Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 / MoSIG M1 Second Semester 2014/2015

Lesson 19 29 April 2015

Support Vector Machines using Kernels

Contents

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

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

Kernel Functions

Linear discriminant functions can provide very efficient 2-class classifiers, provided that the class features can be separated by a linear decision surface.

For many domains, it is possible to find a "kernel" function,

$$k(\vec{Z},\vec{X}) = \langle \varphi(\vec{Z}), \varphi(\vec{X}) \rangle$$

that will transform the data into a space where the two classes are separate.

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

We will use a decision surface $g(\vec{X}) = k(\vec{Z},\vec{X}) + b$

We learn the discriminant in an inner product space $k(\vec{Z},\vec{X}) = \langle \varphi(\vec{Z}), \varphi(\vec{X}) \rangle$ where the vector $\vec{Z}$ is learned from the training data.

that it $g(\vec{X}) = \langle \varphi(\vec{Z}), \varphi(\vec{X}) \rangle + b = \varphi(\vec{Z})^T \varphi(\vec{X}) + b$

This will give us a discriminant function of the form:

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

where $\vec{W} = \varphi(\vec{Z}) = \sum_{n=1}^{N} a_n y_n \phi(\vec{X}_n)$ is learned from the transformed training data.

Support Vector Machines with Kernels

Let us assume that a training data composed of N training samples $\{\vec{X}_n\}$ and their indicator variable, $\{y_n\}$, where , y_n is -1 or +1.

We will seek a linear decision surface $g(\vec{X}) = \vec{W}^T\phi(\vec{X}) + b$ such that the the training data fall into two separable classes. That is

$$\forall n: \; y_n(\vec{W}^T\phi(\vec{X}) + b) > 0$$

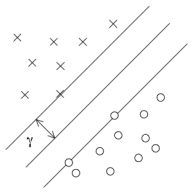
If we assume that the data is separable, then for all training samples:

$$y_n g(\vec{X}_n) > 0$$

For any training sample $\vec{X}_n$ the perpendicular distance to the decision surface is:

$$d_n = \frac{y_n g(\vec{X}_n)}{\|\vec{W}\|} = \frac{y_n(\vec{W}^T\phi(\vec{X}_n) + b)}{\|\vec{W}\|}$$

The margin is the smallest distance from the decision surface:



$$\gamma = \min\{y_n(\vec{W}^T\phi(\vec{X}_n) + b)\}$$

For a decision surface, ($\vec{W}$, b), the support vectors are the subset S of the training sample, $S \subset \{\vec{X}_n\}$ that minimize the margin, γ_n,

$$\gamma = \min_n\{\gamma_n\} = \min_n\left\{\frac{1}{\|\vec{W}\|} y_n(\vec{W}^T\vec{X}_n + b)\right\}$$

We will seek to maximize the margin by finding the S training samples that maximize:

$$\arg\max_{\vec{W},b}\left\{\frac{1}{\|\vec{W}\|}\min_n\left\{y_n(\vec{W}^T\phi(\vec{X}_n)+b)\right\}\right\}$$

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

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):

$$y_n(\vec{W}^T\phi(\vec{X}_n)+b)=1 \quad \text{that is:} \quad y_n g(\vec{X}_n)=1$$

For all other sample points:

$$y_n(\vec{W}^T\phi(\vec{X}_n)+b)>1$$

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

The training sample where $y_n(\vec{w}^T\phi(\vec{X}_n)+b)=1$ 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 D+1 active constraints.

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

The factor of ½ is a convenience for later analysis.

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

$$L(\vec{W},b,\vec{a})=\frac{1}{2}\|\vec{W}\|^2-\sum_{n=1}^{N}a_n\left\{y_n(\vec{W}^T\phi(\vec{X}_n)+b)-1\right\}$$

Setting the derivatives to zero, we obtain:

$$\frac{\partial L}{\partial \vec{W}} = 0 \Rightarrow \vec{W} = \sum_{n=1}^{N} a_n y_n \phi(\vec{X}_n)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n y_n = 0$$

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

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

with constraints:

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

$$\sum_{n=1}^{N} a_n y_n = 0$$

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_k variables (the dimension of the Kernel space). This would normally take $O(D_k^3)$ computations.

In going to the dual formulation, we have converted this to a dual problem over N data points, requiring $O(N^3)$ computations.

This can appear to be a problem, but the solution only depends on a small number of points $N_s << N$.

To classify a new observed point, we evaluate:

$$g(\vec{X}) = \sum_{n=1}^{n} a_n y_n k(\vec{X}_n, \vec{X}) + b$$

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

$$a_n \geq 0$$
$$y_n g(\vec{X}_n) - 1 \geq 0$$
$$a_n \{ y_n g(\vec{X}_n) - 1 \} \geq 0$$

For every observation in the training set, $\{\vec{X}_n\}$, either

$$a_n = 0 \quad \text{or} \quad y_n g(\vec{X}_n) = 1$$

Any point for which $a_n = 0$ does not contribute to $g(\vec{X}) = \sum_{n=1}^{N} a_n y_n k(\vec{X}_n, \vec{X}) + b$

and thus is not used! (is not active) .

The remaining N_s samples for which $a_n \neq 0$ are called the "Support vectors". These points lie on the margin at $y_n g(\vec{X}_n) = 1$ 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.

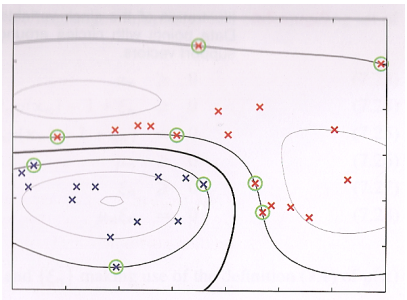
Now that we have solved for S and **a**, we can solve for b:

we note that for any active training sample n in S

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

averaging over all support vectors in S gives:

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



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, z_n for each training sample.

$z_n \geq 1$

We will define

$z_n = 0$ for training samples on the correct side of the margin, and

$z_n = |y_n - g(\vec{X}_n)|$ for other training samples.

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

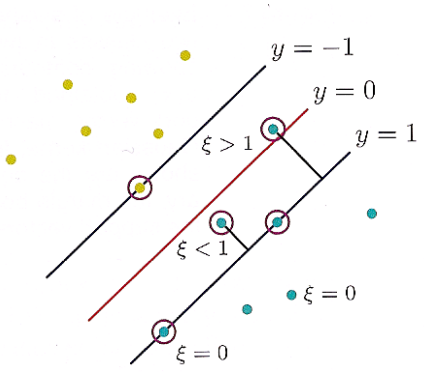
$0 < z_n \leq 1$

For a sample on the decision surface:

$z_n = 1$

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

$z_n > 1$



Soft margin SVM: Bishop p 332 (note use of ξ_n in place z_n)

This is called a soft margin SVM.

To softly penalize points on the wrong side, we minimize :

$$C\sum_{n=1}^{N} z_n + \frac{1}{2}\|\vec{w}\|^2$$

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

because any misclassified point $z_n > 1$, the upper bound on the number of misclassified points is $\sum_{n=1}^{N} z_n$.

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

To solve for the SVM we write the Lagrangian:

$$L(\vec{W},b,z,\vec{a},\mu) = \frac{1}{2}\|\vec{W}\|^2 + C\sum_{n=1}^{N} z_n - \sum_{n=1}^{N} a_n\{y_n g(\vec{X}_n) - 1 + z_n\} - \sum_{n=1}^{N} \mu_n z_n$$

where $\{a_n \geq 0\}$ and $\{ \mu_n \geq 0\}$ are the Lagrange multipliers.

The KKT conditions are

$$a_n \geq 0$$
$$y_n g(\vec{X}_n) - 1 + z_n \geq 0$$
$$a_n\{y_n g(\vec{X}_n) - 1 + z_n\} \geq 0$$
$$\mu_n \geq 0$$
$$z_n \geq 1$$
$$\mu_n z_n = 0$$

We optimize for $\vec{w}$, b, and $\{z_n\}$, using $g(\vec{X}) = \vec{W}^T\phi(\vec{X}) + b$

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

$$\frac{\partial L}{\partial w} = 0 \Rightarrow \vec{W} = \sum_{n=1}^{N} a_n y_n \phi(\vec{X}_n)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n y_m = 0$$

$$\frac{\partial L}{\partial z_n} = 0 \Rightarrow a_n = C - \mu_n$$

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

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

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

$0 \le a_n \le C$ and $\sum_{n=1}^{N} a_n y_n = 0$

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

For the remaining points $y_n g(\vec{X}_n) = 1 - S_n$

For samples ON the margin $a_n < C$ hence $\mu_n > 0$ requiring that $S_n = 0$

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

as before to solve for b we note that :

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

averaging over all support vectors in S gives:

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

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