1. Introduction

In recent 10 years, support vector machines (SVMs) are more and more spectacular. Support vector machines are learning systems [11]. The learning strategy was introduced first by Vapnik and his co-workers in 1979 [9]. An SVM possesses an algorithm, the learning algorithm. By feeding the algorithm with a set of training data, it can determine a classifier or an optimal hyper-plane. Then we use it to classify test data to observe the classification accuracy. If the accuracy is over a threshold, we use the classifier to classify the new-coming unknown data. This is the goal of the development of the SVMs.

The SVM is a principle and very powerful method. It has good performance in a wide variety of applications such as text categorization, image recognition, hand-written digit recognition, and the determination of cancer cell etc [4].

Unfortunately, not all data can be classified by an SVM, since SVM classify data by a linear classifier. To use SVM, we may need to map the input space into a high dimensional feature space to realize the linearity of the classifier. The high dimension of input space will decrease performance of the support vector machine. Therefore, from an efficiency viewpoint, we hope to classify test points based on fewer vector components (we call features in section 3). Feature selection is a way to achieve the goal [2][8].

This paper consists of five sections. In section 2, we describe the construction of the SVM with linear and non-linear cases. We also introduce

method to classify non-linear separable case by SVM. In section 2, we describe the construction of the SVM with linear and non-linear cases. And we introduce another solution by mapping training data into a high dimensional feature space for non-linear case. In section 3, we describe the feature selection. We introduce the strategy proposed by L. Hermes and J. M. Buhmann [6]. We call it the L-J method. The L-J method is used to select features in the input space. In section 4, we propose a method to select features in the feature space, and combine the method with the L-J method to form a two-stage feature selection. An example is given to illustrate our method. Finally, in section 5, some conclusions are discussed.

2. Support Vector Machines (SVMs)

In a classic classification (or pattern recognition) problem, each data unit is represented as a vector $x = (x^1, x^2, ..., x^n)^T \in \Re^n$ consisting of individual components, which represent some properties of the data. First, consider a classification problem with two classes. Suppose that there are $+$ training examples $x_i = (x_i^1, x_i^2, ..., x_i^n)^T \in \Re^n$, each x_i is labeled by $y_i \in \{+1, -1\}$, $i = 1, 2, ..., +$. Vapnik constructed an SVM to separate the training data into two classes, one class containing all the vectors with $y_i = +1$ and the other containing all the vectors with $y_i = -1$.

The SVM separates these two classes by an optimal hyper-plane $w \cdot x + b = 0$, *where* "·" *means the inner product*. w is perpendicular to the hyper-plane. The function $f(x) = w \cdot x + b$ is called the classifier. The optimality of the decision hyper-plane is defined by maximizing the distance of the nearest training example to the hyper-plane. The training of SVM is to find the classifier $f(x) = w \cdot x + b$ so that we can use it to classify data.

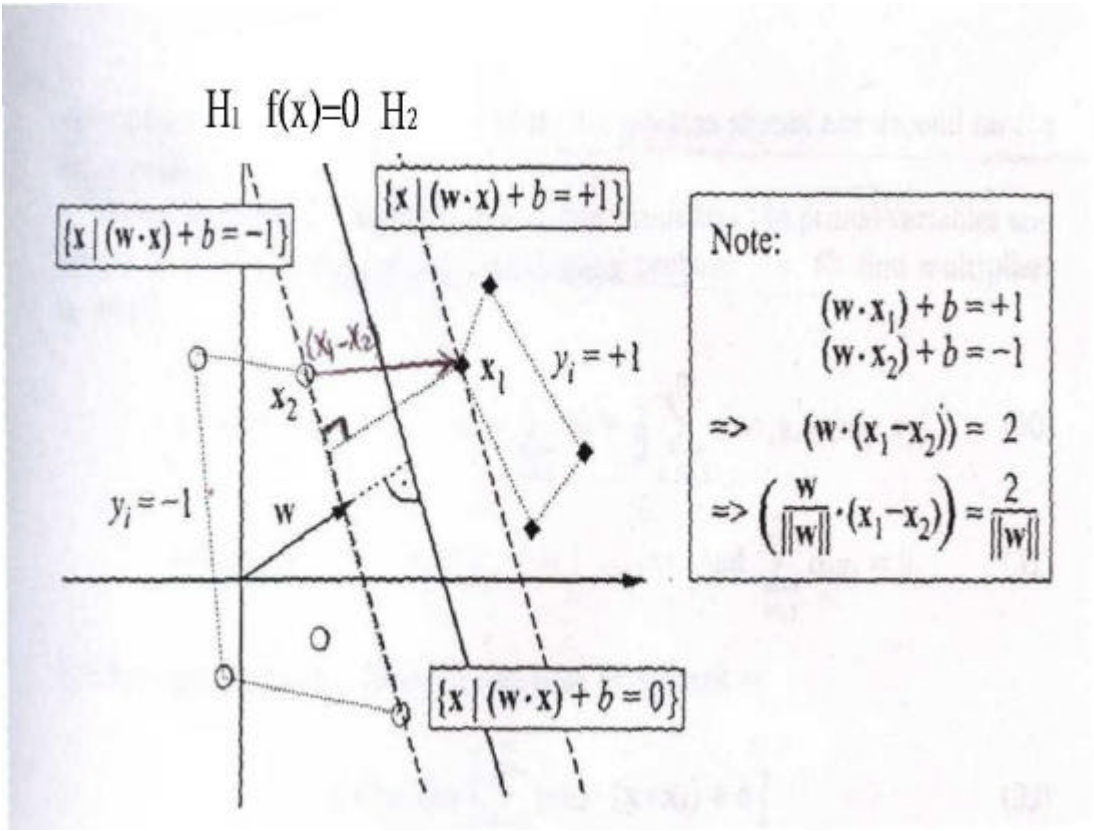

Figure 1. A binary classification problem

In Figure 1, H_1 and H_2 are the hyper-planes containing training points closest to the hyper-plane $f(x) = 0$. There are no training points lying between H_1 and H_2. The distance between H_1 and H_2 is $\frac{2}{\|w\|}$. The distance is called the margin [1]. The hyper-plane is better if the margin is larger.

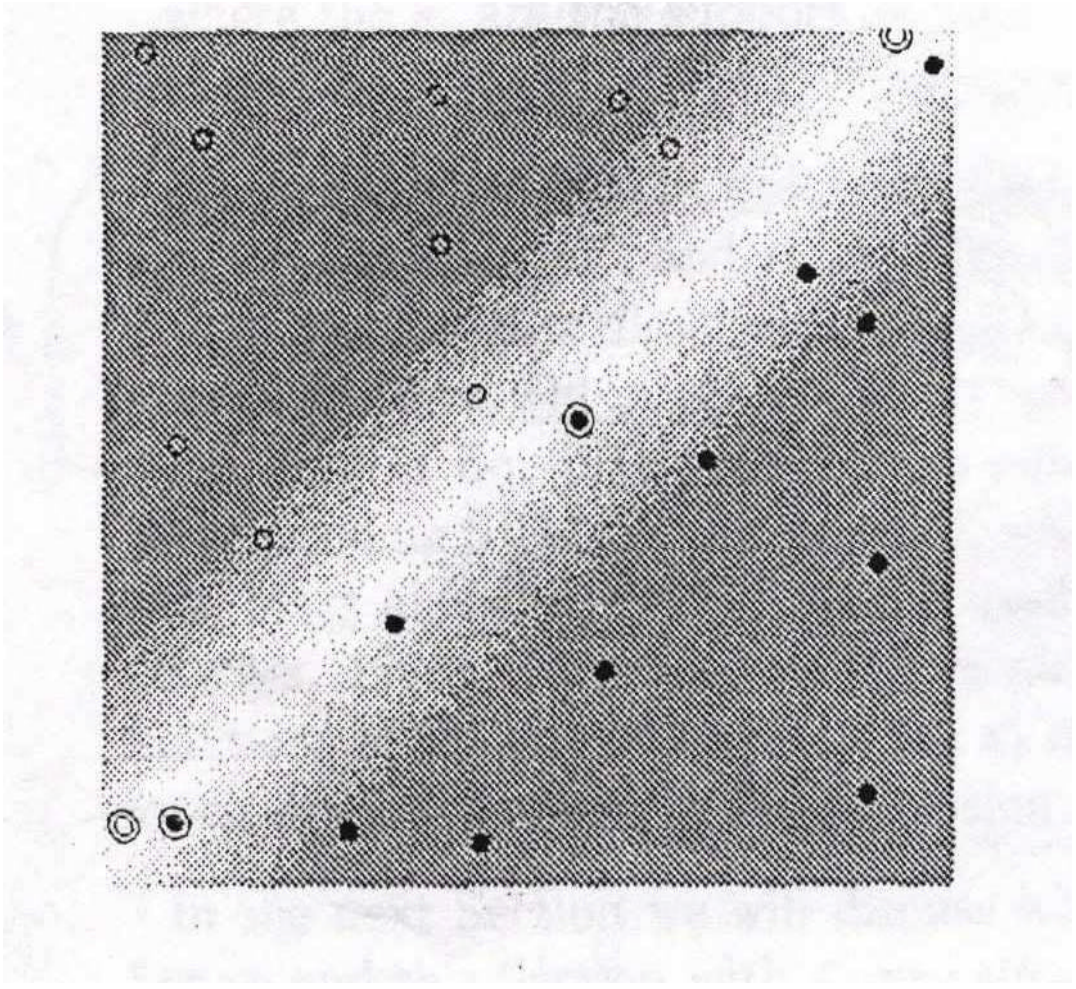

Figure 2. A linearly separable example

Now, consider the two-classes linearly separable pattern problem as shown in Figure 2, where $x_i \in \Re^n$ are labeled by y_i with $y_i \in \{+1, -1\}$, $i = 1, 2, ..., +$. We want to find the optimal hyper-plane to maximize $\frac{2}{\|w\|}$. Without loss of generality, we may assume that

$$\begin{cases} w \cdot x_i + b \geq +1 \quad if \quad y_i = +1, \\ w \cdot x_i + b \leq -1 \quad if \quad y_i = -1. \end{cases}$$

The two conditions can be combined into the following:

$$y_i (w \cdot x_i + b) - 1 \geq 0, \quad \forall i = 1, 2, ..., +.$$

These are constraints to the maximizing margin problem.

Instead of maximizing $\frac{2}{\|w\|}$, we minimize $\frac{\|w\|}{2}$, or equivalently to minimize $\frac{\|w\|^2}{2}$. Then the linearly programming problem is modeled as:

$$\begin{aligned} &Minimize \quad f(x) = \frac{1}{2}\|w\|^2 \\ &subject \ to \quad y_i (w \cdot x_i + b) - 1 \geq 0, \forall i = 1, 2, ..., +. \end{aligned} \tag{1}$$

Now, we introduce non-negative Lagrange multipliers a_i, $i = 1, 2, ..., +$, and get Lagrangian:

$$\begin{aligned} &\underset{w,b}{Minimize} \quad L_P : \frac{1}{2}\|w\|^2 - \sum_{i=1}^{+} \alpha_i y_i (w \cdot x_i + b) + \sum_{i=1}^{+} \alpha_i \\ &subject \ to \quad \alpha_i \geq 0, \forall i = 1, 2, ..., +. \end{aligned} \tag{2}$$

The optimization programming is called the **primal form** [1].

Since the solution of (2) satisfies $\frac{\partial L_P}{\partial w} = 0$ and $\frac{\partial L_P}{\partial b} = 0$, therefore

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

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

Substituting them into L_P, we obtain

$$\begin{aligned} &\underset{\alpha_i}{Maximize}\ L_D : \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_j \cdot x_i \\ &subject\ to\ \alpha_i \geq 0, \forall i = 1,\ 2,\ ...,\ +, \\ &\qquad \sum_{i=1}^{+} \alpha_i y_i = 0. \end{aligned} \tag{3}$$

(3) is the **dual form** of the problem [5].

Since the problem minimizes L_P with respect to w, b and maximizes L_D with respect to α_i, the optimization problem is equivalent to find the saddle point of L_P and L_D. We call the points in the solution with $\alpha_i > 0$ **support vectors** (or **s.v.**). The support vectors lie on either H_1 or H_2. Besides, all other training points have zero value of α_i, and lie on either H_1 or H_2, or on side of H_1 or H_2 [1]. In Figure 2, the points marked by ◎ or ⊙ are the support vectors and the others not. Thus, we have

$$w = \sum_{i=1}^{+} \alpha_i y_i x_i = \sum_{i \in I_{s.v.}} \alpha_i y_i x_i ,$$

and $f(x) = w \cdot x + b = \sum_{i \in I_{s.v.}} \alpha_i y_i x_i \cdot x + b$,

where $I_{s.v.} = \left\{ j \;\middle|\; x_j \ \ is \ \ an \ \ s.v. \right\}$.

Furthermore, the support vectors are the vectors that are closest to the hyper-plane $f(x)=0$. $|f(x)|$ can be viewed as the distance of vector x to the optimal hyper-plane, and the classifier can be completely determined by the support vectors.

Lagrange proposed in 1797 the Lagrangian theory to characterize the solution of an optimization problem with no inequality constraints. Kuhn and Tucker proposed extended the theorem to optimization problems with inequality constraints. The theorem provides a necessary and sufficient condition for the optimal solution of the optimization problem. The condition is known as the **Karush-Kuhn-Tucker** (**KKT**) **conditions [4].** The theorem provides a way to solve an SVM problem. We list the theorem here for reference.

Theorem (Kuhn-Tucker)*: Assuming that there is an optimization problem with convex domain* $\Lambda \subseteq \Re^n$*:*

$$\begin{aligned} &\textit{Minimize} && f(x),\ \ x \in \Lambda, \\ &\textit{subject to} && q_i(x) \le 0,\ \ i = 1,\ 2,\ ...,\ k, \\ & && s_j(x) = 0,\ \ j = 1,\ 2,\ ...,\ m, \end{aligned}$$

with $f \in C^1$ *convex and* q_i, s_j *affine, we can obtain the Lagrangian function* $L(x, \boldsymbol{a}, \boldsymbol{b}) = f(x) + \sum_{i=1}^{k} \boldsymbol{a}_i q_i(x) + \sum_{j=1}^{m} \boldsymbol{b}_j s_j(x)$. *The necessary and sufficient conditions for a normal point* x^* *to be an optimal solution*

are the existence of α^*, β^* *such that*

$$\frac{\partial L(x^*, \alpha^*, \beta^*)}{\partial x} = 0,$$

$$\frac{\partial L(x^*, \alpha^*, \beta^*)}{\partial \beta} = 0,$$

$$q_i(x^*) \le 0, \quad i = 1, \ 2, \ ..., \ k,$$

$$\alpha_i^* \ge 0, \quad i = 1, \ 2, \ ..., \ k,$$

$$\alpha_i^* q_i(x^*) = 0, \quad i = 1, \ 2, \ ..., \ k.$$

These equations are called the ***KKT conditions****. The last equation is known as the* ***KKT complementarity condition***.

By the theorem, to solve an optimization SVM problem is equivalent to find the solution (w^*, b^*, α^*) for the KKT conditions. For example, consider the optimization problem (1), the KKT conditions are

$$w - \sum_{i \in I_{s.v.}} \alpha_i y_i x_i = 0,$$

$$- \sum_{i \in I_{s.v.}} \alpha_i y_i = 0,$$

$$y_i (w \cdot x_i + b) - 1 \ge 0, \forall i = 1, \ 2, \ ..., \ +,$$

$$\alpha_i \ge 0, \forall i = 1, \ 2, \ ..., \ +,$$

$$\alpha_i [y_i (w \cdot x_i + b) - 1] = 0, \forall i = 1, \ 2, \ ..., \ +.$$

The last one is the **KKT complementarity condition** to the primal form. Since $w = \sum_{i \in I_{s.v.}} \alpha_i y_i x_i$, b can easily be solved from the KKT complementarity condition by choosing any i for which $\alpha_i > 0$.

Though we consider above only the linearly separable case. The concept

can also be extended to the linearly non-separable case, i.e., when the linear programming (1) has no feasible solution. Figure 3 shows a linearly non-separable example. The cross indicates the point that is wrongly classified.

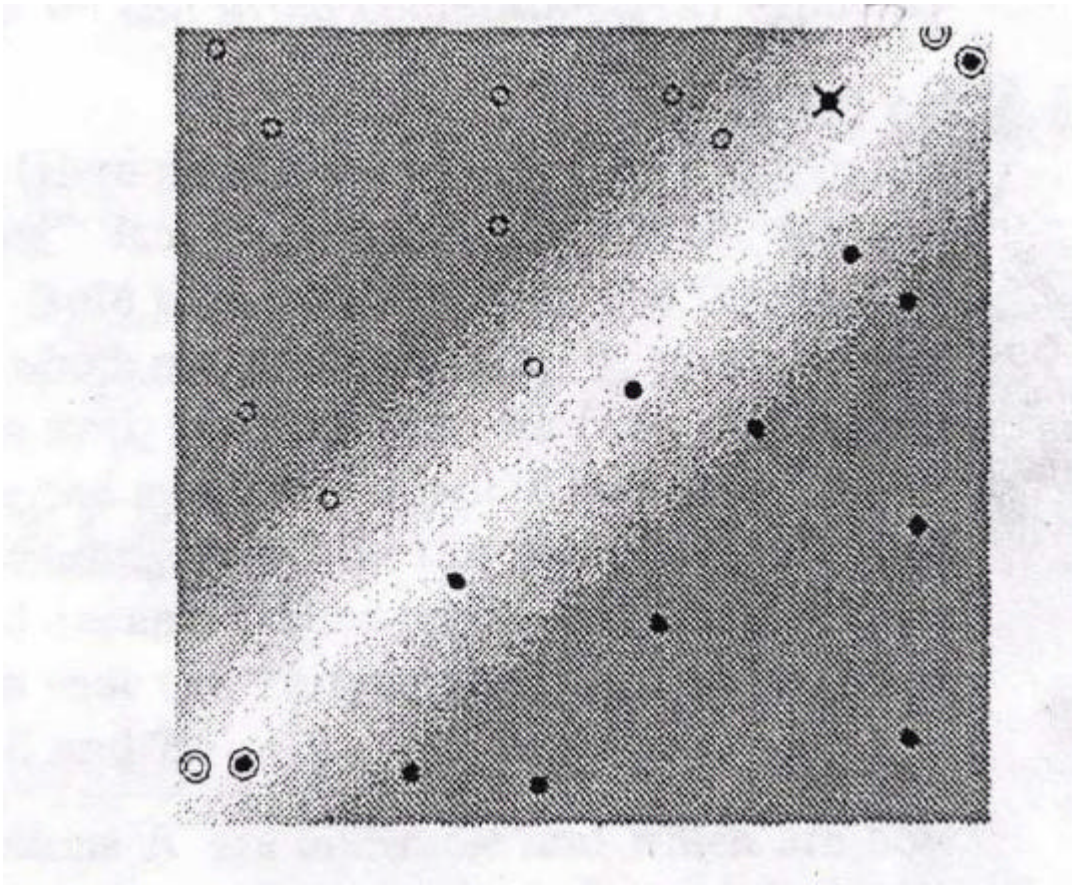

Figure 3. A linearly non-separable example

For the linearly non-separable case, assume that the point x_i is wrongly

classified. Then x_i must violate the constraint: $y_i(w \cdot x_i + b) \geq 1$. The constraint is changed to be $y_i(w \cdot x_i + b) \geq 1 - \xi_i \ \ with \ \xi_i > 0$ so that x_i satisfies it. We may introduce non-negative slack variables $\xi_i \geq 0, \ \forall i = 1, \ 2, \ ..., \ +$, in the constraints of (1) [3]. Then the constraint of (1) is weakened to

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \ \ \forall i = 1, \ 2, \ ..., \ +.$$

The objective function is rewritten to keep the constraint violation as small as possible. In addition, the original objective function is still required to minimize $\frac{\|w\|^2}{2}$. Thus, we observe the new objective function for non-linear case:

$$\begin{aligned} &Minimize \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{+}\xi_i \\ &subject \ \ to \ \ y_i(w \cdot x_i + b) - 1 + \xi_i \geq 0, \ \ i = 1, \ 2, \ ..., \ +, \\ &\qquad\qquad \xi_i \geq 0, \ \ i = 1, \ 2, \ ..., \ +. \end{aligned} \tag{4}$$

C is a parameter chosen by user to assign a penalty to errors. By introducing two non-negative Lagrange multipliers α_i, μ_i for each i. The primal Lagrangian changes to

$$\begin{aligned} &Minimize \quad L_P : \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{+}\xi_i - \sum_{i=1}^{+}\alpha_i\{y_i(w \cdot x_i + b) - 1 + \xi_i\} - \sum_{i=1}^{+}\mu_i\xi_i \\ &subject \ \ to \qquad 0 \leq \alpha_i \leq C, \ \ i = 1, \ 2, \ ..., \ +, \\ &\qquad\qquad \mu_i \geq 0, \ \ i = 1, \ 2, \ ..., \ +. \end{aligned} \tag{5}$$

The K.K.T. conditions for the optimization programming (4) are

$$\frac{\partial L_P}{\partial w} = w - \sum_{i \in I_{s.v.}} \alpha_i y_i x_i = 0,$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i \in I_{s.v.}} \alpha_i y_i = 0,$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0, \forall i = 1, 2, \ldots, +,$$

$$y_i (w \cdot x_i + b) - 1 + \xi_i \geq 0, \forall i = 1, 2, \ldots, +,$$

$$\xi_i \geq 0, \forall i = 1, 2, \ldots, +,$$

$$C \geq \alpha_i \geq 0, \forall i = 1, 2, \ldots, +,$$

$$\mu_i \geq 0, \forall i = 1, 2, \ldots, +,$$

$$\alpha_i \{ y_i (w \cdot x_i + b) - 1 + \xi_i \} = 0, \forall i = 1, 2, \ldots, +,$$

$$\mu_i \xi_i = 0.$$

The last two equations are the KKT complementarity conditions for the non-linear programming.

In 1992, Boser, Guyon and Vapnik applied the method for pattern recognition learning proposed by Aizerman (1964) to solve the linearly non-separable problem. First, they map the training examples to a higher dimensional space H, called **feature space**, by a mapping

$$\Phi : \Re^n \rightarrow H .$$

H is often referred to as a Hilbert space. The object of mapping into feature space is to transform a non-linear problem in the input space to a linear one in the feature space. Then we can linearly separate the points in the feature space.

Note that $f(x) = \sum_{i \in I_{s.v.}} \alpha_i y_i x_i \cdot x + b$ for the linear case and

$f(x) = \sum_{i \in I_{s.v.}} \alpha_i y_i \Phi(x_i) \cdot \Phi(x) + b$ for the non-linear case. The training vectors never appear isolated but always in the form of inner products between pairs of vectors. Vapnik et al. defined a new function, the **kernel function**, to replace the inner product for non-linear pattern problem. Then one can implicitly perform a non-linear mapping to a high dimensional feature space. The kernel function offers an alternative way to increase the computational power of the linear learning machines by projecting the data into a high dimensional feature space.

*Definition (**kernel function**): Let H be a feature space. Given an input space $S \subset \Re^n$, the **kernel function** K is defined by $K(x, z) = (\Phi(x) \cdot \Phi(z)), \ \forall x, \ z \in S$, i.e., the inner product on the feature space, where Φ is a mapping from $\Re^n$ into H.*

When we train the SVM with a set of training data $\{x_1, x_2, ..., x_+\}$, the kernel function is $K(x_i, x_j) = (\Phi(x_i) \cdot \Phi(x_j))$. Vapnik showed that the kernel function $K(x_i, x)$ should satisfy the following **Mercer's condition** (Courant and Hilbert, 1953) [9].

***Mercer's condition**: Let S be a finite input space (consisting of training points and test points) and K(x,z) a symmetric function on S. Then K(x,z) is a kernel function if and only if the matrix $G = \left[K(x_i, x_j)\right]_{i,j=1}^{+} \in \Re^{+\times+}$ is **positive semi-definite**, i.e.,*

$$\int_{S\times S} K(x,z)g(x)g(z)dxdz > 0 \quad if \quad \int_S g^2(x)dx < \infty \quad (or \quad g \in L_2(S)\ ).$$

The Mercer's condition for function K is a necessary and sufficient condition for K to be a kernel function.

With the kernel function K, we can rewrite the classifier in the feature space for the non-linear case:

$$f(\Phi(x)) = w \cdot \Phi(x) + b = \sum_{i \in I_{s.v.}} \alpha_i y_i \Phi(x_i) \cdot \Phi(x) + b = \sum_{i \in I_{s.v.}} \alpha_i y_i K(x_i, x) + b,$$

where $w = \sum_{i \in I_{s.v.}} \alpha_i y_i \Phi(x_i)$. Figure 4 shows an example where the data cannot be separable by a linear hyper-plane, but can be linearly separable in the feature space. Although the linearity is satisfied, the computation turns to be more complex since the dimension of the input space increases. One way to solve the problem is by feature selection. We will introduce the concept in next section.

SVMs provide a new way for pattern recognition. Although the SVM classifiers described above are binary classifier, they can be easily extended to classify the multi-class. For an m-class problem, m support vector machines are constructed. Each support vector machine is used to separate a class from the others.

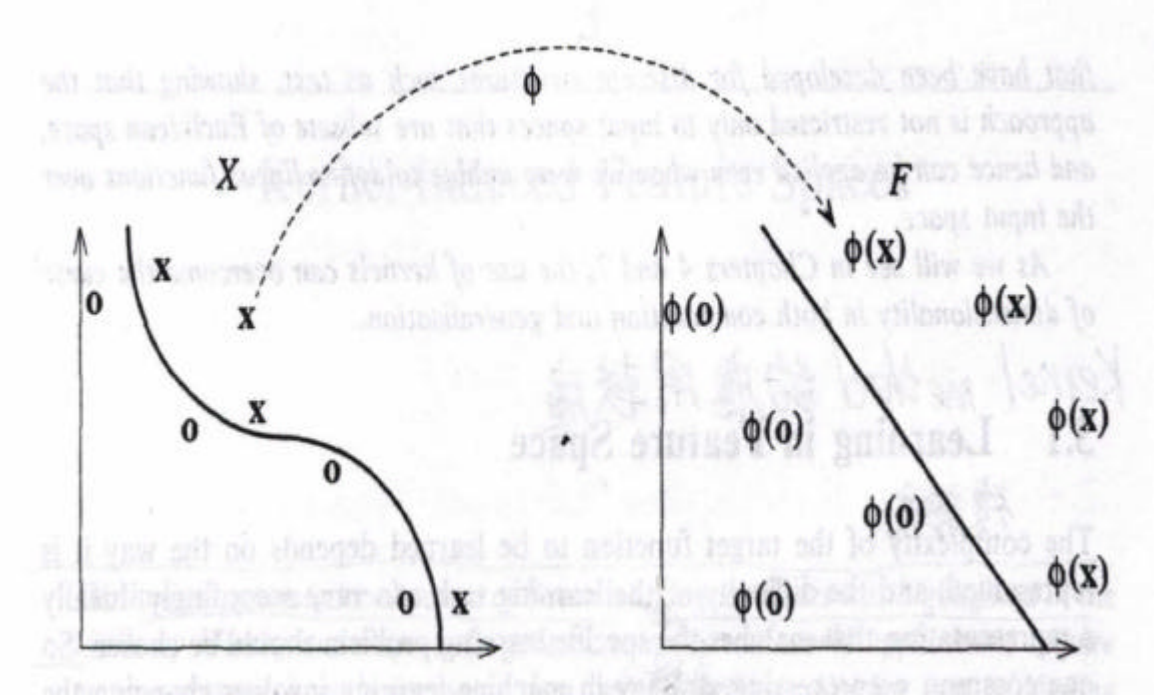

Figure 4. A feature map can simplify the classification task

3. Feature Selection

Figure 5 shows the process to derive the decision rule to recognize pattern. In the field of pattern recognition, the raw data describing the physical system are referred to as the measurement space. The measurement space can be viewed as the mathematical model of the physical system. Some pattern classification algorithm would be applied to the pattern space. The pattern space may be the same as the measurement space. **Feature selection** (or **preprocessing**) can be viewed as a transformation between the measurement space and the pattern space [2] [8].

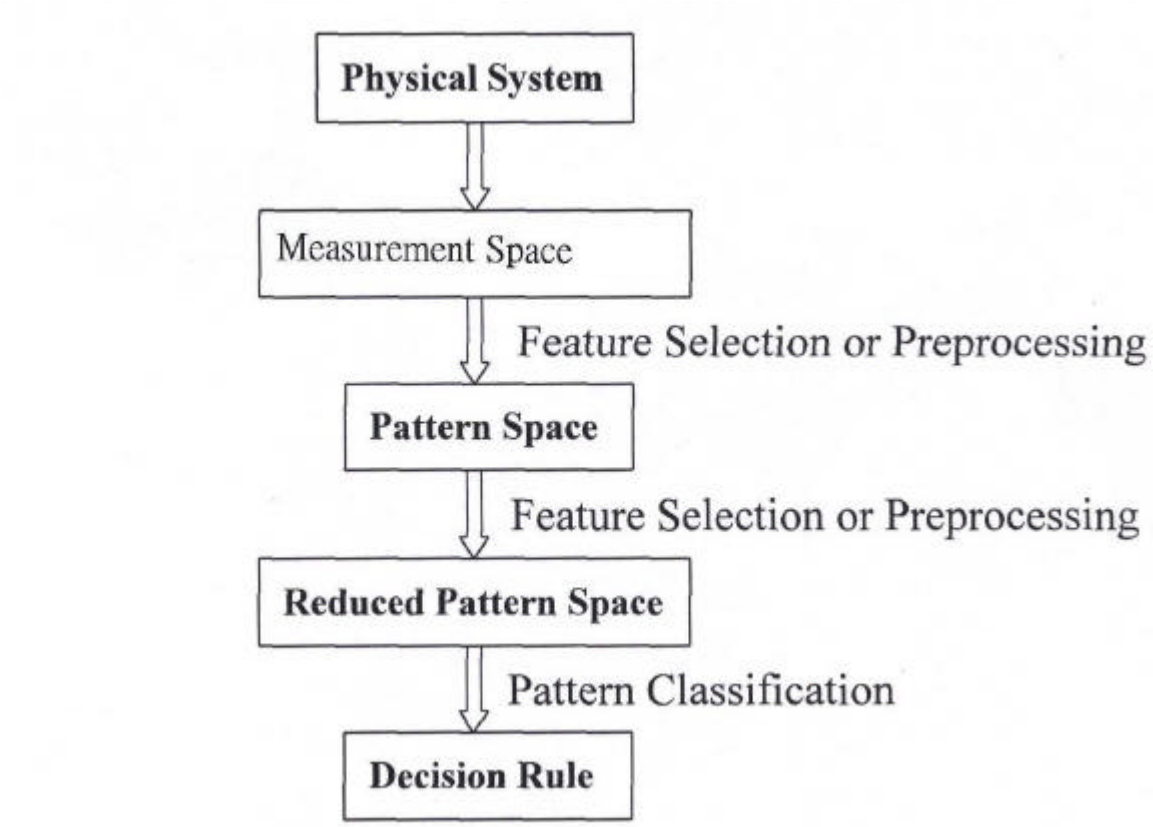

Figure 5. Stages in the derivation of the decision rule

Feature selection is a process by which a sample in the measurement space is described by a finite and usually smaller set of numbers called features, say x^1, x^2, ..., x^n. The features become components of the pattern space. And the feature selection is regarded as a procedure to determine that which variables (attributes) are to be measured first or last. Feature selection may be a multistage process to enhance the accuracy or performance of

classification.

The features selected should satisfy: (a) they properly describe the pattern, (b) they are easily to handle, and (c) they are invariant to translation and rotation of the training examples. It is difficult to select features satisfy all three requirements at the same time since each sample has its own particular characteristics. The comparability (or similarity) of features among samples can influence the process of feature selection.

Feature selection can reduce computational complexity. One way of feature selection is to identify a smaller feature set that still retains the essential information of the original attributes. There are some criteria [8]:

(1) low dimensionality,

(2) retention of sufficient information,

(3) enhancement of distance in pattern space as a measure of the similarity of physical patterns, and

(4) consistency of feature throughout the samples.

In 2000, L. Hermes and J. M. Buhmann proposed a strategy to rank individual components according to their influence on the decision hyper-plane [6]. We call it the **L-J method** in this paper. The L-J method is to select a suitable subset of features to replace the original one. When mapping the input data into a feature space, the dimension of the feature space may be raised too high. This is known as the over-fitting problem. Sometimes, feature selection can also reduce the risk of over-fitting caused by the proliferation of features.

Each component represents some information of the data, for instance, the income, age, ID, and the weight etc of a customer. In view of efficiency, the classification should avoid any unnecessary computation by reducing the input dimension as few as possible. But diminishing the input dimension will confront the danger of descending the classification accuracy. Thus, we should know which component is relevant to a given classification model. Note that replace the original set of features without significant loss of classification accuracy. Thus, data components discarded should be limited to those ones that do not carry any information about the classifier. The L-J method proposed an approach to evaluate the importance of a feature to the classifier.

The L-J method is a selection approach that defines scores for the available features at training. The strategy ranks the features according to their influence on the decision hyper-plane. The influence of the jth feature is evaluated by the angle between $\nabla f(x)$ and e_j. In brief, the L-J method is base on the idea that: if feature x^i is not important to the distance of x to the classifier $f(x)$, then the angle between $\nabla f(x)$ and e_j should approximate to a right angle.

At first, with a given training set they train the support vector machine by using complete data components. After constructing the classifier $f(x)$, they estimated the importance of separate feature components to $f(x)$. Note

that $|f(x)|$ can be viewed as the distance from x to the hyper-plane. The gradient of $f(x)$ at point x, $\nabla f(x)$, is perpendicular to the optimal hyper-plane. If the projection of the unit vector e_j on $\nabla f(x)$ is small, the jth component of x has little influence on the distance from x to the decision hyper-plane. They calculate the angles $\theta_j(x_i)$ between $\nabla f(x_i)$ and the unit vectors e_j, $i=1, 2, ..., n$, *and* $j=1, 2, ..., n$ by

$$\theta_j(x) = \min_{b\in\{0,1\}} \{b\pi + (-1)^b \cdot \arccos(\frac{\nabla f(x)\cdot e_j}{\|\nabla f(x)\|})\} \in [0,\frac{\pi}{2}].$$

Moreover,

$$\nabla f(x) = \sum_{i\in I_{s.v.}} \alpha_i y_i \nabla_x K(x_i, x)$$

since $$f(x) = \sum_{i\in I_{s.v.}} \alpha_i y_i K(x_i, x) + b. \qquad (6)$$

If $\theta_j(x) \approx \frac{\pi}{2}$, the feature x^j has only weak influence on the assignment $f(x)$ of x. Smaller $\theta_j(x)$ corresponds to more influential feature. L. Hermes and J. M. Buhmann added the number of points in order to permit some error on support vectors. They included all vectors within a δ-region around the support vectors, that is, they chose the points x_i satisfies $|f(x_i)-1| < \delta$. Let I_δ be the indices set of all training vectors which match the condition and curtail $\theta_j(x_i)$ by θ_{ij}. To consider the influence of the jth component of all support vectors, L. Hermes and J. M. Buhmann average the angles θ_{ij} over I_δ for each j. Then normalize the average to a value

$\tilde{f}_j \in [0,1]$. They defined $\tilde{f}_j \in [0,1]$ as:

$$\tilde{f}_j = 1 - \frac{\sum_{i \in I_d} f_{ij}}{|I_d|} \cdot \frac{2}{p}.$$

$\tilde{f}_j$ can be used to measure the importance of the jth features. If $\tilde{f}_j$ is large, x^j is ranked first. After ranking the features, we choose the components that have smaller $\tilde{f}_j$ if we have to drop some ones.

4. Our Results

In the L-J method, the criterion for ranking features depends on the value of $\tilde{f}_j$. The L-J method is used to select features in the input space. We extend the concept of the L-J method to derive a new approach for feature selection in the feature space. Our result can avoid the situation that the L-J method cannot decide the desertion feature. Moreover, we combine the L-J method and our result to form a two-stage feature selection approach.

We describe the two-stage strategy as following:

Phase 1 (L-J method):

Step 1: Construct a classifier from a training set.

Step 2: Calculate the angles f_{ij}, $j = 1, 2, ..., n$ between $\nabla f(x)$ and e_j

for $i = 1, 2, ..., +$.

Step 3: Normalize f_{ij}, $\forall i = 1, 2, ..., +$, to get the score of the jth feature $\tilde{f}_j$ for $j = 1, 2, ..., n$.

Step 4: Rank the features in the input space according to $\tilde{f}_j$, $j = 1, 2, ..., n$.

After dropping the unnecessary features, we map the input data into a feature space. Then we enter the Phase 2.

Phase 2:

Step 1: We measure q_j, the angles between the weight vector w and the unit vectors e_j:

$$q_j = \arccos\left|\frac{w \cdot e_j}{\|w\|}\right| = \arccos\left|\frac{w^j}{\|w\|}\right| \in [0, \frac{p}{2}], \quad j = 1, 2, ..., m \qquad (5)$$

Step 2: Rank features in the feature space according to q_j.

Step 3: Dropping features in the feature space according to q_j if necessary.

As mentioned in the L-J method, if e_i is roughly orthogonal to w, then the feature x^i does not has a great effect on the distance to the decision hyper-plane. The difference to the L-J method is that the smaller value of q_j

indicates more important feature x^j. We can use q_j to score the importance of the jth-component to the classifier $f(x)$.

We call $\nabla f(x)$ the **distance vector** of x in this paper since it is along the direction perpendicular to the decision hyper-plane. We compare the distance vector of x_j, $\nabla f(x_j)$, with w. In the linear case, $\nabla f(x_i)$ is parallel to w. Therefore, all $f_j(x_i)$ are the same $\forall i = 1, 2, ..., +$, and so $q_j = f_j(x_i)$ *for all* i. In the linearly non-separable case, assume that the original training data are $\{x_1, x_2, ..., x_+\} \subset \Re^n$, and map these data into an m dimensional feature space H, m > n.

$\Re^n \xrightarrow{\Phi(X)} H$.

The training data $\{\Phi(x_1), \Phi(x_2), ..., \Phi(x_+)\} \subset H$ can be linearly separable by the hyper-plane $f(\Phi(x)) = w \cdot \Phi(x) + b = 0$ in the feature space. We view the non-linear case as a new linearly separable problem in the feature space.

The smaller q_j, the larger the estimated importance of the feature is. Then features are ranked according to q_j. Once we decide to dispense with some component, we replace the component by suitable value (depends on the form of $\Phi(x)$) to insure that the test processes still perform in the feature space. In the following example, we will see the weakness of the L-J method. We show that how does our result supplement the L-J method.

Example.

Suppose that there are six training examples in the input space $\Re^2$: $x_1=(1,\ -1)^T$, $x_2=(0,\ 1)^T$, $x_3=(2,\ 0)^T$, $x_4=(-3,\ 0)^T$, $x_5=(-1,\ -1)^T$, $x_6=(0,\ 2)^T$ with $y_1=+1$, $y_2=+1$, $y_3=-1$, $y_4=-1$, $y_5=+1$, $y_6=-1$.

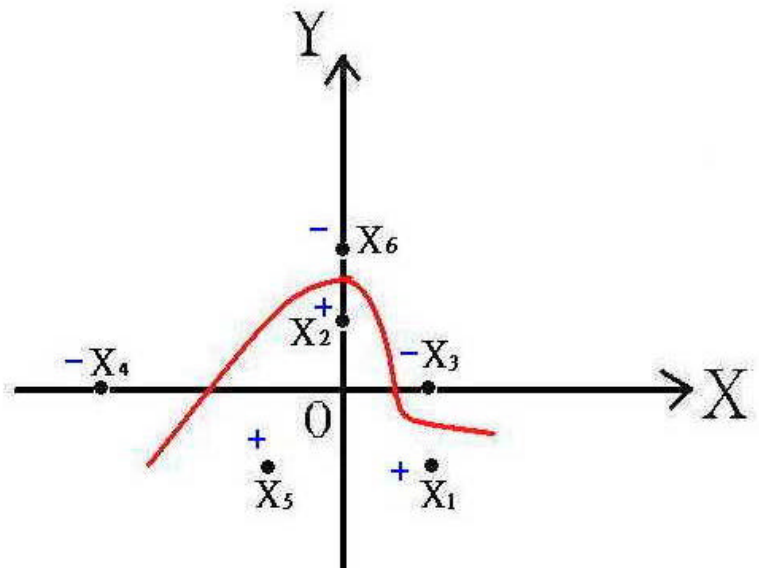

Figure 6. The non-linearly separable graph for example 1

From Figure 6, this is a non-linear model. We map the training data to $\Re^3$ by $\Phi(x)=(x^{1^2},\ \sqrt{2}x^1x^2,\ x^{2^2})^T$, where $x=(x^1,\ x^2)^T$. Then the kernel function is $K(x_i,x_j)=(x_i\cdot x_j)^2$, $i,\ j\in\{1,\ 2,\ ...,\ 6\}$. The training data in the

feature space are $\Phi(x_1) = (1, -\sqrt{2}, 1)^T$, $\Phi(x_2) = (0, 0, 1)^T$, $\Phi(x_3) = (4, 0, 0)^T$, $\Phi(x_4) = (9, 0, 0)^T$, $\Phi(x_5) = (1, \sqrt{2}, 1)^T$, $\Phi(x_6) = (0, 0, 4)^T$, as shown in Figure 7. $\Phi(x_1)$, $\Phi(x_2)$, and $\Phi(x_5)$ are under the plane L : $x + z - 3 = 0$.

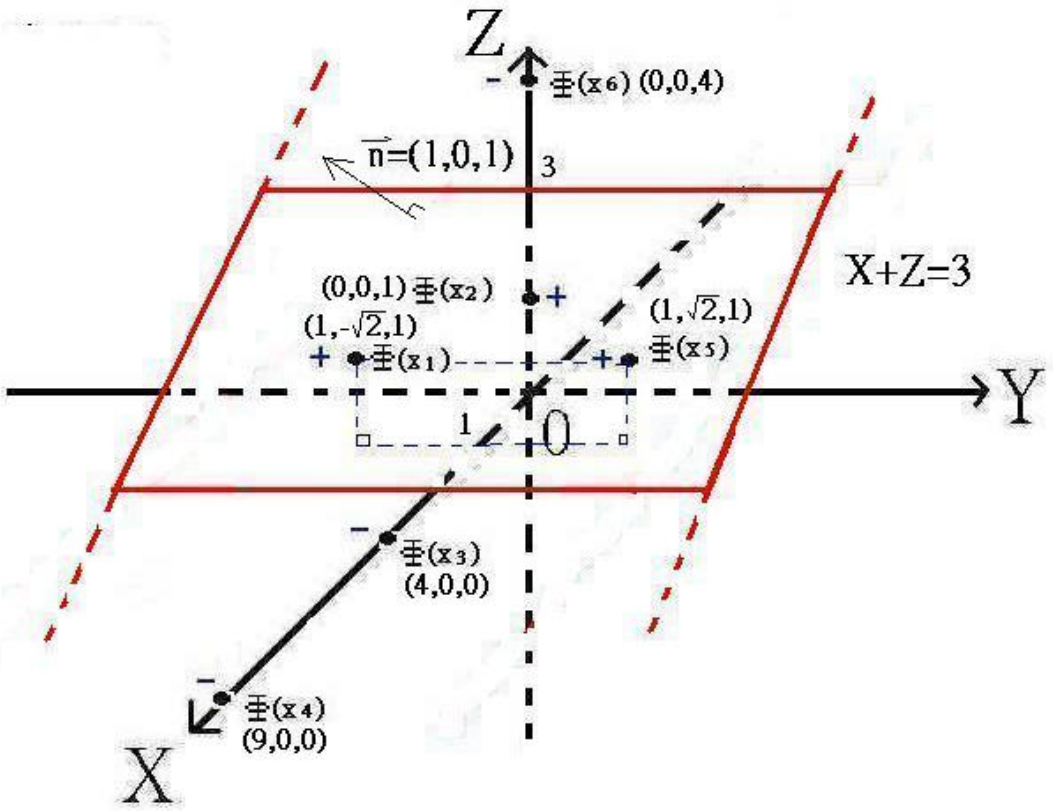

Figure 7. The training data in the feature space

Let us see along the y-axis, that is, project the graph onto the X-Z plane. We can observe a classifier from the geometric graph (Figure 8). The

classification hyper-plane is $x+z-3=0$, i.e., $w=(1,\ 0,\ 1)^T$ *or* $(-1,\ ,0,\ -1)^T$ and $b=-3$. Now, we want to explain the classifier is the optimal hyper-plane. From figure 8, the plane H_1 : $x+z=2$ contains $\Phi(x_1),\ \Phi(x_5)$, and H_2 : $x+z=4$ contains $\Phi(x_3),\ \Phi(x_6)$. The distance between H_1 and H_2 is $\sqrt{2}$. The two classes separated by L are two convex sets, C_1 and C_2. C_1 is below H_1 and C_2 is above H_2. We call the segment connecting $\Phi(x_1)$ and $\Phi(x_5)$ by s_1, and the segment connecting $\Phi(x_3)$ and $\Phi(x_6)$ s_2.

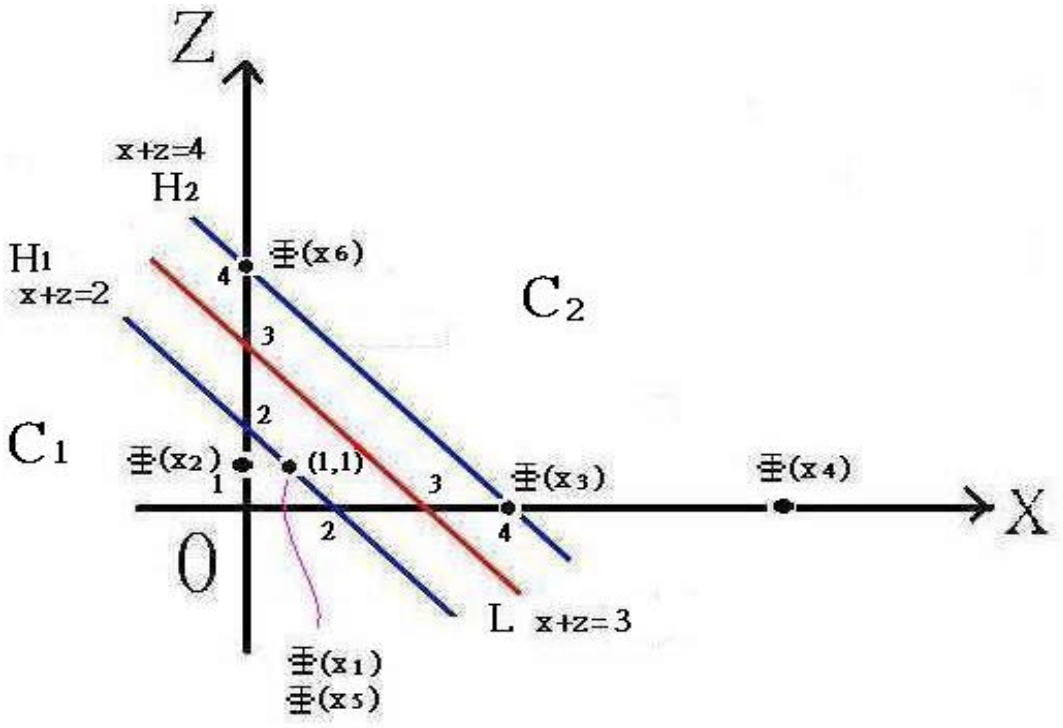

Figure 8. The X-Z plane of Figure 7

In this case, s_1 is on the bounderary of C_1. Similarly, s_2 is on the bounderary of C_2. So, the distance of the two convex sets is $\sqrt{2}$. Note that the distance of any two other convex sets is never larger than $\sqrt{2}$ if one contains s_1 and the other contains s_2, respectively. Thus, there is no other classifier has margin more than $\sqrt{2}$. $L: x+z-3=0$ is the optimal hyper-plane in the example.

Since the α_i of vectors not on the margin are 0. We only need to consider the vectors on the plane, i.e., $\Phi(x_1)$, $\Phi(x_3)$, $\Phi(x_5)$ and $\Phi(x_6)$. Solve the problem:

$$\begin{cases} \sum_{i=1}^{6} \alpha_i y_i = 0, \\ w = \sum_{i=1}^{6} \alpha_i y_i \Phi(x_i). \end{cases}$$

We will obtain that $\alpha_1 = \alpha_3 = \alpha_5 = \alpha_6 = \frac{1}{2}$ and $\alpha_2 = \alpha_4 = 0$. Substituting those α_i into equation (6), we get $\nabla f(x_1) = (-2,\ 2)^T$, $\nabla f(x_3) = (-4,\ 0)^T$, $\nabla f(x_5) = (2,\ 2)^T$, and $\nabla f(x_6) = (0,\ -4)^T$. By the L-J method, we have $\tilde{f}_1 = \tilde{f}_2 = \frac{1}{2}$. In this model, we cannot rank the features in the input space by the L-J method.

With our approach, we get $q_1 = q_3 = \frac{p}{4}$ and $q_2 = \frac{p}{2}$. This means that the 2nd component does not influence the classifier (the phenomenon also can be observed from Figure 8). We can reduce the computation complexity by dropping the 2nd component in the feature space. ð

With various distributions of data, there are sometimes some noises hiding behind the data. The feature selection is one way to filter out the noises. After selecting features in the input space by the L-J method, some noises still remain in the feature space. Our approach can get rid of the noise in the feature space. So the two-stage feature selection can filter out both the noises in the input space and in the feature space.

5. Conclusions

With a set of training data, we construct an SVM for the input space. When we use the SVM to test a new set of input vectors (i.e. test data), we observe the classification accuracy. When the accuracy is over a threshold, we keep on using the SVM to classify the sustained coming test data. There are two major problems in the development of SVM. One is the suitable choice of the kernel functions. The other is how to reduce the computation complexity caused by larger amount of features. The former not yet has a good conclusion. The later is often solved by feature selection.

In this paper, we propose a new feature selection to rank features in the

feature space. The L-J method offers an approach to select features in the input space. In the non-linear case, we combine our result with the L-J method to form a two-stage feature selection. We first select features in the input space by the L-J method. After mapping into the feature space, we then use our result to drop the unnecessary components in the feature space. By the two-stage feature selection, we can reduce the risk of over-fitting and drop noises both in the input space and the feature space. We supplement the weakness of the L-J method.

There is one weakness on our result. When the dimension of the feature space is infinite, we cannot really get w. A. Webb proposed iterative feature selection schemes [12]. Further improvements may be probably achieved by combining our approach with the iterative feature selection schemes.

Reference

[1] K.J.C. Burges., "A tutorial on Support Vector Machines for Pattern Recognition", Data mining and knowledge discovery, 1995.

[2] Chi-hau Chen, "Statistical Pattern Recognition", Spartan Hayden, 1973.

[3] C. Cortes and V.N. Vapnik. "Support Vector Networks", Machine Learning, 20:273-297, 1995.

[4] N. Cristianini and J. Shawe-Taylor, "An Introduction to Support Vector

Machines and other kernel-based learning methods", Cambridge, 2000.

[5] R. Fletcher. "Practical Methods of Optimization", John Wiley & Sons, 2nd edition, 1987.

[6] L. Hermes and J.M. Buhmann, "Feature Selection for Support Vector Machines", Proc. of the IEEE, 2000, p.712-p.715.

[7] A. Madevska and D. Nikolic, "A new approach of modifying SVM outputs", Proc. of the IEEE, 2000, p.395-p.398.

[8] W.S. Meisel, "Computer-Oriented Approaches to Pattern Recognition", New York and London, 1972.

[9] V.N. Vapnik., "Estimation of dependencies based on empirical Data", Springer New York, 1979.

[10] V.N. Vapnik., "Statistical Learning Theory", John Wiley & Sons, 2nd edition, 1998.

[11] V.N. Vapnik., "The Nature of Statistical Learning Theory", Springer New York, 2000.

[12] A. Webb, "Statistical Pattern Recognition", Amold, 1999.