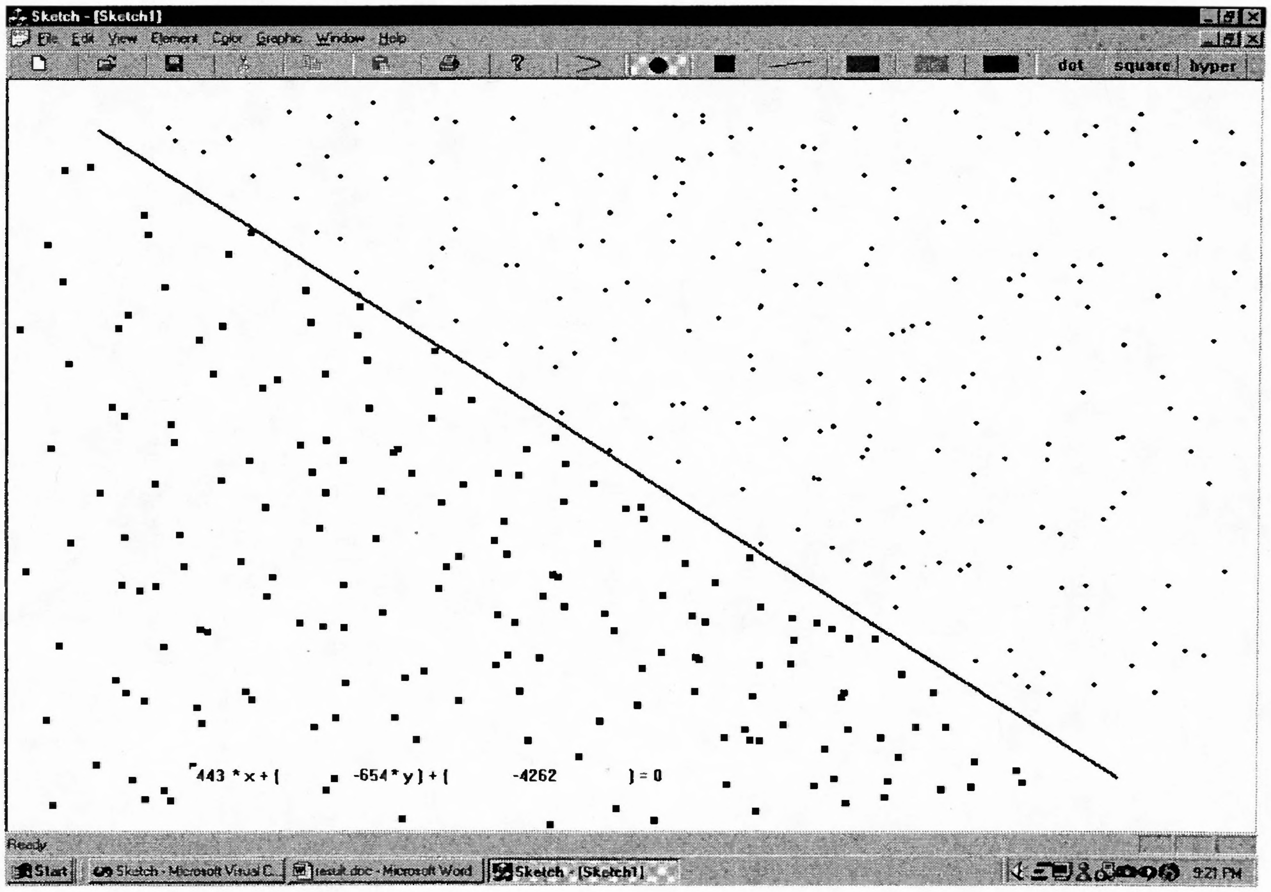


110 *x+(-56 *y)+(-25764)=0

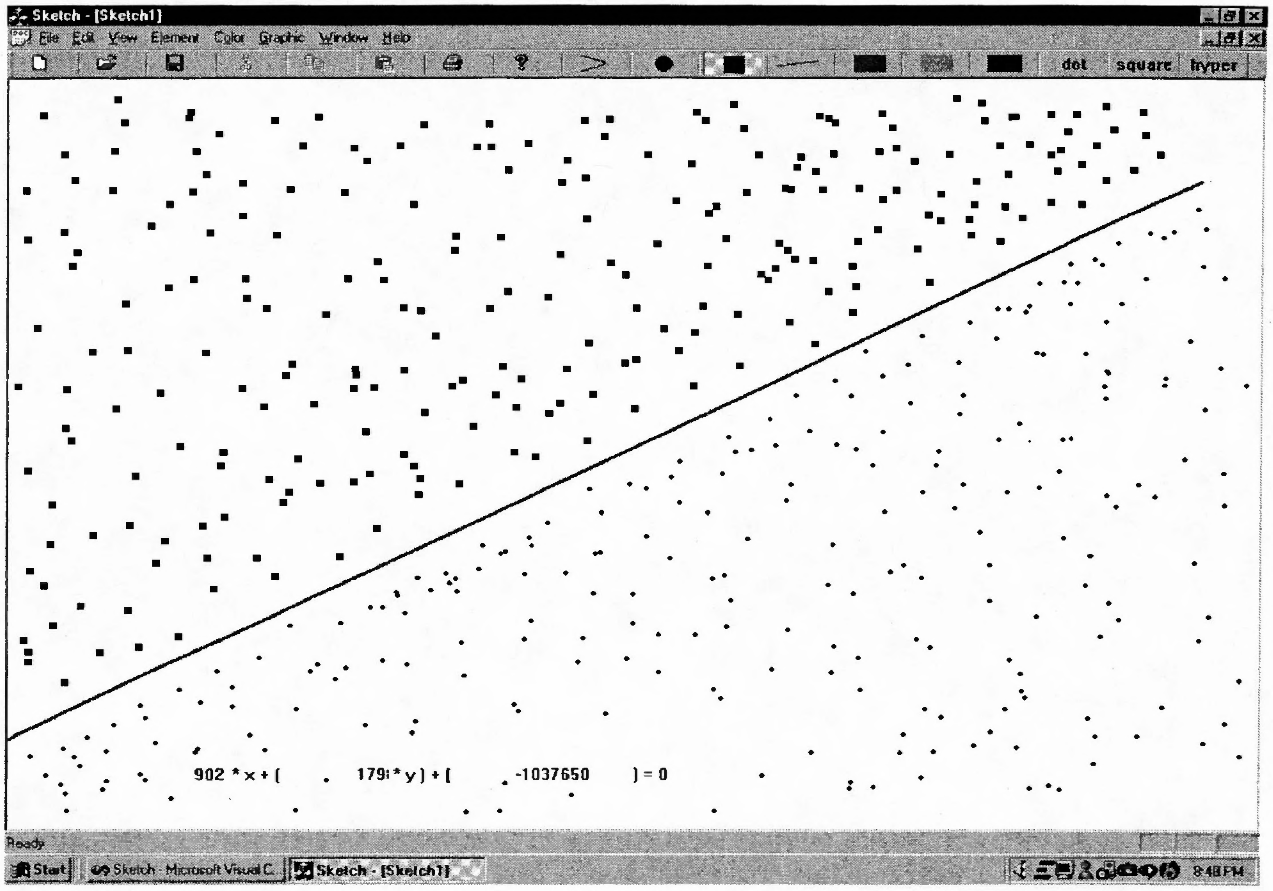

Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot square hyper
nearest
nearest
point1
nearest
Ready
Start
Sketch - Microsoft Visual C
result.doc - Microsoft Word
Sketch - [Sketch1]
3:43 PM


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
9:28 PM

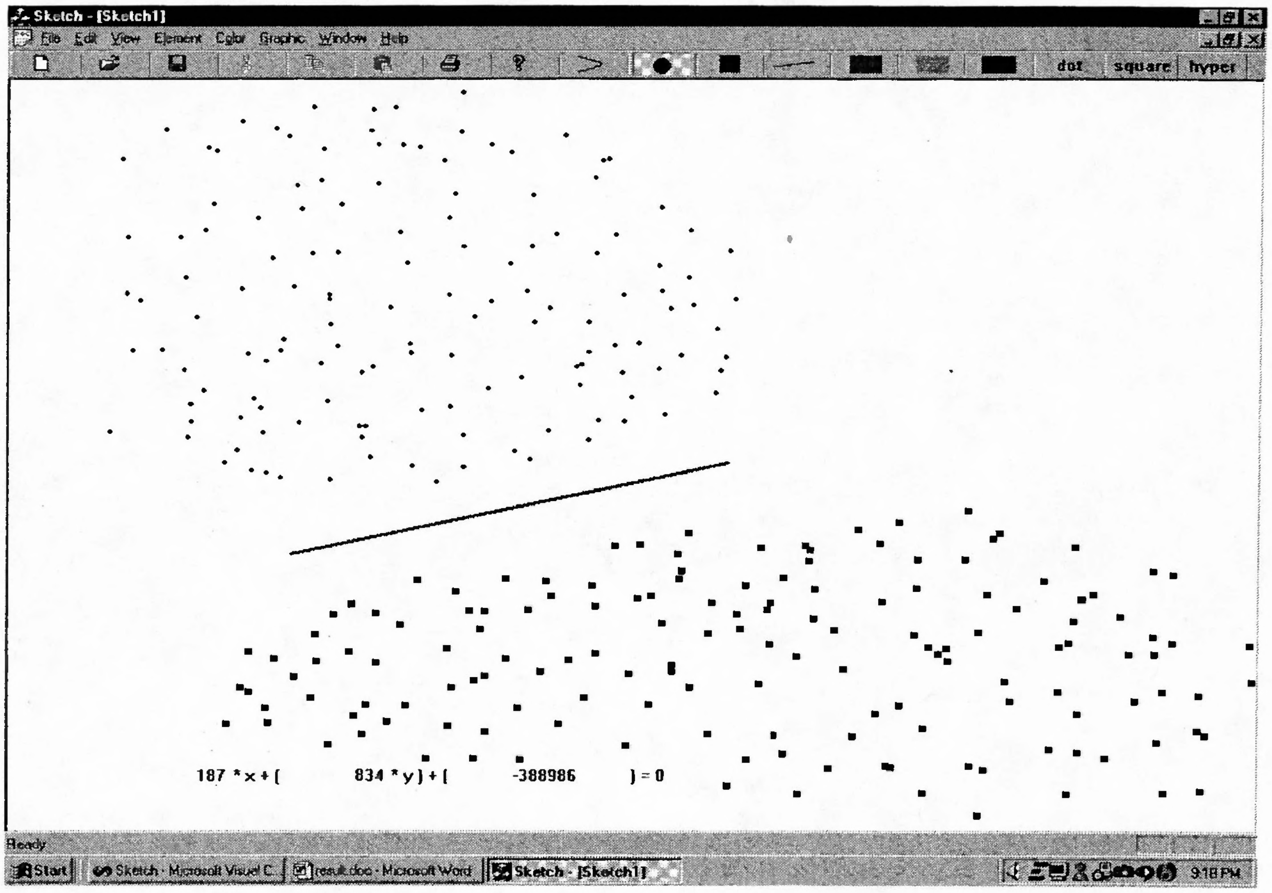
```
-67  *x + (        38  *y) + (        22731        ) = 0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
9:21 PM

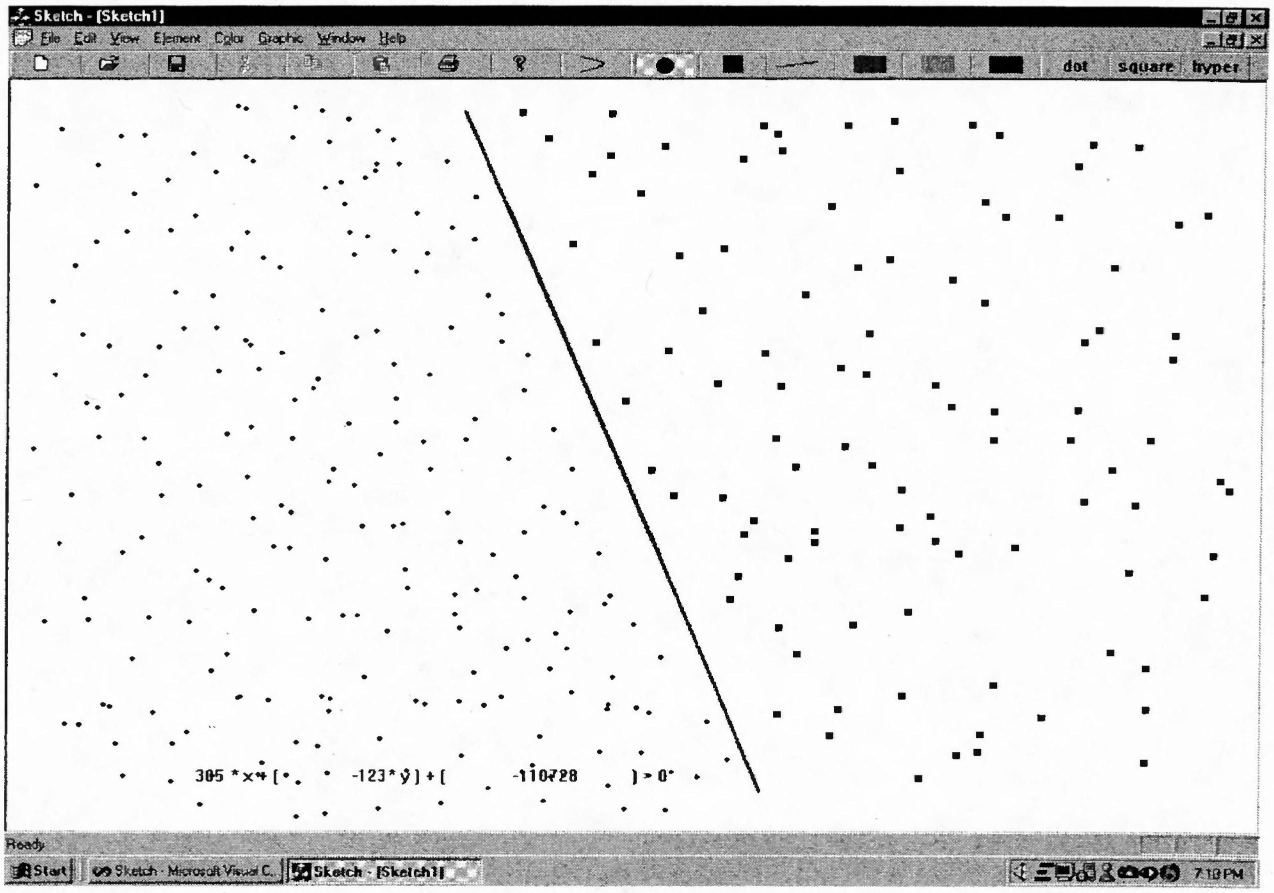
```
443 * x + (        -654 * y ) + (         -4262          ) = 0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot square hyper
Ready
Start
Sketch - Microsoft Visual C...
Sketch - [Sketch1]
8:48 PM

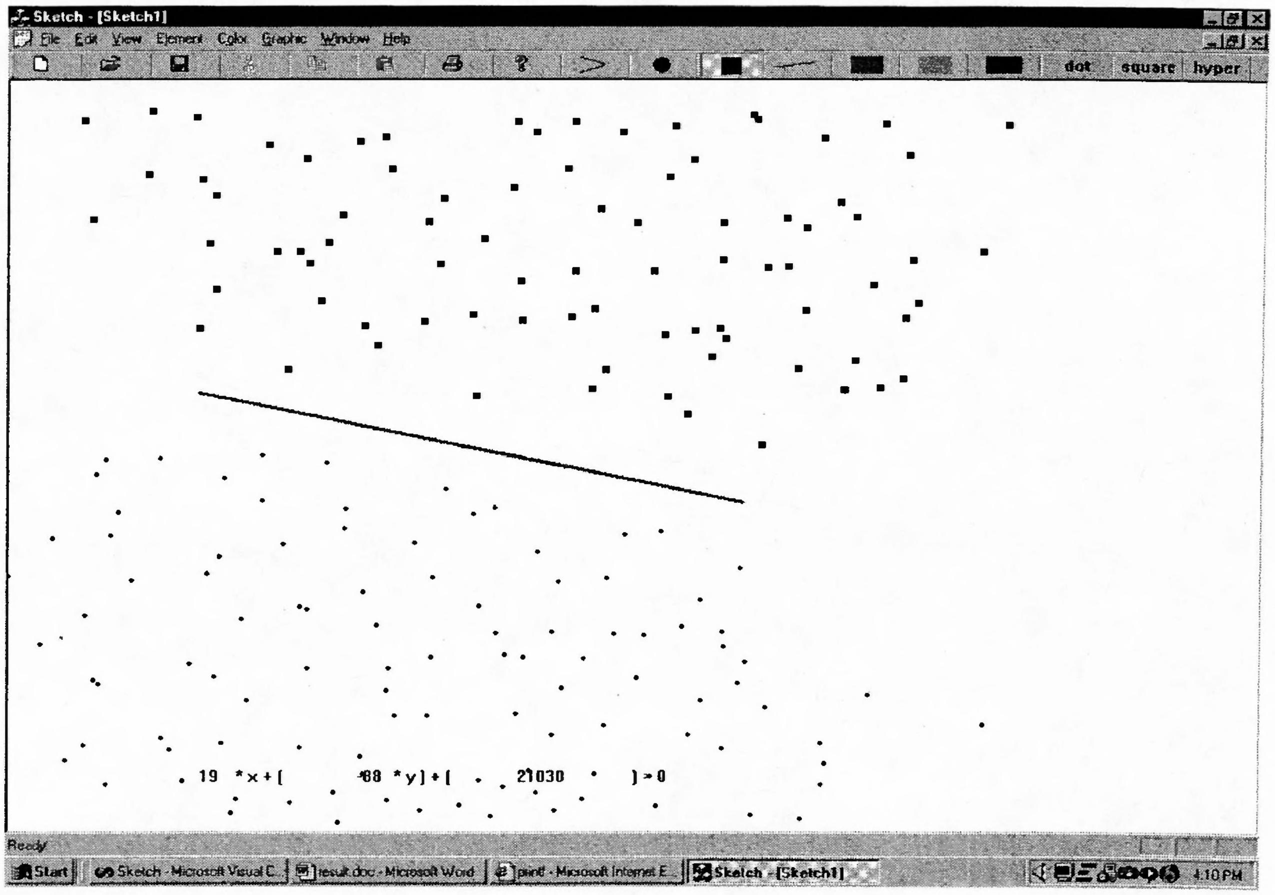
```
902 * x + (        179: * y ) + (        -1037650        ) = 0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
9:18 PM

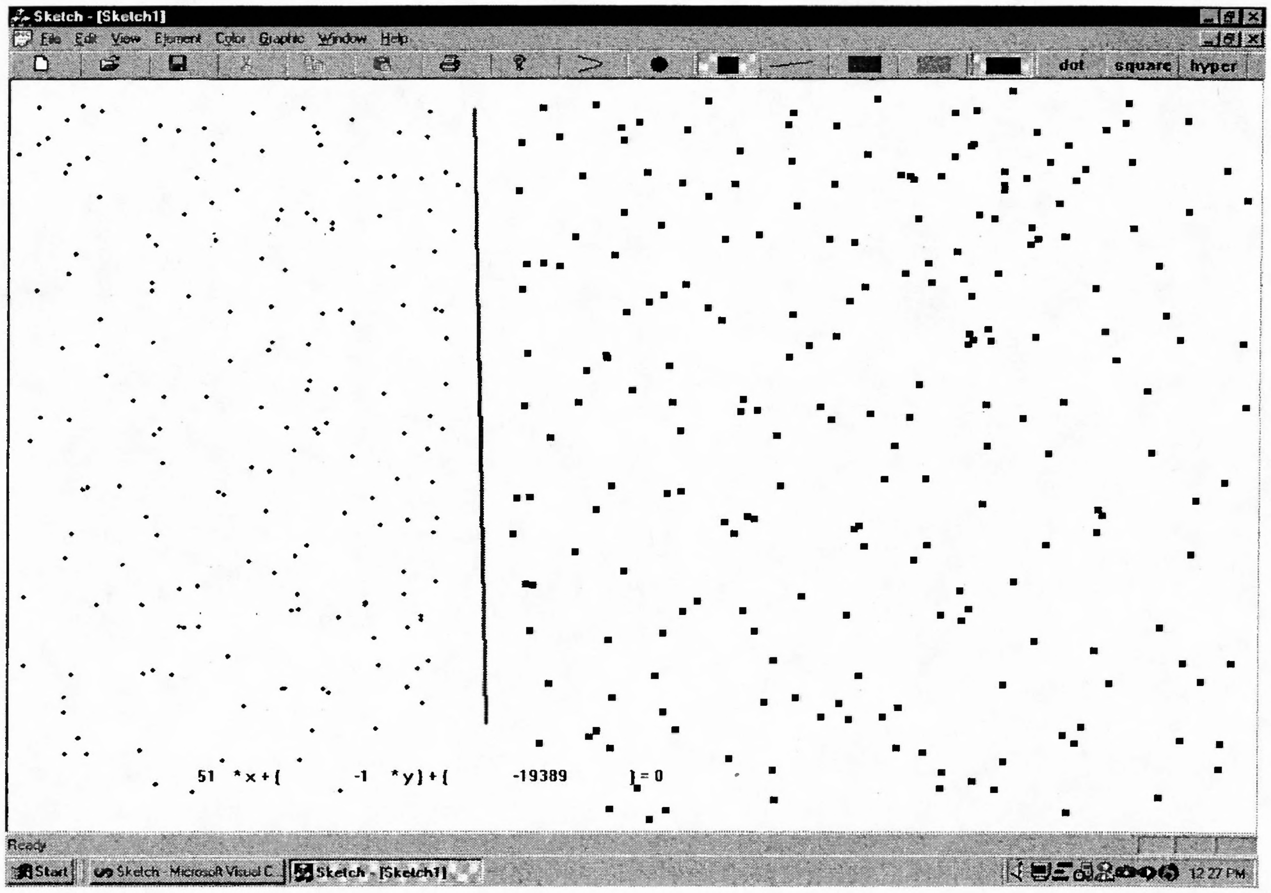
```
187 *x + (          834 * y) + (          -388986          ) = 0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot square hyper
Ready
Start
Sketch - Microsoft Visual C...
Sketch - [Sketch1]
7:10 PM

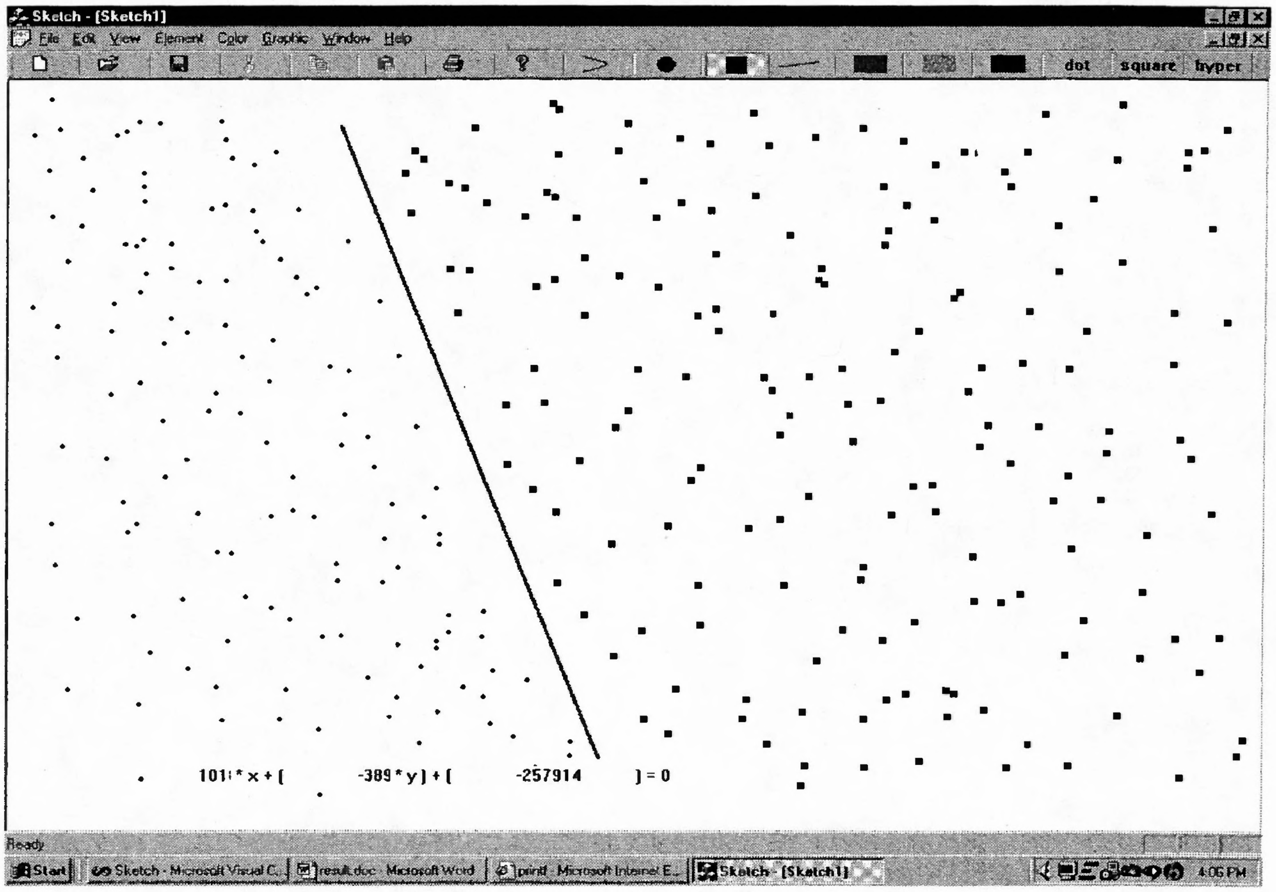
```
305 * x + ( -123 * y ) + ( -110728 ) > 0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot square hyper
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
paint - Microsoft Internet E...
Sketch - [Sketch1]
4:10 PM

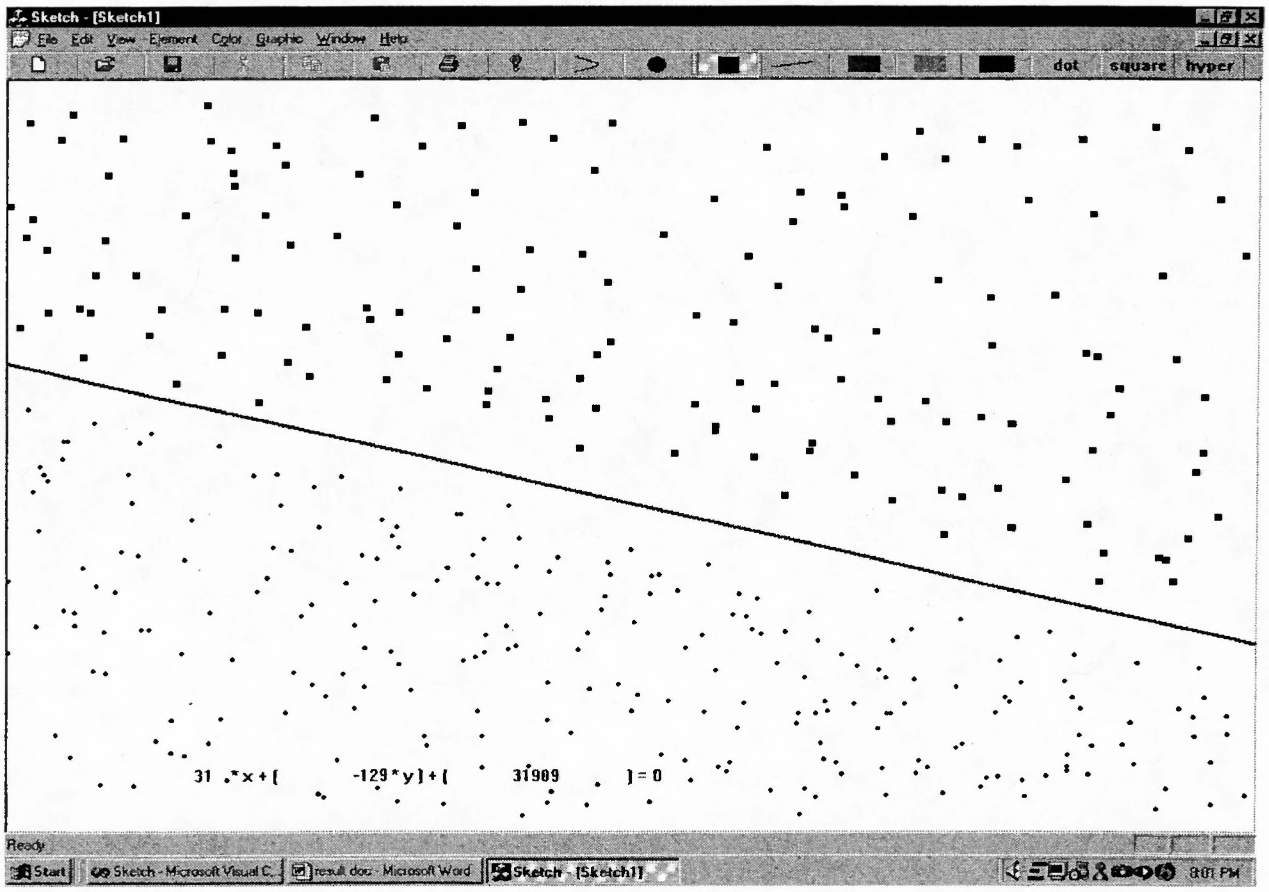
```
19  *x+(        -88 *y)+(      21030       )>0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
Sketch - Microsoft Visual C...
Sketch - [Sketch1]
12:27 PM

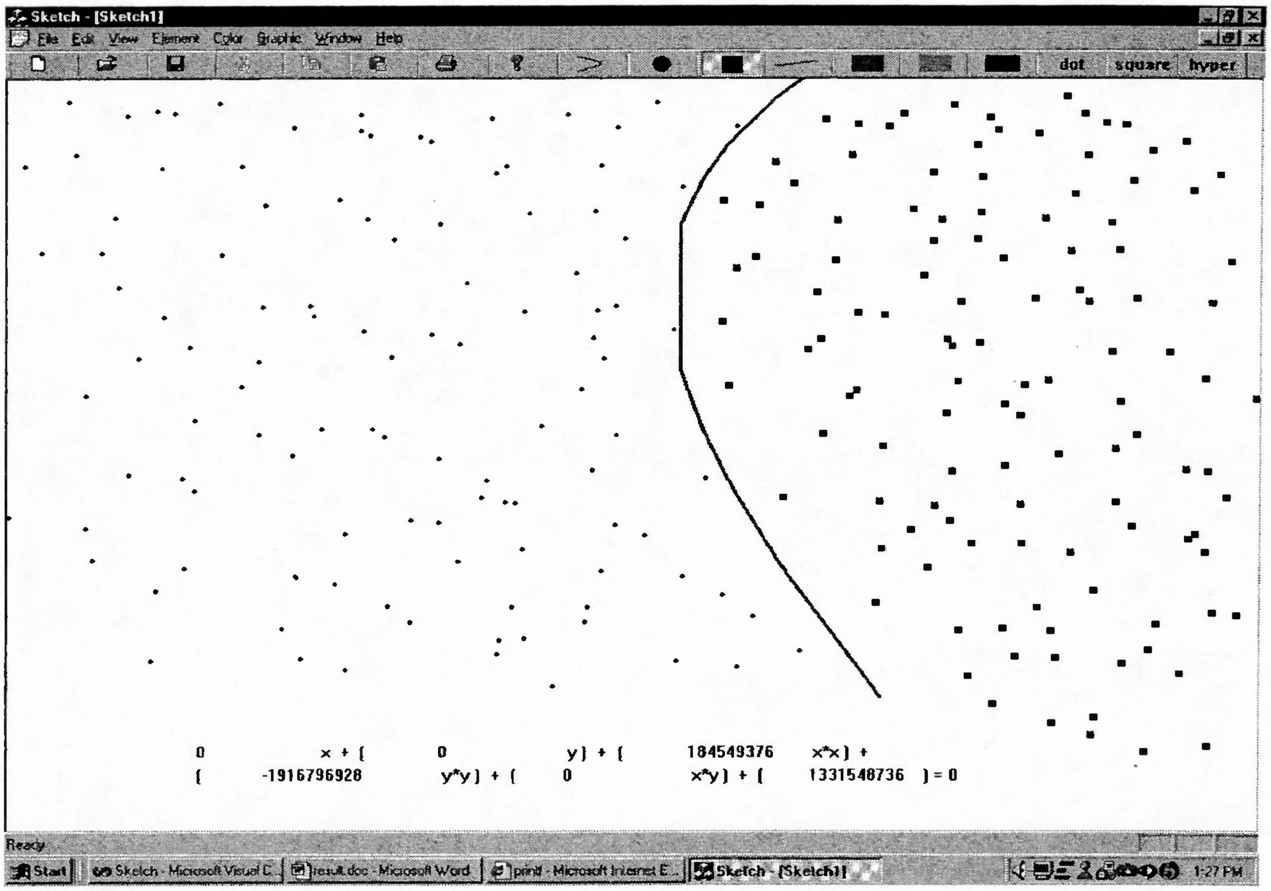
```
51   *x+(        -1   *y)+(         -19389         )=0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
print - Microsoft Internet E...
Sketch - [Sketch1]
4:06 PM

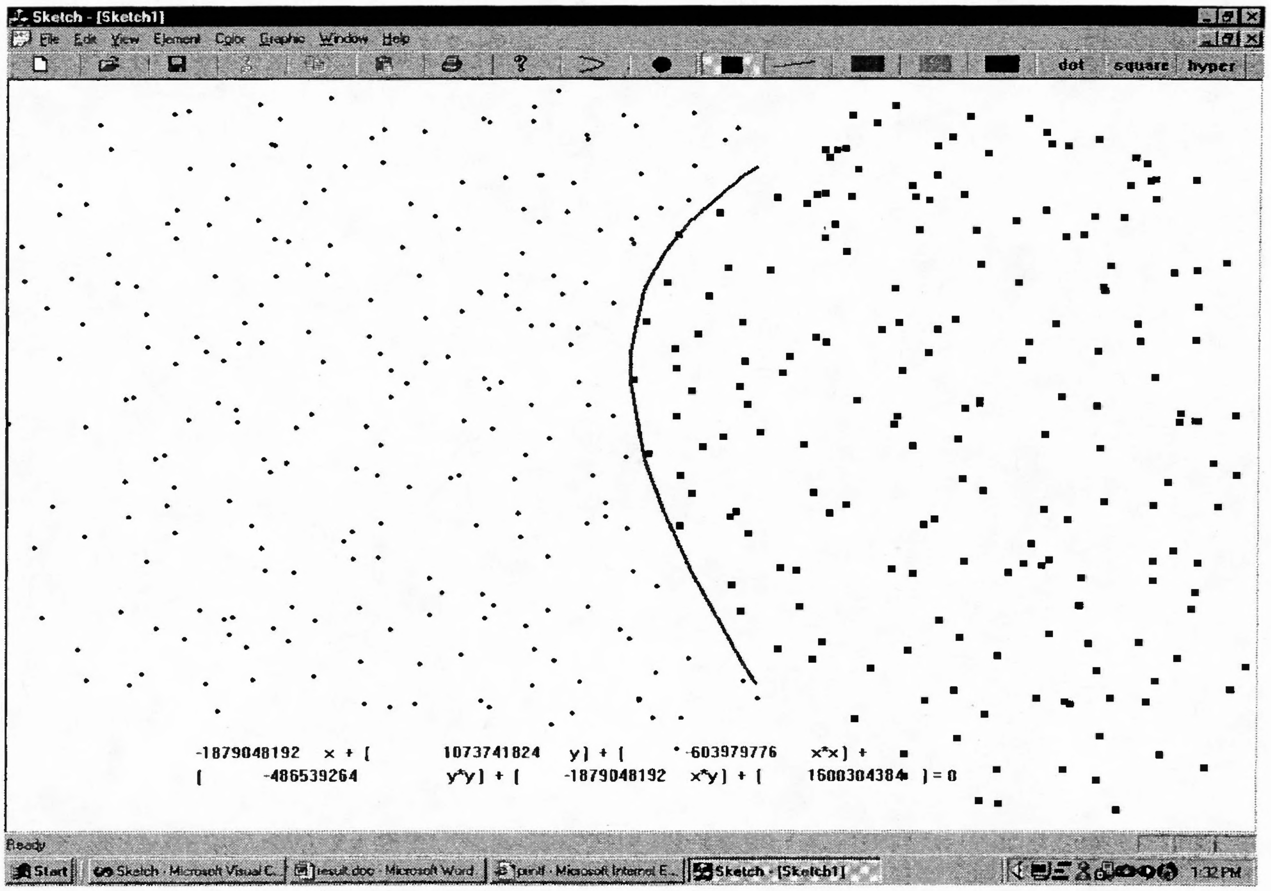
```
1011 * x + (          -389 * y ) + (          -257914          ) = 0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
8:01 PM

```
31 *x + (        -129*y) + (        31909        ) = 0
```

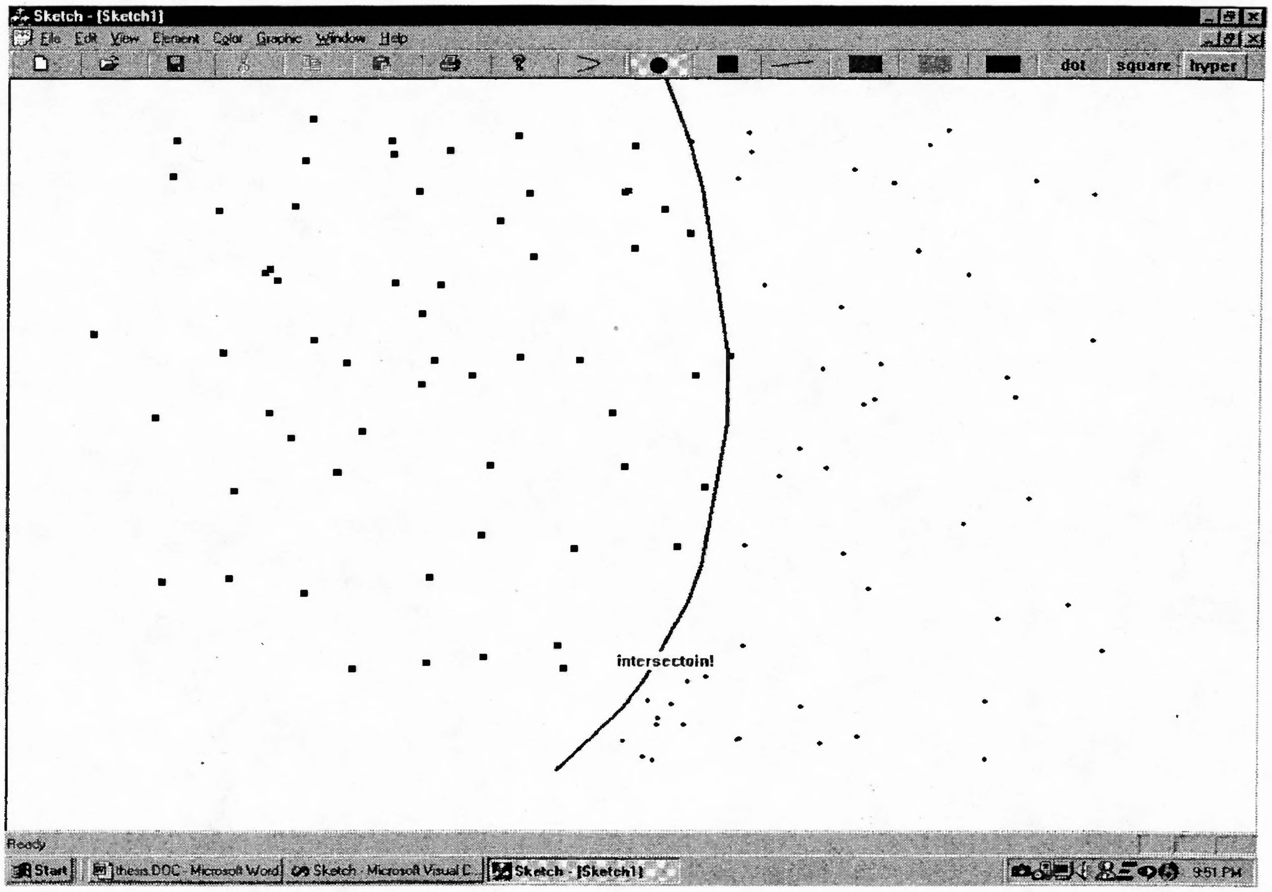

Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
print - Microsoft Internet E...
Sketch - [Sketch1]
1:27 PM

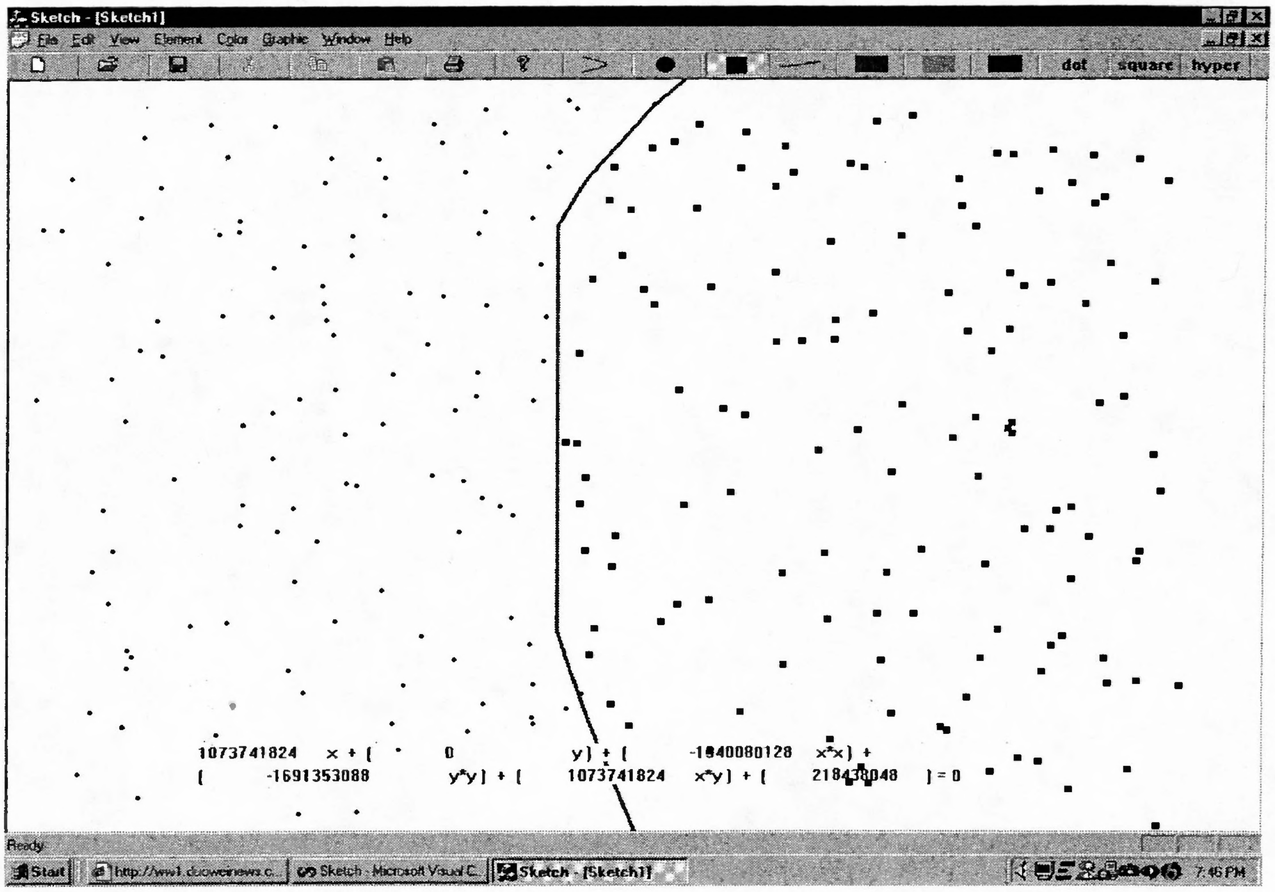
```
0             x + (        0              y) + (        184549376      x*x) +
(        -1916796928          y*y) + (      0               x*y) + (       1331548736   ) = 0
```



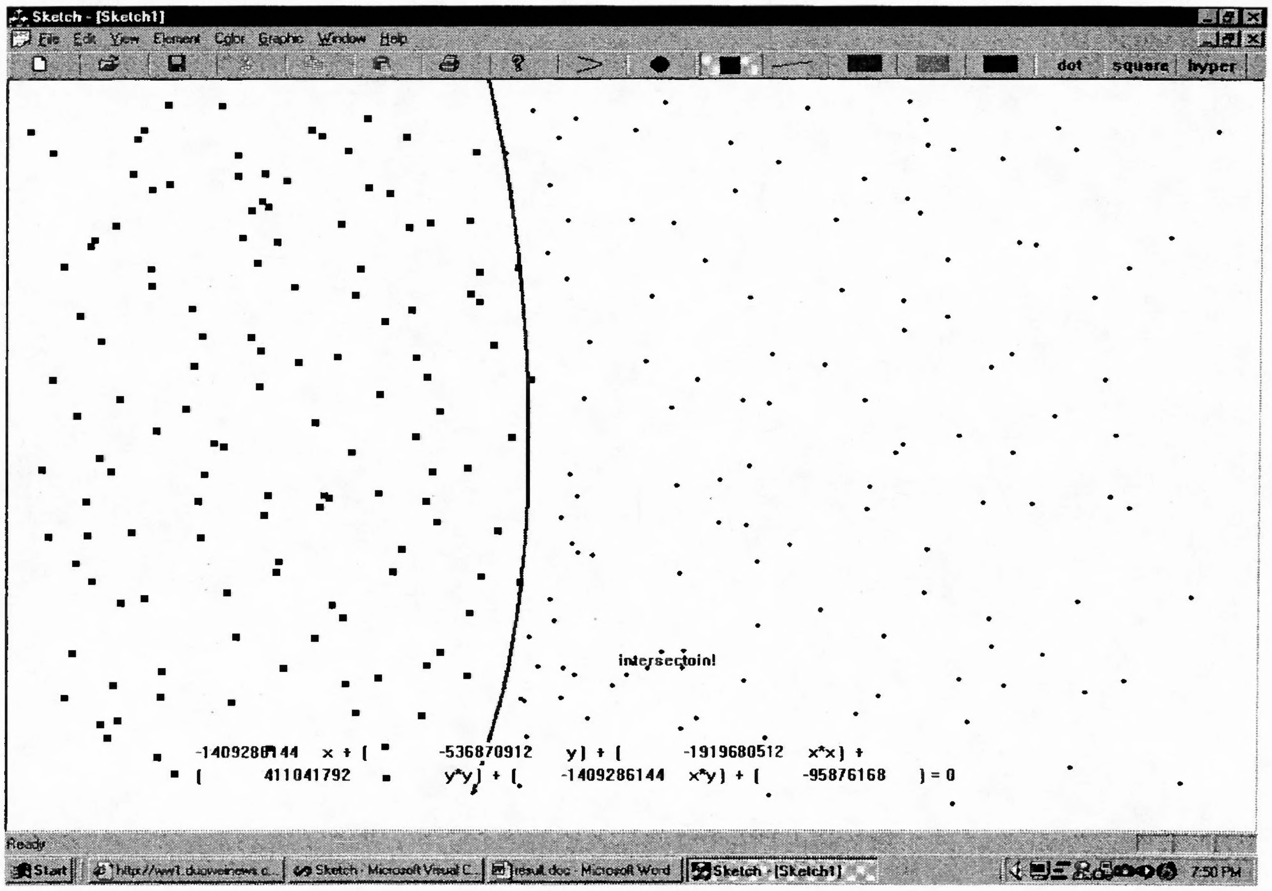
Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot square hyper
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
pinif - Microsoft Internet E...
Sketch - [Sketch1]
1:32 PM

```
-1879048192   x + [          1073741824    y] + [        -603979776     x*x] +
[          -486539264              y*y] + [      -1879048192   x*y] + [       1600304384  ] = 0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
intersectoin!
Ready
Start
thesis.DOC - Microsoft Word
Sketch - Microsoft Visual C...
Sketch - [Sketch1]
9:51 PM

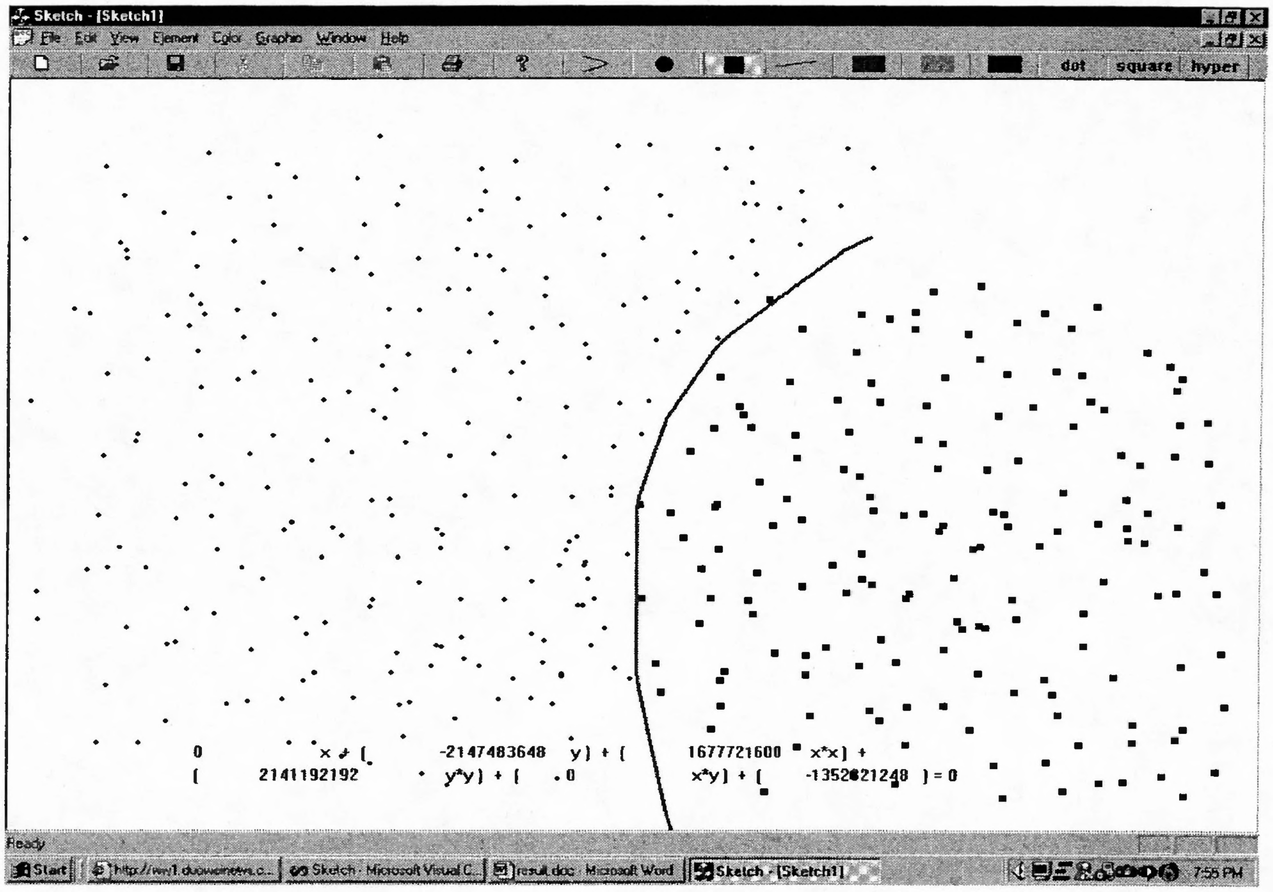

Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
http://ww1.duoweinews.c...
Sketch - Microsoft Visual C...
Sketch - [Sketch1]
7:46 PM

```
1073741824    x + (         0              y) + (        -1940080128   x*x) +
(        -1691353088          y*y) + (     1073741824    x*y) + (       218438048    ) = 0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
http://ww1.duoweinews.c...
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
7:50 PM

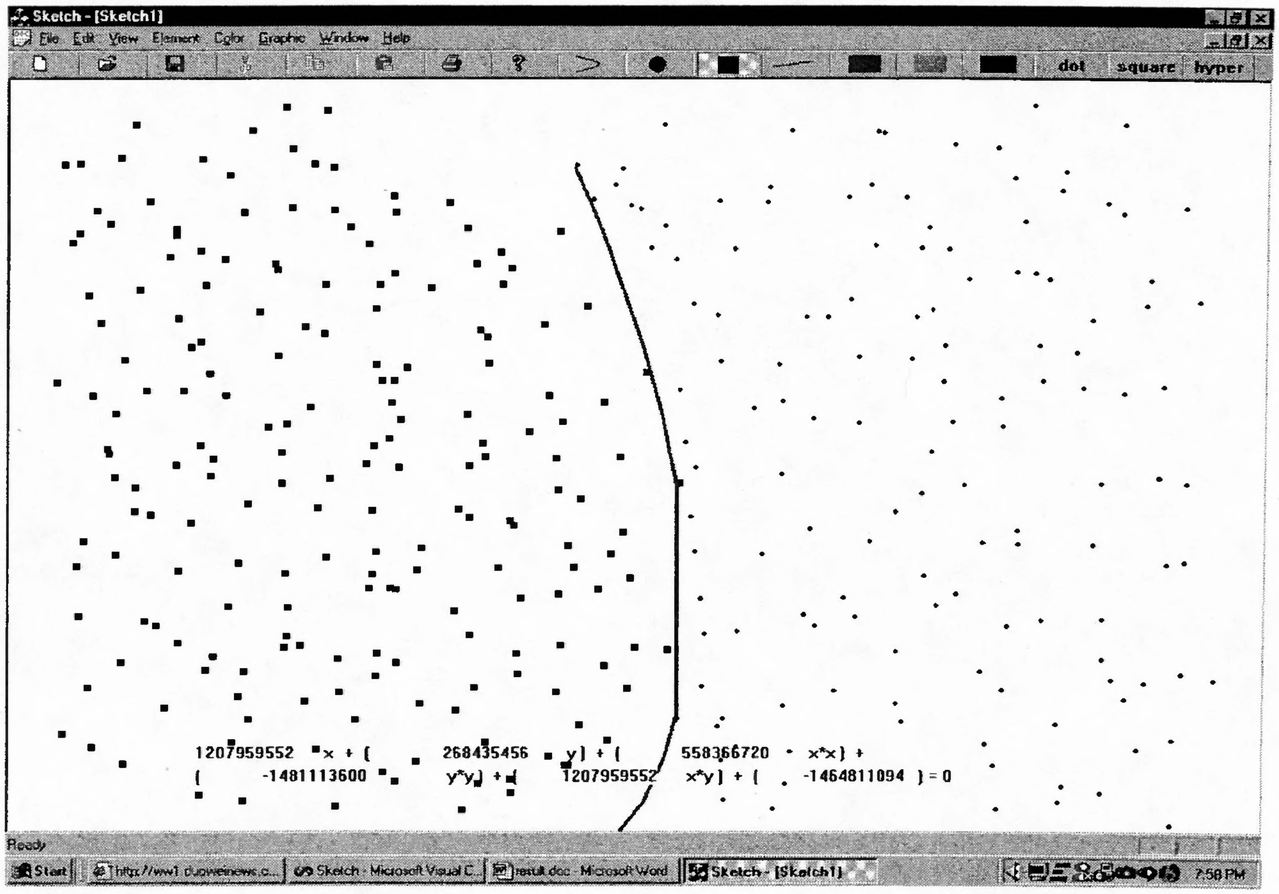
intersectoin!

```
-1409286144   x + (        -536870912   y) + (        -1919680512   x*x) +
(        411041792           y*y) + (     -1409286144   x*y) + (      -95876168    ) = 0
```

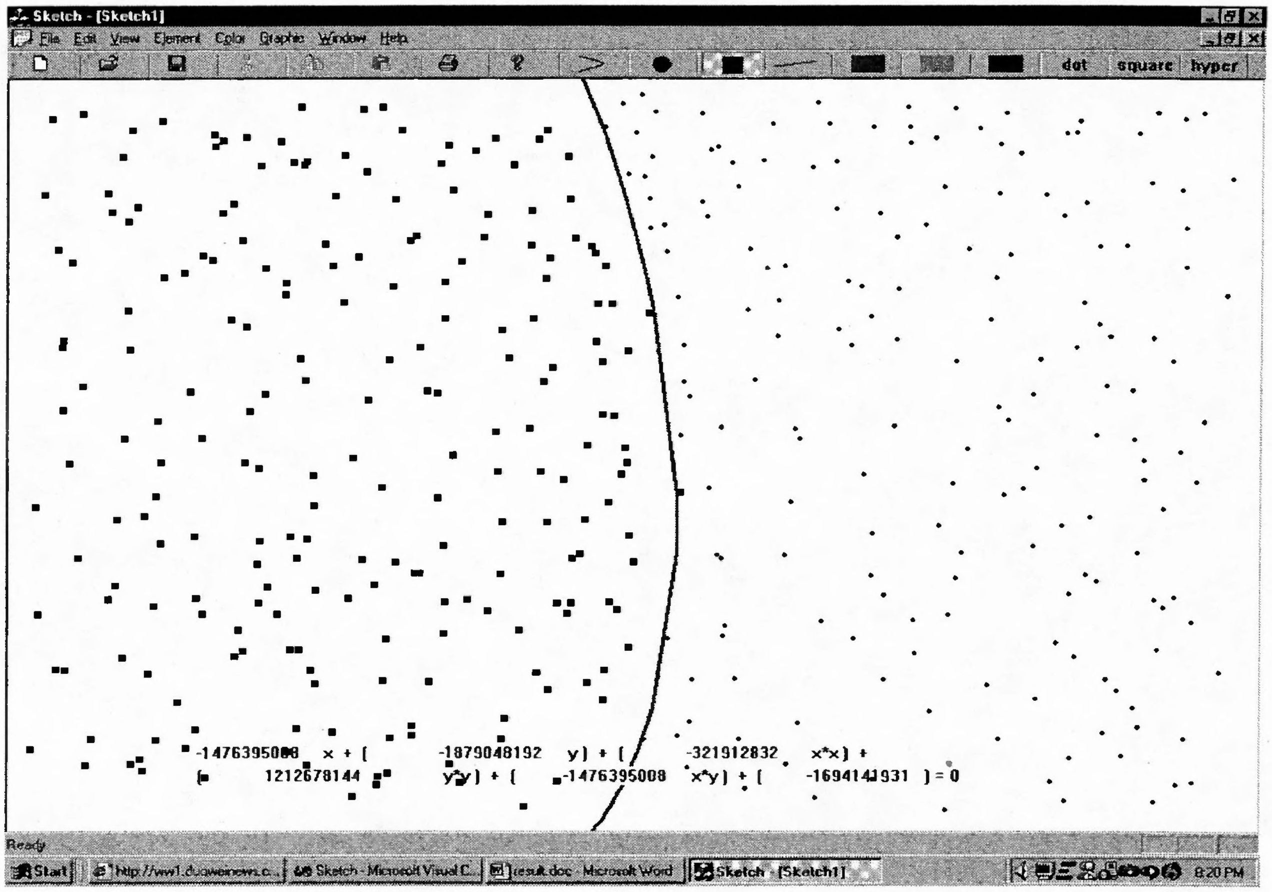


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot square hyper
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
7:55 PM

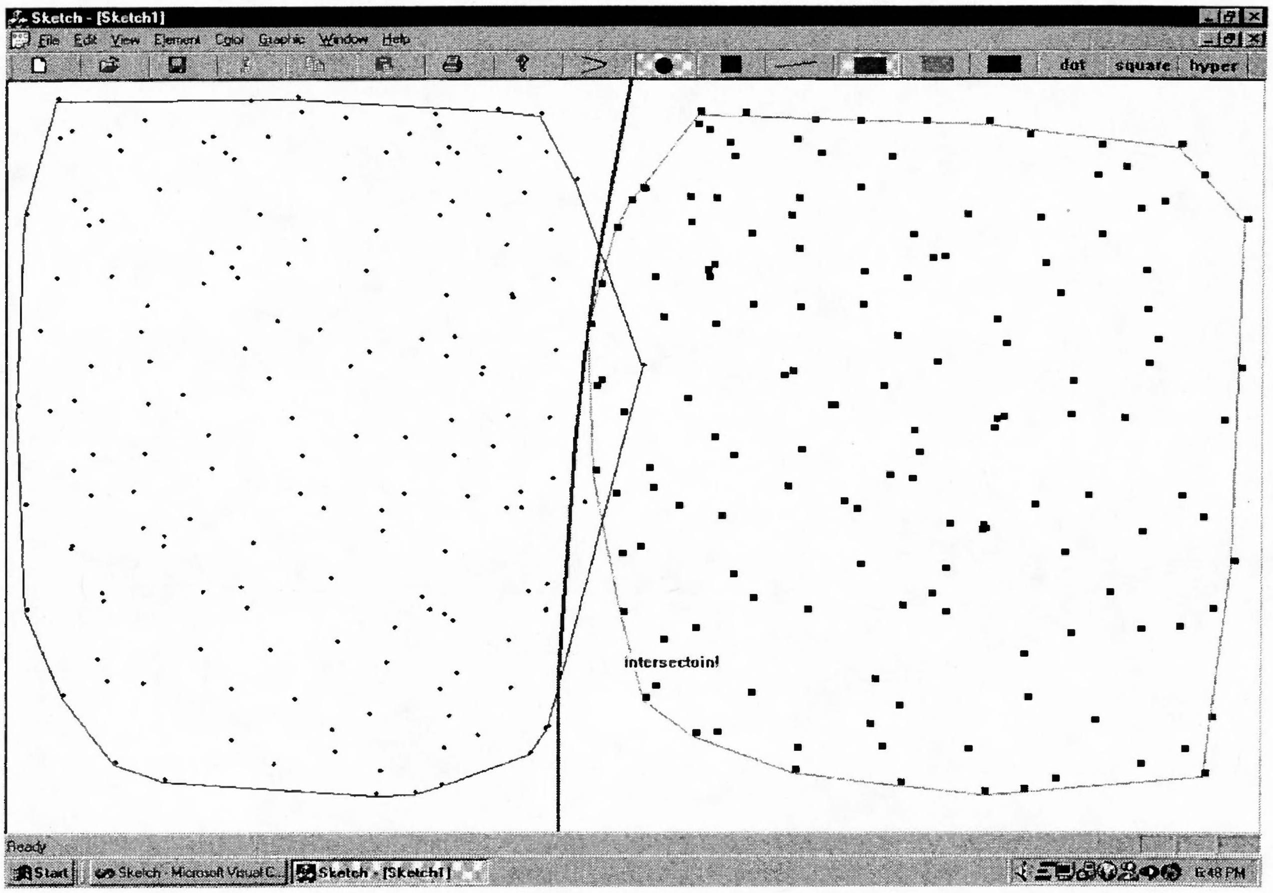
```
0              x + (        -2147483648    y) + (        1677721600    x*x) +
(       2141192192          y*y) + (     0            x*y) + (      -1352[illegible]21248  ) = 0
```

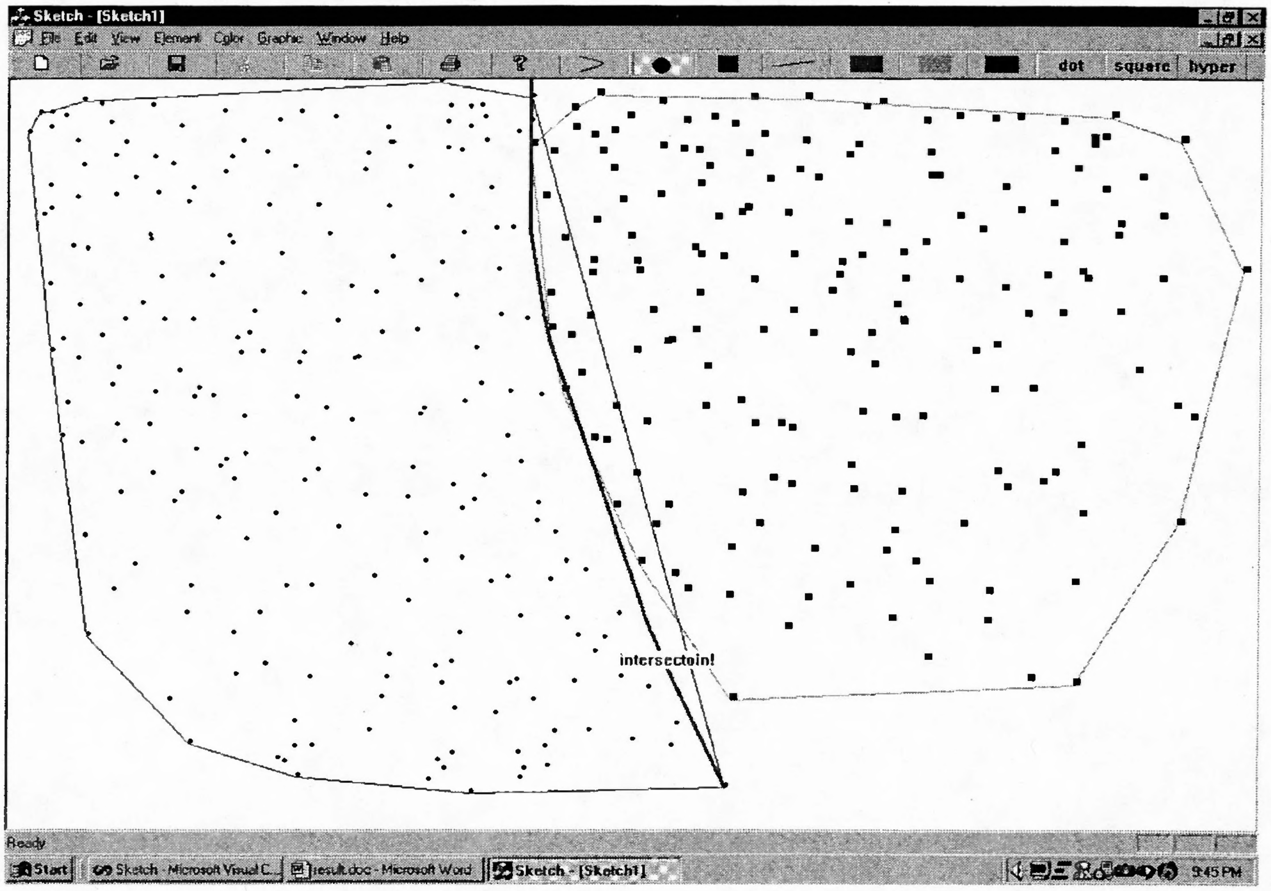

Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
http://ww1.duoweinews.c...
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
7:58 PM

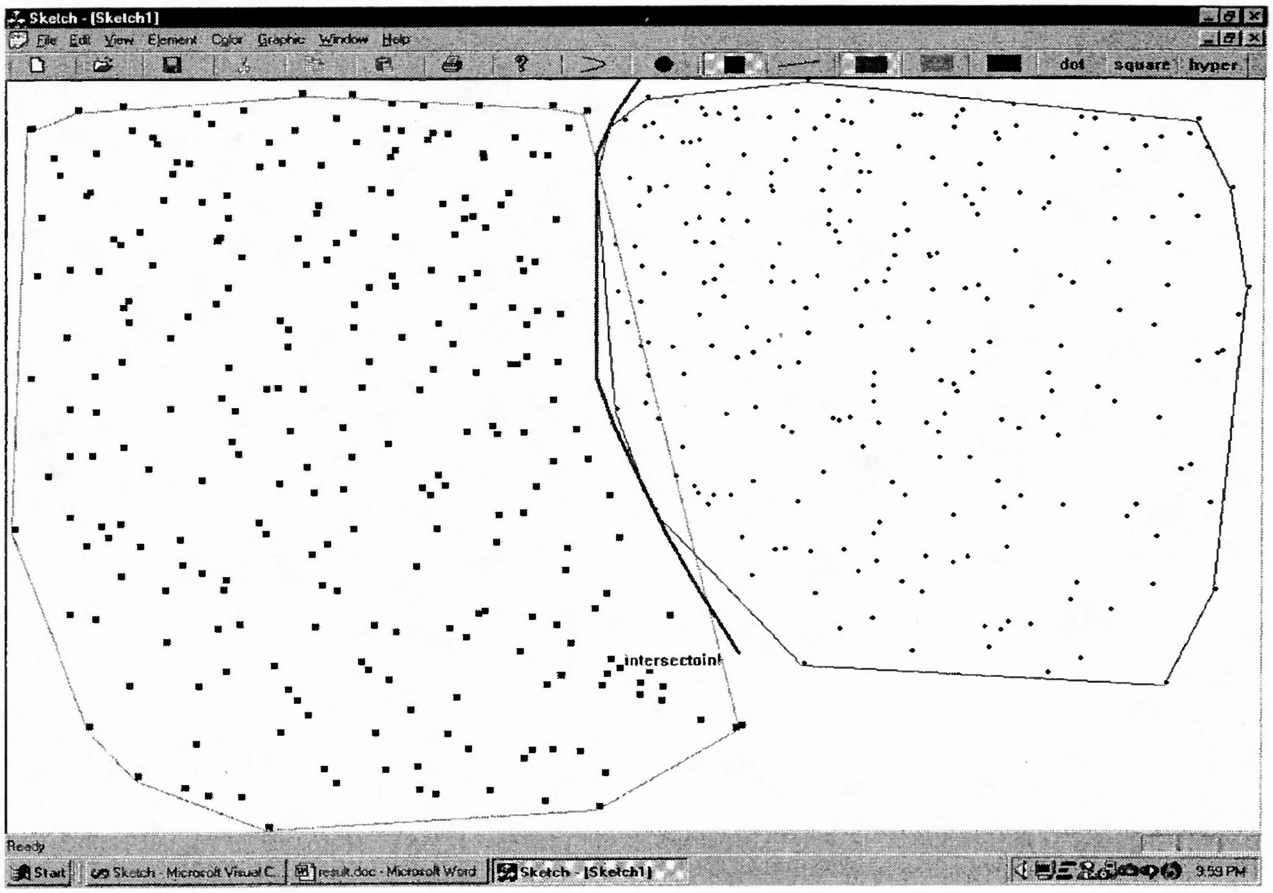
```
1207959552   x + (        268435456    y) + (        558366720    x*x) +
(        -1481113600        y*y) + (      1207959552    x*y) + (     -1464811094  ) = 0
```

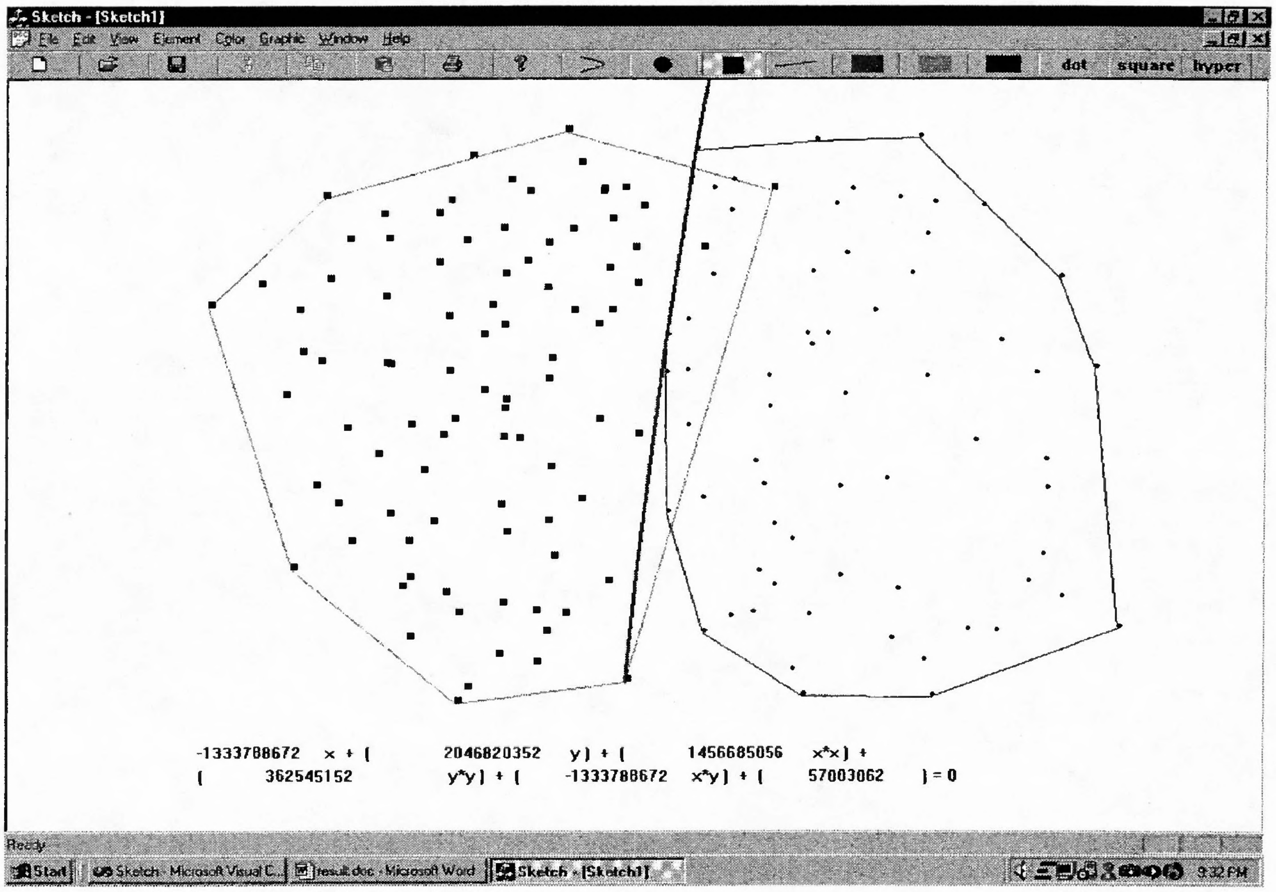

Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
Ready
Start
http://ww1.duoweinews.c...
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
8:20 PM

```
-1476395008   x + (          -1879048192   y) + (          -321912832    x*x) +
(          1212678144            y*y) + (     -1476395008   x*y) + (     -1694141931  ) = 0
```


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
intersectoinl
Ready
Start
Sketch - Microsoft Visual C...
Sketch - [Sketch1]
6:48 PM


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
intersectoin!
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
9:45 PM


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot
square
hyper
intersectoin!
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
9:59 PM


Sketch - [Sketch1]
File Edit View Element Color Graphic Window Help
dot square hyper
Ready
Start
Sketch - Microsoft Visual C...
result.doc - Microsoft Word
Sketch - [Sketch1]
9:32 PM

```
-1333788672   x + (        2046820352   y ) + (        1456685056   x*x ) +
(        362545152          y*y ) + (     -1333788672  x*y ) + (      57003062     ) = 0
```