Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 / MoSIG M1 Second Semester 2012/2013

Lesson 20 2 May 2013

Kernel Methods and Support Vector Machines

<u>Contents</u>

Sources Bibliographiques :
"Neural Networks for Pattern Recognition", C. M. Bishop, Oxford Univ. Press, 1995.
"A Computational Biology Example using Support Vector Machines", Suzy Fei, 2009 (on line).

Kernel Functions

Linear discriminant functions can be very efficient classifiers, provided that the class features can be separated by a linear decision surface.

The linear discriminant function is: $g(\vec{X}) = \vec{W}^T\vec{X} + b$

The decision rule is IF $\vec{W}^T\vec{X} + b > 0$ THEN $E \in C_1$ else $E \notin C_1$

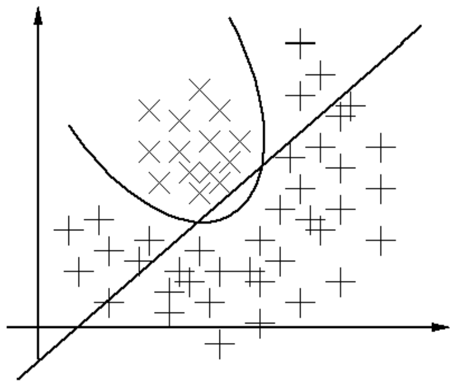
For many domains, it is easier to separate the classes with a linear function if you can transform your feature data into a space with a higher number of dimensions.

One way to do this is to transform the features with a “kernel” function.

Instead of a decision surface: $g(\vec{X}) = \vec{W}^T\vec{X} + b$

We use a decision surface $g(\vec{X}) = \vec{W}^T\varphi(\vec{X}) + b$

This can be used to construct non-linear decision surfaces for our data.



The trick is to learn with the non-linear mapping, but to use the resulting discriminant without actually computing the mapping.

Formally, a "kernel function" is any function that satisfies the Mercer condition. Mercer’s condition requires that for a finite set of observations S, one can select a measure $\mu(T) = |T|$ for all $T \in S$ such that

$$\sum_{i=1}^{M}\sum_{j=1}^{M} k(\vec{X}_i, \vec{X}_j)c_i c_j \geq 0$$

This condition is satisfied by inner products. Inner products are of the form

$$\vec{W}^T\vec{X} = \langle \vec{W}, \vec{X} \rangle = \sum_{d=1}^{D} w_d x_d$$

Thus $k(\vec{x},\vec{z}) = \langle \varphi(\vec{x}), \varphi(\vec{z}) \rangle$ is a valid kernel function.

The Mercer function can be satisfied by other functions.
For example, the Gaussian function:

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

is a valid Kernel.

We can learn the discriminant in an inner product space $\vec{W}^T\varphi(\vec{X}) = \langle \vec{W}, \varphi(\vec{X}) \rangle$ where the vector W will be learned from the mapped values of our training data.

This will give us a discriminant function of the form:

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

We will see that we can learn in the kernel space, and then recognize without actually having to compute the kernel function!

Kernels can be extended to infinite dimensional spaces and even to non-numerical and symbolic data!

Quadratic as a Kernel Function

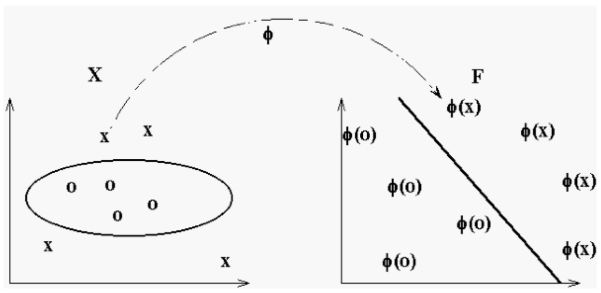
A common kernel is the quadratic kernel

$$k(\vec{x},\vec{z}) = (\vec{z}^T\vec{x} + c)^2 = \vec{\phi}(\vec{z})^T\vec{\phi}(\vec{x})$$

For D=2, this gives $\vec{\varphi}(\vec{X}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1c \\ \sqrt{2}x_2c \end{pmatrix}$

This Kernel maps a 2-D feature space to a 5 D kernel space.

This can be used to map a plane to a hyperbolic surface.



Gaussian Kernel

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

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

This satisfies Mercer's condition because the exponent is separable

$$\|\vec{x}-\vec{z}\|^2 = \vec{x}^T\vec{x} - 2\vec{x}^T\vec{z} + \vec{z}^T\vec{z}$$

Intuitively, you can see this as placing a Gaussian function multiplied by the indicator variable (y_m = +/- 1) at each training sample, and then summing the functions.

The zero-crossings in the sum of Gaussianss defines the decision surface. Depending on σ, this can provide a good fit or an over fit to the data.

If σ is large compared to the distance between the classes, this can give an overly flat discriminant surface.

If σ is small compared to the distance between classes, this will overfit the samples.

A good choice for σ will be comparable to the distance between the closest members of the two classes.

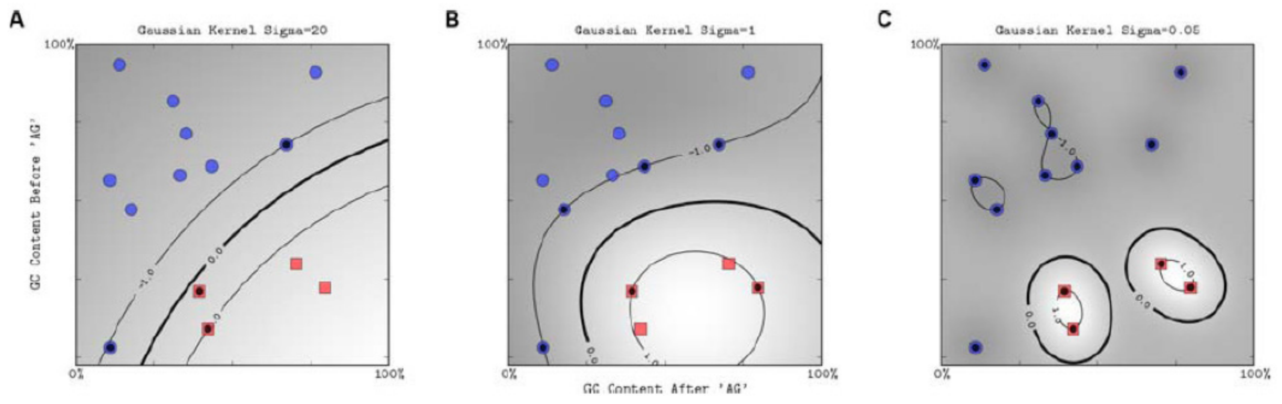


Figure from lecture "A Computational Biology Example using Support Vector Machines" Suzy Fei, 2009 (on line).

Among other properties, the feature vector can have infinite number of dimensions.

Kernel functions for Symbolic Data

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

Consider for example, a non-vector space composed of a set of words S.
Consider two subsets of S: $A \subset S$ and $B \subset S$

We can define a kernel function of A and B using the intersection operation.

$$k(A,B) = 2^{|A \cap B|}$$

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

Kernels for Bayesian Reasoning

We can define a Kernel for Baysian reasoning for evidence accumulation.

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

$$k(\vec{X},\vec{Z}) = p(\vec{X})p(\vec{Z})$$

This is clearly a valid kernel because it is a 1-D inner product. Intuitively, it says that two feature vectors, $\vec{X}$ and $\vec{Z}$, are similar if they both have high probability.

We can extend this for conditional probabilities to

$$k(\vec{Z},\vec{X}) = \sum_{n=1}^{N} p(\vec{Z})p(\vec{X} \mid A)p(A)$$

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

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.

Training data is a set of M training samples $\{\vec{X}_m\}$ and their indicator variable, $\{y_m\}$. As in our last lecture, 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

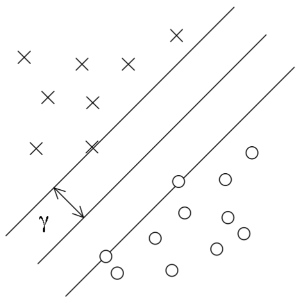
$$y_m g(\vec{X}_m) > 0$$

Hard-Margin SVMs - Separable Training Data

We assume that the two classes can be separated by a linear function.
That is, 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.

The margin, γ, is the minimum distance of any sample from the hyper-plane
For a support Vector Machine, we will determine the decision surface that maximizes the margin, γ.



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.

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

$y_m g(\vec{X}_m) > 0$ for all training points

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

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

For a decision surface, (W, b), the support vectors are the samples from the training set, $\{\vec{X}_m\}$ that minimize the margin, γ_m,

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

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, but the problem can be converted 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, $\{\vec{X}_m\}$, either

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

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.

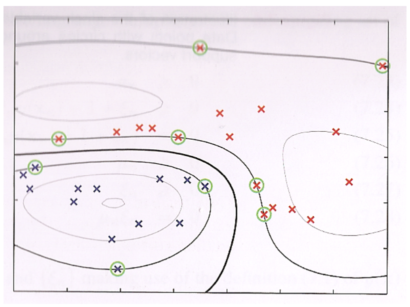
Now that we have solved for S and **a**, we can solve for b:

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
$S_m = |y_m - g(\vec{X}_m)|$ for other samples.

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

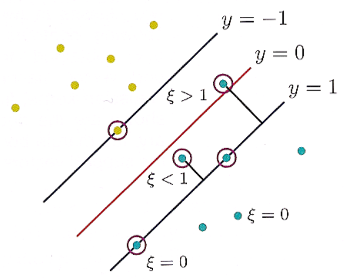
$$0 < S_m \leq 1$$

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

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

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.

$0 \leq a_m \leq C$

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

(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 $S_m \leq 1$ 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 $0 < a_m < C$.