Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 / MoSIG M1 Second Semester 2011/2012

Lesson 21 2 May 2012

Kernel Methods and Support Vector Machines

Contents

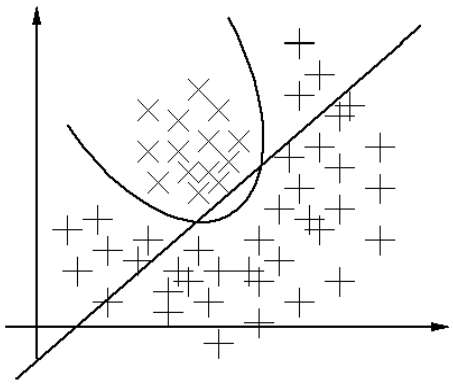
Sources Bibliographiques :
"Neural Networks for Pattern Recognition", C. M. Bishop, Oxford Univ. Press, 1995.

Kernel Methods

Linear methods are very well suited for use with very high dimensional feature space provided that the patterns can be separated by a plane.

Kernel Methods transform a non-linear function into a linear function in a much higher dimensional space. Thus they enable linear discriminant methods to be applied to a large class of problems where the data are dispersed in a non-linear manner.

Kernel Methods provide an elegant solution for clustering and classifying patterns in complex non-linear data by mapping the data into a higher dimensional space where the data can be separated by a linear method.



Kernels make it possible to
1) Solve the computational problems of high dimensional spaces
2) Be extended to infinite dimensional spaces
3) Be extended to non-numerical and symbolic data!

Linear methods are very well suited for use with very high dimensional feature space. We can map a quadratic decision space into a linear space by adding additional dimensions. For example, a quadratic surface in a D dimensional space can be transformed into a linear surface in a D(D+1)/2 space by from the D dimensional space to a space P > D using a kernel function, K().

For example a D=2 space $\vec{X}$ can be projected on to D=5 space $K(\vec{X}) = (x_1, x_2, x_1^2, x_1 x_2, x_2^2)$. A linear function of $K(\vec{X})$ is quadratic in $\vec{X}$.

Kernel Functions

Formally, a Kernel is a function that returns the inner product of a function applied to two arguments.

$$f(\vec{X}) = \sum_{m=1}^{M} a_m \, y_m \left\langle \phi(\vec{X}_m), \phi(\vec{X}) \right\rangle + b$$

The key notion of a kernel method is an inner product space.

$$\left\langle \vec{x}, \vec{z} \right\rangle = \sum_{d=1}^{D} x_d z_d$$

A common kernel function as a quadratic mapping of a feature space, $\phi(x)$.

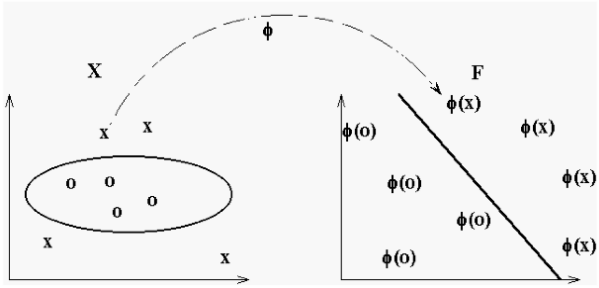
$$k(\vec{X}_1, \vec{X}_2) = \vec{\phi}(\vec{X}_1)^T \vec{\phi}(\vec{X}_2)$$

Note that the kernel is a symmetric function of its arguments, so that

$$k(\vec{X}_1, \vec{X}_2) = k(\vec{X}_1, \vec{X}_2)$$

There are a large variety of possible kernel functions that can be used, depending on the problem.

example: Polynomial Kernel:



For example, a quadratic kernel in a space where D=2 is

$$k(\vec{x}, \vec{z}) = (\vec{x}^T \vec{z})^2 = (x_1 z_1 + x_2 z_2)^2 = (x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2)$$

This can be expressed as an inner product space where

$$\phi(\vec{x}) = x_1^2 + \sqrt{2}x_1x_2 + x_2^2$$

giving:

$$k(\vec{x},\vec{z}) = \vec{\phi}(\vec{x})^T \vec{\phi}(\vec{z})$$

In order to be "valid", a kernel must correspond to a scalar product of some feature space. That is, there must exist a space such that

$$k(\vec{X}_1,\vec{X}_2) = \vec{\phi}(\vec{X}_1)^T \vec{\phi}(\vec{X}_2) = \sum_{n=1}^{N} \phi_n(\vec{X}_1)\cdot\phi_n(\vec{X}_2)$$

A necessary, and sufficient condition that a Kernel function be "valid" is that the Gram matrix be positive and semi-definite for all choices of $\{\vec{X}_m\}$

The Gram Matrix (or Grammian) for $\vec{x}$ is $\vec{x}^T\vec{x}$

The Gram Matrix projects a linear vector $\vec{x}$ onto a quadratic surface $\vec{x}^T\vec{x}$

Gaussian Kernel

The Gaussian exponential is very often used as a kernel function.
In this case:

$$k(\vec{x},\vec{x}') = e^{-\frac{\|\vec{x}-\vec{x}'\|}{2\sigma^2}}$$

This is often called the Gaussian Kernel. It is NOT a probability density.
We can see that it is a valid kernel because:

$$\|\vec{x}-\vec{x}'\|^2 = \vec{x}^T\vec{x} - 2\vec{x}^T\vec{x}' + \vec{x}'^T\vec{x}'$$

Among other properties, the feature vector can have infinite dimensionality.

Kernel function for Symbolic Data

Kernel functions can be defined over graphs, sets, strings and text!

Consider for example, a non-vectoral space composed of a Set of words S.
Consider two subsets of S: $A_1 \subset S$ and $A_2 \subset S$

The can compute a kernel function of $A_1$and A_2 using the interesection

$$k(\vec{x},\vec{x}') = 2^{|A_1 \cap A_2|}$$

where |A| denotes the number of elements (the cardinality) of a set.

Probabilistic generative models tend to be more robust with missing data and data of variable length, while Probabilistic Discriminative models tend to give better performance and lower cost.

We can combine generative and discriminative models using a kernel.

Given a generative model *p(X)* we can define a kernel as:

$$k(\vec{x},\vec{x}') = p(\vec{x})p(\vec{x}')$$

This is clearly a valid kernel because it is a 1-D inner product. Intuitively, it says that two feature vectors, x, are similar if they both have high probability.

We can extend this with conditional probabilities to

$$k(\vec{x},\vec{x}') = \sum_{n=1}^{N} p(\vec{x} \mid n)p(\vec{x}' \mid n)p(n)$$

Two vectors, $\vec{x},\vec{x}'$ will give large values for the kernel, and hence be seen as similar, if they have significant probability for the same components.

Kernel functions enable application of linear classifiers to non-linear problems.

Support Vector Machines

A significant limitation for linear leaning methods is that the kernel function must be evaluated for every training point during learning.

An alternative is to use a learning algorithm with sparse support - that is a uses only a small number of points to learn the separation boundary.

A Support Vector Machine (SVM) is such an algorithm.

SVM's are popular for problems of classification, regression and novelty detection. The solution of the model parameters corresponds to a convex optimisation problem. Any local solution is a global solution.

We will use the two class problem, K=2, to illustrate the principle. Multi-class solutions are possible.

Our linear model is for the decision surface is

$$g(\vec{X}) = \vec{w}^T \phi(\vec{X}) + b$$

Where $\phi(\vec{X})$ is a feature space transformation that maps a hyper-plane in F dimensions into a non-linear decision surfaces in D dimensions. F >> D

Training data is a set of M training samples $\{\vec{X}_m\}$ and their indicator variable, $\{y_m\}$. For a 2 Class problem, y_m is -1 or +1.

A new, observed point (not in the training data) will be classified using the function $sign(g(\vec{X}))$, so that a classification of a training sample is correct if

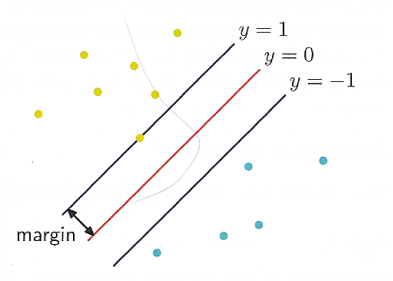
$$y_m g(\vec{X}_m) > 0$$

Hard-Margin SVMs - Separable Training Data

Let us assume, for the moment, that the data is linearly separable.
There exists a hyper-plane $g(\vec{X}) = \vec{w}^T \phi(\vec{X}) + b$ such that $y_m g(\vec{X}_m) > 0$ for all m.

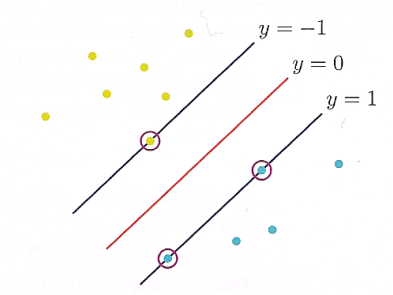
Generally there will exist many solutions for separable data.
For a support Vector Machine, the decision boundary is chosen to maximize the margin, γ.

Recall that the margin, γ is the minimum distance of any sample from the hyper-plane



Bishop p 327 (fig 7.1)
(Bishop uses *y(X)* for the discriminant function and t_m for the indicator variable.)

What we are going to do is design the decision boundary to that it has aN equal distance from a small number of support points.



Bishop p 327 (fig 7.1)
(Bishop uses *y(X)* for the discriminant function and t_m for the indicator variable.)

The distance for a point from the hyper-plane is $\frac{|g(\vec{X})|}{\|\vec{w}\|}$

since we are only interested in points where $y_m g(\vec{X}_m) > 0$

The distance for the point $\vec{X}_m$ to the decision surface is:

$$\frac{y_m g(\vec{X}_m)}{\|\vec{w}\|} = \frac{y_m(\vec{w}^T \phi(\vec{X}_m) + b)}{\|\vec{w}\|}$$

We will seek to maximize the margin by solving

$$\arg\max_{w,b}\left\{\frac{1}{\|\vec{w}\|}\min_m\left\{y_m(\vec{w}^T\phi(\vec{X}_m)+b)\right\}\right\}$$

The factor $\frac{1}{\|\vec{w}\|}$ can be removed from the optimization because $\|\vec{w}\|$ does not depend on m.

Direct solution would be very difficult. We will convert this to an equivalent problem.

Note that rescaling the problem changes nothing. Thus we will scale the equation such for the sample that is closest to the decision surface (smallest margin):

$$y_m(\vec{w}^T\phi(\vec{X}_m)+b)=1 \quad \text{that is:} \quad y_m g(\vec{X}_m)=1$$

For all other sample points:

$$y_m(\vec{w}^T\phi(\vec{X}_m)+b)\geq 1$$

This is known as the Canonical Representation for the decision hyperplane.

The training sample where $y_m(\vec{w}^T\phi(\vec{X}_m)+b)=1$ are said to be the "active" constraint. All other training samples are "inactive".

By definition there is always at least one active constraint.

When the margin is maximized, there will be two active constraints.

Thus the optimization problem is to maximize $\arg\min_{w,b}\left\{\frac{1}{2}\|\vec{w}\|^2\right\}$ subject to the active constraints.

The factor of ½ is a convenience for later analysis.

To solve this problem, we will use Lagrange Multipliers, $a_m \geq 0$, with one multiplier for each constraint. This gives a Lagrangian function:

$$L(\vec{w},b,\vec{a}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{m=1}^{M} a_m \left\{ y_m(\vec{w}^T \phi(\vec{X}_m) + b) - 1 \right\}$$

Setting the derivatives to zero, we obtain:

$$\frac{\partial L}{\partial w} = 0 \Rightarrow \quad \vec{w} = \sum_{m=1}^{M} a_m y_m \phi(\vec{X}_m)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \quad \sum_{m=1}^{M} a_m y_m = 0$$

Eliminating $\vec{w},b$ from $L(\vec{w},b,\vec{a})$ we obtain :

$$L(\boldsymbol{a}) = \sum_{m=1}^{M} a_m - \frac{1}{2} \sum_{m=1}^{M} \sum_{n=1}^{M} a_m a_n y_m y_n k(\vec{X}_m, \vec{X}_n)$$

with constraints:

$a_m \geq 0$ for m=1, ..., M

$$\sum_{m-1}^{M} a_m y_m = 0$$

where the kernel function is : $k(\vec{X}_1, \vec{X}_2) = \vec{\phi}(\vec{X}_1)^T \vec{\phi}(\vec{X}_2)$

The solution takes the form of a quadratic programming problem in D variables (the Kernel space). This would normally take $O(D^3)$ computations.

In going to the dual formulation, we have converted this to a dual problem over M data points, requiring $O(M^3)$ computations.
This can appear to be a problem, but the solution only depends on a small number of points!

To classify a new observed point, we evaluate:

$$g(\vec{X}) = \sum_{m=1}^{M} a_m y_m k(\vec{X}, \vec{X}_m) + b$$

The solution to optimization problems of this form satisfy the "Karush-Kuhn-Tucker" condition, requiring:

$$a_m \geq 0$$
$$y_m g(\vec{X}_m) - 1 \geq 0$$
$$a_m \left\{ y_m g(\vec{X}_m) - 1 \right\} \geq 0$$

For every data point in the training samples, $\left\{ \vec{X}_m \right\}$, either

$$a_m = 0 \quad \text{or} \quad y_m g(\vec{X}_m) = 1$$

Any point for which $a_m = 0$ does not contribute to $g(\vec{X}) = \sum_{m=1}^{M} a_m y_m k(\vec{X}, \vec{X}_m) + b$ and thus is not used! (is not active) .

The remaining points, for which $a_m \neq 0$ are called the "Support vectors".
These points lie on the margin at $t_m y(\vec{X}_m) = 1$ of the maximum margin hyperplane.
Once the model is trained, all other points can be discarded!

Let us define the support vectors as the set S.

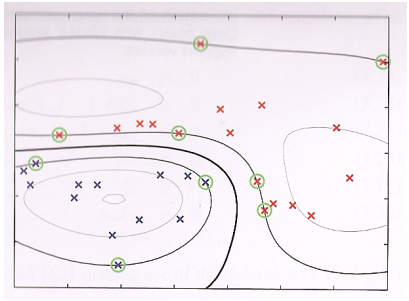
Now that we have solved for S and **a**, we can solve for b:

we note that : $y_m \left(\sum_{n \in S} a_n y_n k(\vec{X}_m, \vec{X}_n) + b \right) = 1$

averaging over all support vectors in S gives:

$$b = \frac{1}{N_S} \sum_{m \in S} \left(y_m - \sum_{n \in S} a_n y_n k(\vec{X}_m, \vec{X}_n) \right)$$

This can be expressed as minimization of an error function, $E_\infty(z)$ such that the error function is zero if $z \geq 0$ and ∞ otherwise.



From Bishop p 331.

Soft Margin SVM's - Non-separable training data.

So far we have assumed that the data are linearly separable in $\phi(\vec{X})$.
For many problems some training data may overlap.

The problem is that the error function goes to ∞ for any point on the wrong side of the decision surface. This is called a "hard margin" SVM.

We will relax this by adding a "slack" variable, S_m for each training sample:

$$S_m \geq 1$$

We will define

$S_m = 0$ for samples on the correct side of the margin, and
$S_m = \left| y_m - g(\vec{X}_m) \right|$ for other samples.

For a sample inside the margin, but on the correct side of the decision surface:

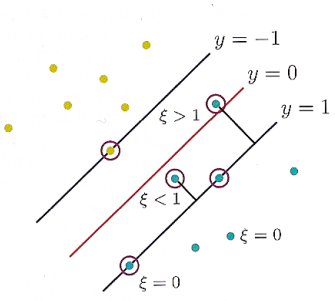
$$0 < S_m \leq 1$$

For a sample on the decision surface:

$$S_m = 1$$

For a sample on the wrong side of the decision surface:

$$S_m > 1$$



Soft margin SVM: Bishop p 332 (note use of ξ_m in place S_m)

This is sometimes called a soft margin. To softly penalize points on the wrong side, we minimize :

$$C\sum_{m=1}^{M} S_m + \frac{1}{2}\|\vec{w}\|^2$$

where C > 0 controls the tradeoff between slack variables and the margin.

because any misclassified point $S_m > 1$, the upper bound on the number of misclassified points is is $\sum_{m=1}^{M} S_m$.

C is an inverse factor. (note that C=∞) is the earlier SVM with hard margins.

To solve for the SVM we write the Lagrangian:

$$L(\vec{w},b,\vec{a}) = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{m=1}^{M} S_m - \sum_{m=1}^{M} a_m\left\{y_m g(\vec{X}_m) - 1 + S_m\right\} - \sum_{m=1}^{M} \mu_m S_m$$

The KKT conditions are

$$a_m \geq 0$$
$$y_m g(\vec{X}_m) - 1 + S_m \geq 0$$
$$a_m\left\{y_m g(\vec{X}_m) - 1 + S_m\right\} \geq 0$$
$$\mu_m \geq 0$$
$$S_m \geq 1$$
$$\mu_m S_m = 0$$

Solving the derivatives of $L(\vec{w},b,\vec{a})$ for zero gives

$$\frac{\partial L}{\partial w} = 0 \Rightarrow \quad \vec{w} = \sum_{m=1}^{M} a_m y_m \phi(\vec{X}_m)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \quad \sum_{m=1}^{M} a_m t_m = 0$$

$$\frac{\partial L}{\partial S} = 0 \Rightarrow \quad a_m = C - \mu_m$$

using these to eliminate w, b and $\{S_m\}$ from *L(w, b, a)* we obtain

$$L(\boldsymbol{a}) = \sum_{m=1}^{M} a_m - \frac{1}{2}\sum_{m=1}^{M}\sum_{n=1}^{M} a_m a_n y_m y_n k(\vec{X}_m,\vec{X}_n)$$

This appears to be the same as before, except that the constraints are different.

$$0 \le a_m \le C$$

$$\sum_{m=1}^{M} a_m y_m = 0$$

(referred to as a "box" constraint). Solution is a quadratic programming problem, with complexity $O(M^3)$. However, as before, a large subset of training samples have $a_m = 0$, and thus do not contribute to the optimization.

For the remaining points $y_m g(\vec{X}_m) = 1 - S_m$

For samples ON the margin $a_m < C$ hence $\mu_m > 0$ requiring that $S_m = 0$

For samples INSIDE the margin: $a_m = C$ and $S_m \le 1$ if correctly classified and $S_m > 1$ if misclassified.

as before to solve for b we note that :

$$y_m \left(\sum_{n \in S} a_n y_n k(\vec{X}_m, \vec{X}_n) + b \right) = 1$$

averaging over all support vectors in S gives:

$$b = \frac{1}{N_S} \sum_{m \in T} \left(y_m - \sum_{n \in S} a_n y_n k(\vec{X}_m, \vec{X}_n) \right)$$

where $\mathcal{T}$ denotes the set of support vectors such that $0 < a_m < C$.