e-PG Pathshala

Subject : Computer Science

Paper: Machine Learning

Module: Support vector machines i

Module No: CS/ML/17

Quadrant I – e-text

Welcome to the e-PG Pathshala Lecture Series on Machine Learning. In this module we will be discussing another popular machine learning technique – Support Vector Machinesin detail.

Learning Objectives:

The learning objectives of this module are as follows:

- To understand the principles of Linear and Nonlinear SVM
- To know the interpretation of Kernel Functions
- To acquire knowledge about solving optimization problem

17.1 Introduction

Support Vector Machines or SVMs as they are popularly called were proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in recent times. It is basically an algorithm for learning linear classifiers. Support Vector Machines is motivated by the idea of maximizing margins. Though basically SVMs are essentially linear, efficient extension to non-linear SVMs is possible through the use of kernels.

Support vector machines are based on three main ideas. We first need to define an optimal hyperplane in a computationally efficient way: that is we need to maximize margin. We then extend the above definition for non-linearly separable problems by having a penalty term for misclassifications. We also map data to high dimensional space where it is easier to classify with linear decision surfaces, in other words reformulate the problem so that data is mapped implicitly to this space.

17.2 Linear Separators

Binary classification can be viewed as the task of separating items into two classes in feature space (Figure 17.1). The line that separates the two classes

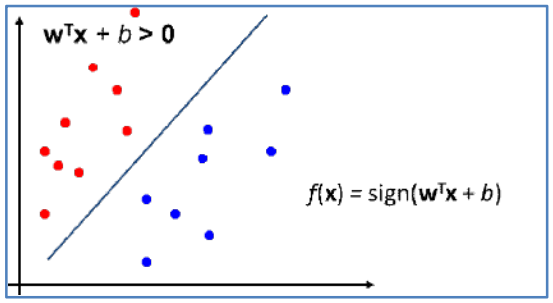

Figure 17.1 Binary Classification

is specified by the equation w^Tx+b where all items that fall in the space $w^Tx+b>0$ are represented by red dots and all items that fall in the space $w^Tx+b<0$ are represented by blue dots.

17.2.1 Linear Classifier

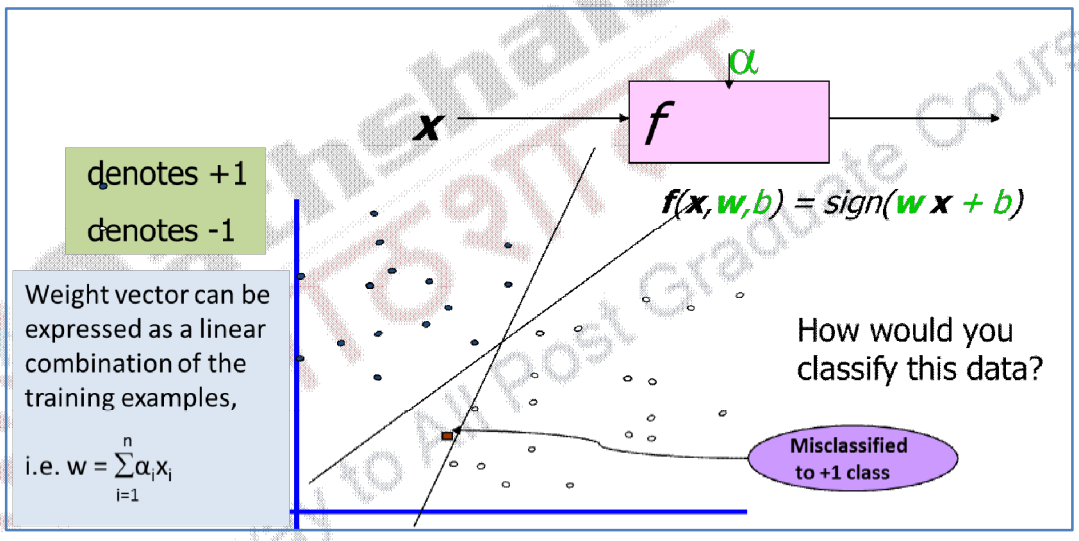

Figure 17.2 Linear Separator

Figure 17.2 shows that more than one linear separator can be defined that separates two sets of examples. In this figure given input points x, they are processed by the function f(x,w,b), whose sign value decides the class of the item. Here the weight vector w can be defined as a weighted linear combination of the training examples. Different linear separators are defined based on different values of weight (w) or slope and different values of b or intercept (Figure 17.3).Lots of possible solutions for finding the separating plane exist but we need to find the optimal one according to some criterion of expected goodness.

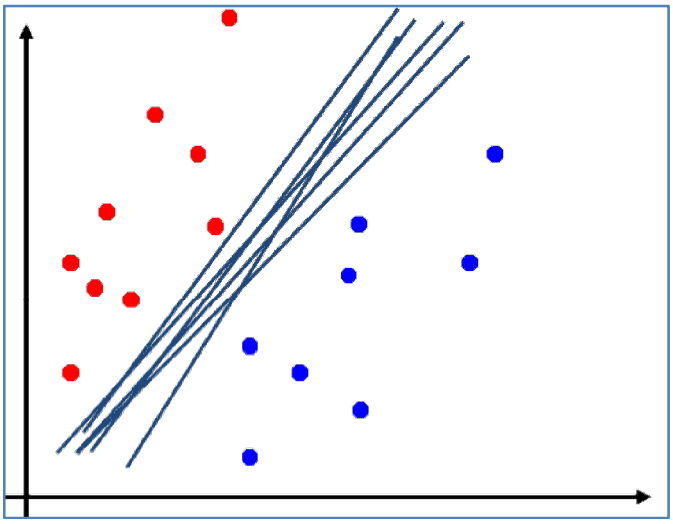

Figure 17.3 Different Linear Separators

17.3 Classification Margin

Now let us define the classification margin. The distance from example x_i to the separator is given by

$$r = \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$$

The marginρis shown in Figure 17.4 and is the distance between two of the closest data points or examples on either side of the separator.

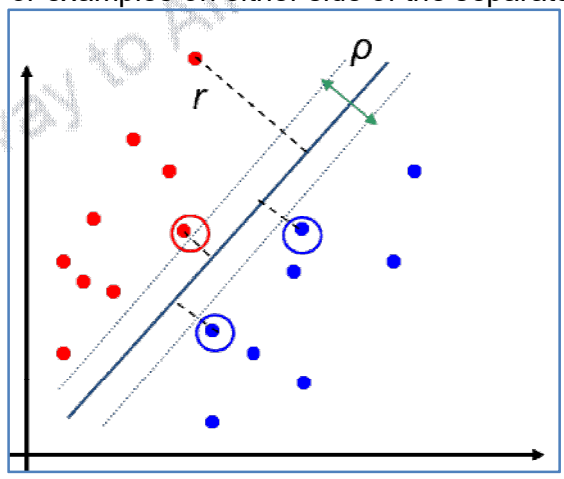

Figure 17.4 Classification Margin

Examples closest to the hyperplane are *support vectors*. In other words, the marginρ of the separator is the distance between support vectors. Figure 17.5

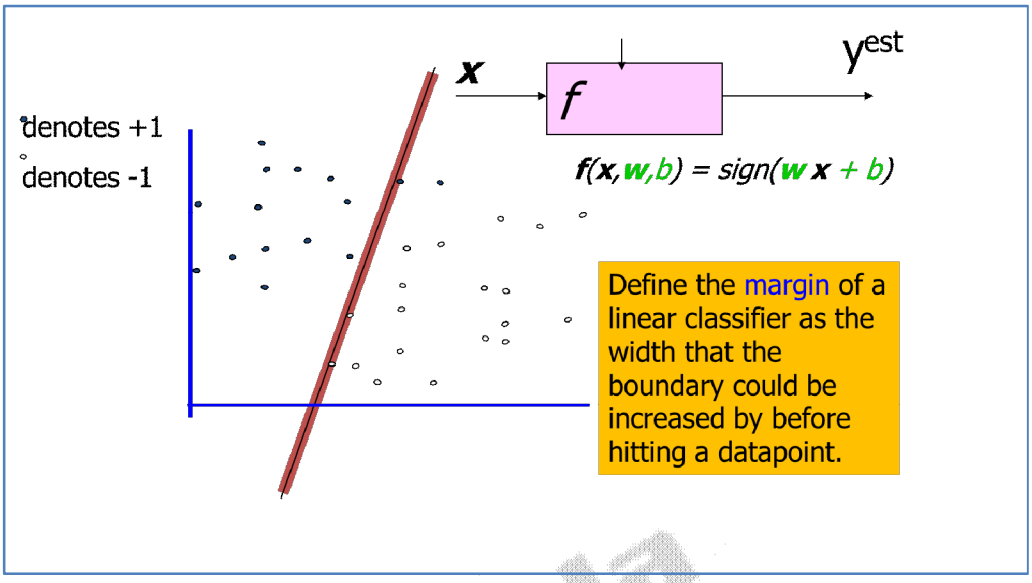

Figure 17.5 Margin of SVM

shows the margin of the linear classifier which is defined as the width that the boundary or hyperplane between the two classes can be increased before hitting a data point.

17.3.1 Maximum Margin Classification

Maximizing the margin is good according to intuition since this allows larger separation between the two sets of examples. Moreover maximizing the margin is also according to PAC(Probably Approximately Correct) theory. PAC theory states that any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be wrong. This maximization also implies that only support vectors matter; other training examples are ignorable.

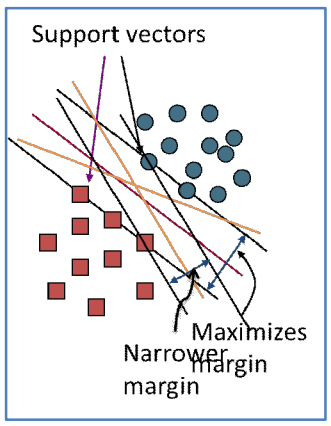

Figure 17.6 Maximum Margin

Support Vectors are those datapoints that the margin pushes up against. The maximum margin linear classifier is as the name suggests the linear classifier with the maximum margin.Support Vector Machine (SVM) finds an

optimalsolution that maximizes the distance between the hyperplane and the "difficult points" close to decision boundary. The intuition is that if there are no points near the decision surface, then there are no very uncertain classification decisions.

SVMs maximize the *margin* around the separating hyperplane.The decision function that is the function that separates the examples can now be fully specified by a subset of training samples, *the support vectors*.This is the simplest kind of SVM (Called an LSVM) (Figure 17.7). Solving SVMs is a *quadratic programming* problem. SVMs are considered as one of the most successful current text classification methods.Maximizing the margin is good according to intuition and PAC (Probably Approximately Correct)theory. This implies that only support vectors are important; other training examples are ignorable and this implication works very well empirically.The maximum margin linear classifier is the linear classifier with the, um, maximum margin. This is the simplest kind of SVM (Called an LSVM).

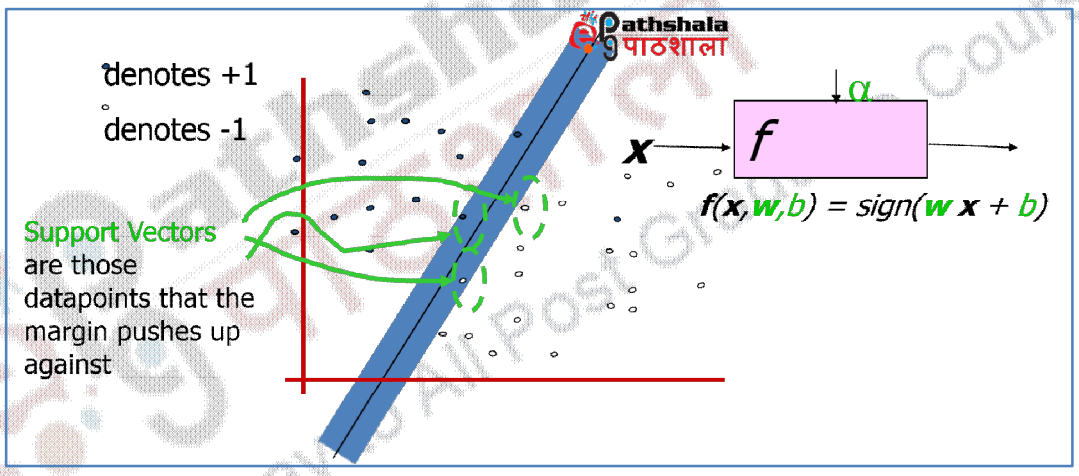

Figure 17.7 Linear SVM

17.3.2 Theoretical Justification for Maximum Margins

Vapnik has proved that t*he class of optimal linear separators has VC* (Vapnik–Chervonenkis) dimension *h bounded by*

$$h \leq \min\left\{\left\lceil \frac{D^2}{\rho^2} \right\rceil, m_0\right\} + 1$$

*where ρ is the margin, D is the diameter of the smallest sphere that can enclose all of the training examples, and $m_0$is the dimensionality.*Intuitively, this implies that regardless of dimensionality $m_0$we can minimize the VC dimension by maximizing the margin *ρ.*Thus, complexity of the classifier is kept small regardless of dimensionality.

17.3.3Linear SVMs Mathematically

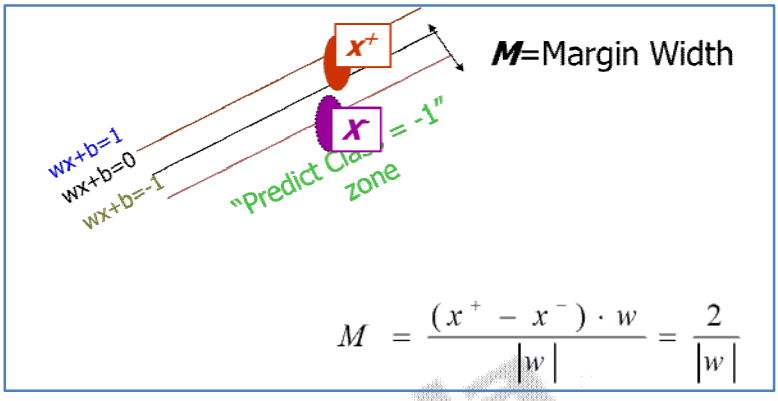

Figure 17.8 Mathematical Concept of Linear SVM

Let us assume that the functional margin of each data item is at least 1, then the following two constraints is true for a training set $\{(\mathbf{x}_i, y_i)\}$

$$\mathbf{w}^T\mathbf{x}_i + b \geq 1 \quad \text{if } y_i = 1$$
$$\mathbf{w}^T\mathbf{x}_i + b \leq -1 \quad \text{if } y_i = -1$$

For support vectors, the inequality becomes an equality
Then, since each example's distance from the hyperplane is

$$r = y\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

The margin is:

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Separated by a hyper-plane with margin ρ we can rewrite the equation for each training example (x_i, y_i) as:

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b \le -\rho/2 \quad & \text{if } y_i = -1 \\ \mathbf{w}^T\mathbf{x}_i + b \ge \rho/2 \quad & \text{if } y_i = 1 \end{aligned} \quad \Leftrightarrow \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge \rho/2$$

For every support vector x_s the above inequality is an equality. After rescaling w and *b* by $\rho/2$, we obtain distance between each x_s and the hyper plane as

$$r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

The margin can be expressed through (rescaled) w as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Then we can formulate the *quadratic optimization problem:*

Find **w** and *b* such that

$\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized; and for all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w}^T\mathbf{x_i} + b \ge 1$ if $y_i = 1$; $\mathbf{w}^T\mathbf{x_i} + b \le -1$ if $y_i = -1$

Using a better formulation (min **w**= max 1/**w**), this can be reformulated as a minimization problem as follows:

Find **w** and *b* such that

$\Phi(\mathbf{w}) = \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge 1$

Solving the Optimization Problem

Find **w** and b such that $\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge 1$

This is now optimizing a *quadratic* function subject to *linear* constraints. Quadratic optimization problems are a well-known class of mathematical programming problem, and many algorithms exist for solving them. The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\alpha_1 \ldots \alpha_N$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and

(1) $\Sigma\alpha_i y_i = 0$

(2) $\alpha_i \geq 0$ for all α_i

17.4 Soft Margin Classification

What if the training set is not linearly separable or we have a dataset with noise (Figure 17.9)? So far we have assumed a hard margin that is we enforce the condition that all the data points be classified correctly and there is no training error. However in some cases the training set can be noisy. The answer is to use very powerful kernels which we will discuss later. However in the case of some noisy data or if the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples. In other words allow some errors and let some points be moved to where they belong, however at a cost (Figure 17.10). We will still try to minimize training set errors, and to place hyperplane "far" from each class (large margin). We add slack variables ξ_i to allow misclassification of difficult or noisy examples, resulting in a soft margin. These types of SVMs are called C-SVMs.

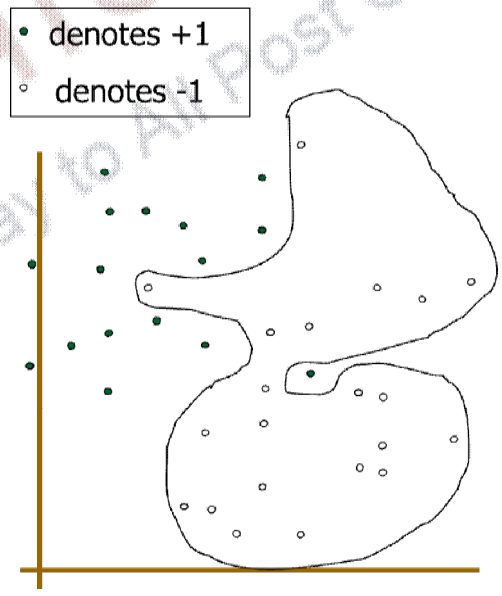

Figure 17.9 Dataset with Noise

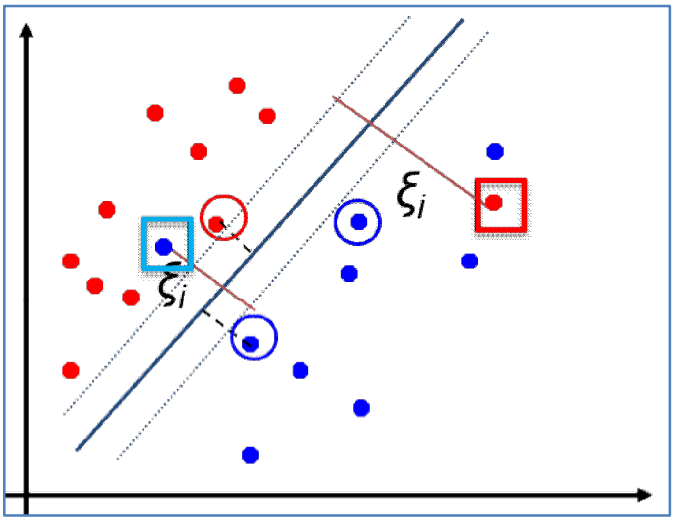

Figure 17.10 Soft Margin Classification Mathematically

The old formulation already discussed is:

Find **w** and b such that
$\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

The modified formulation incorporates slack variables which are basically used to avoid overfitting. Now our quadratic optimization criterion will be to minimize:

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

The new formulation incorporating slack variables:

Find **w** and b such that
$\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$

Parameter *C* can be viewed as a way to control overfitting. In other words ***C*** is a tradeoff parameter between error and margin; and is usually chosen by the user. A large C means a higher penalty to errors.

The Soft Margin Classification solution is similar to the dual problem of separable case except that we need additional Lagrange multipliers for slack variables:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$ (2) $0 \leq \alpha_i \leq C$ for all α_i

Again, x_i with non-zero α_i will be support vectors.

$$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i$$
$$b = y_k(1 - \xi_k) - \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x}_k \quad \text{for any } k \text{ s.t. } \alpha_k > 0$$

Again, we don't need to compute w explicitly for classification:

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$$

17.5 Linear SVMs: Summary

The classifier is a *separating hyperplane.*Most "important" training points are support vectors which define the hyperplane.Quadratic optimization algorithms can identify training points x_i with non-zero Lagrangian multipliers α_i. Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$$

17.6 Non-Linear SVMs

Datasets that are linearly separable with some noise (Figure 17.11) was tackled using soft margins.

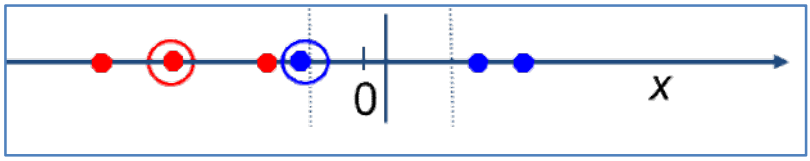

Figure 17.11 Dataset with Noise

But what are we going to do if the dataset is just too hard (Figure 17.12)?

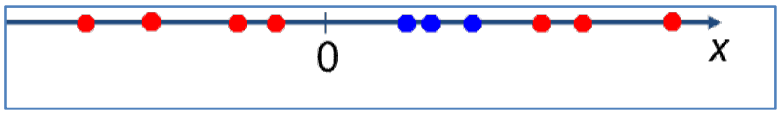

Figure 17.12 Dataset not Linearly Seperable

To tackle these types of problems we map data to a higher-dimensional space.General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable. We map our points with a mapping function $\phi(x)$ to a space of sufficiently high dimension so that they are linearly separable by a hyper-plane. Given a dataset that is not linearly separable in $\mathbb{R}^N$ may be linearly separable in a higher-dimensional space $\mathbb{R}^M$ (where M > N). Thus, if we have a transformation that transforms the dataset to a higher-dimensional space such that it becomes linearly separable in the higher dimensional space, then we can train a linear SVM to find a decision boundary that separates the classes in $\mathbb{R}^M$. Projecting the decision boundary found back to the original space will yield a nonlinear decision boundary. Figure 17.13 shows the non-linear one dimensional dataset transformed to two dimensions. The figure shows that in two dimension the dataset is linearly separable.

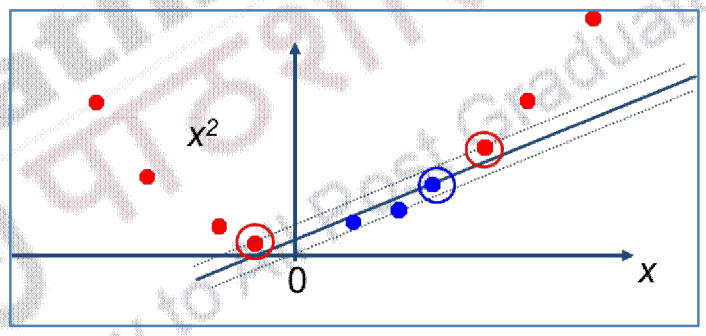

Figure 17.13 Dataset of Figure 17.12 Transformed to Two Dimensions

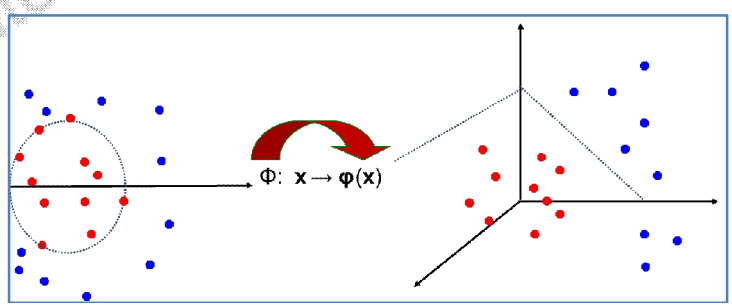

Figure 17.14 Transformations from 2D to 3D Space

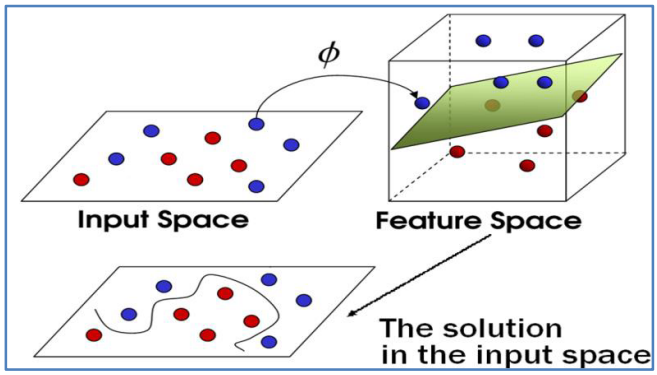

Figure 17.15 Input and Feature Spaces

Figure 17.14 shows the dataset in 2D that is not linearly separable being transformed into 3D space where it becomes linearly separable. The input space is the space where the points $\mathbf{x}_i$ are located while the feature space is the space of $\phi(\mathbf{x}_i)$ after transformation (Figure 17.15).

17.7 The "Kernel Trick"

In general working in high dimensional feature space will be computationally expensive since while constructing the maximal margin hyperplane, we need to evaluate highdimensional inner products. Now the "Kernel Trick" comes to the rescue.For many mappings from a low-dimensional space to a high-dimensional space, there is a simple operation on two vectors in the low-dimensional space that can be used to compute the scalar product of their two images in the high-dimensional space.

$$K(x^a, x^b) = \phi(x^a).\phi(x^b)$$

where we let the kernel do the work, while the ϕ components calculate the scalar product as shown in Figure 17.16

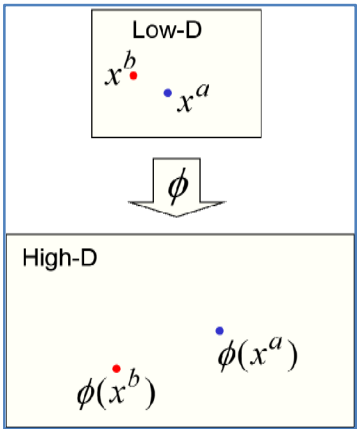

Figure 17.16 Kernel Mapping

The linear classifier relies on inner product between vectors

$$K(x_i,x_j)=x_i^T x_j$$

Every data point is mapped into high-dimensional space as

$$\Phi: \ x \rightarrow \varphi(x),$$
$$K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$$

A *kernel function* is a function that is equivalent to an inner product in some feature space.
Therefore kernel function is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Now we only need to compute $K(x_i,x_j)$ and there is no need to perform computations in high dimensional space explicitly. This is what is called the Kernel Trick.

If we look again at the optimization problem:

$$\text{maximize } L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i^t) \phi(x_j) \ \ s.t \ \alpha_i \geq 0 \ \ \sum_{i=1}^{n} y_i \alpha_i = 0$$

The decision function can be stated as:

$$f(\phi(x)) = sign(w^t \phi(x) + b) = sign(\sum_{i=1}^{n} \alpha_i y_i \phi(x_i^t) \phi(x) + b)$$

There is no need to know this mapping explicitly nor do we need to know the dimension of the new space, because we only use the dot product of feature vectors in both the training and test set.

Example:
2-dimensional vectors $x=[x_1 \ \ x_2]$; let the Kernel function $K(x_i,x_j)=(1 + x_i^T x_j)^2$,
$K(x_i,x_j) = (1 + x_i^T x_j)^2$,

$= 1+ x_{i1}^2 x_{j1}^2 + 2 x_{i1}x_{j1}x_{i2}x_{j2}+ x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_j$

$= [1 \ \ x_{i1}^2 \ \sqrt{2} x_{i1}x_{i2} \ \ x_{i2}^2 \ \sqrt{2}x_{i1} \ \sqrt{2}x_{i2}]^T [1 \ \ x_{j1}^2 \ \sqrt{2} x_{j1}x_{j2} \ \ x_{j2}^2 \ \sqrt{2}x_{j1} \ \sqrt{2}x_{j2}]$

$= \phi(x_i)^T \phi(x_j)$, where $\phi(x) = [1 \ \ x_1^2 \ \sqrt{2} x_1 x_2 \ \ x_2^2 \ \ \sqrt{2}x_1 \ \sqrt{2}x_2]$

Thus, a kernel function *implicitly* maps data to a high-dimensional space. In this example it maps the one dimensional data to a 2D dimensional space.

17.7.1 Kernel Functions

What Functions are Kernels?

For some functions $K(x_i,x_j)$ actually checking that that

$K(x_i,x_j)= \varphi(x_i)^T\varphi(x_j)$ can be cumbersome.

Here the Mercer's theorem helps us. The Mercer's theorem states that *every semi-positive definite symmetric function is a kernel.*A function $K(x_i,x_j)$ is a kernel (there exists a $\phi(x)$ such that $K(x_i,x_j)= \phi(x_i)^T\phi(x_j)$

17.7.2 Examples of Kernel Functions

Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^T\mathbf{x}_j$
Mapping Φ: $\mathbf{x}\rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself

Polynomial of power *p*: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^T\mathbf{x}_j)^p$

Mapping Φ: $\mathbf{x}\rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions

Gaussian (radial-basis function):

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

Mapping Φ: $\mathbf{x}\rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function(a Gaussian)*

Higher-dimensional space still has *intrinsic* dimensionality *d* but linear separators in it correspond to *non-linear* separators in original space.

17.7.3 Important Kernel Issues

One important question is which Kernel to use? This is still a open research question. It is one of the weaknesses of SVM. The next important question is how to verify that rising to higher dimension using a specific kernel will map the data to a space in which they are linearly separable. For most of the kernel functionswe do not know the corresponding mapping function $\phi(x)$and hence we do not know which dimension we transformed to. So even though rising to higher dimension increases the likelihood that they will be separable we can't guarantee that.

17.8 Non-linear SVMs Mathematically

Similar to the non-linear case, the dual problem formulation is:

Find $\alpha_1 \ldots \alpha_n$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

and the solution for the problem is:

$$f(\mathbf{x}) = \Sigma\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b$$

Optimization techniques for finding α_i's for non-linear SVMs remain the same.

17.9 Properties of SVM

- Since only support vectors are used to specify the separating hyperplane there is sparseness of solution space when dealing with large data sets as only support vectors are used to specify the separating hyper-plane..
- Due to the same reason it has the ability to handle large feature spaces- in other words the complexity does not depend on the dimensionality of the feature space.
- Overfitting can be controlled by soft margin approach
- There is a good math property: a simple convex optimization problem which is guaranteed to converge to a single global solution
- Feature Selection is automatic and is decided by support vectors.

In general, SVMs express learning as a mathematical program taking advantage of the rich theory in optimization. It is able to use the kernel trick to map indirectly to extremely high dimensional spaces. SVMs are extremely successful, robust, efficient, and versatile while there are good theoretical indications as to why they generalize well. Many tools are available that support SVMs.

17.10SVM applications

- SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been extended to a number of tasks such as regression, principal component analysis, etc.
- Most popular optimization algorithms for SVMs use *decomposition* to hill-climb over a subset of α_i's at a time, e.g. SMO
- Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner. A method called

Grid Search is a simple and systematic technique resembling exhaustive search that can be used for parameter estimation. We take exponentially increasing values in a particular range and find the set with minimum cross-validation error.

Summary

- Explained the Maximum Margin Classification
- Described Linear and Non-linear SVMs
- Discussed about solving the optimization Problem
- Explained the Kernel Functions
- Listed SVM applications