Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 / MoSIG M1 Second Semester 2014/2015

Lesson 18 24 April 2015

Support Vector Machines and Kernel Functions

Contents

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).

Support Vector Machines

Support Vector Machines (SVM), also known as maximum margin classifiers are popular for problems of classification, regression and novelty detection. The solution of the model parameters corresponds to a convex optimization problem. SVM's use a minimal subset of the training data (the "support vectors) to define the "best" decision surface between two classes. We will use the two class problem, K=2, to illustrate the principle. Multi-class solutions are possible.

The simplest case, the hard margin SVM, require that the training data be completely separated by at least one hyper-plane. This is generally achieved by using a Kernel to map the features into a high dimensional space were the two classes are separable.

To illustrate the principle, we will first examine a simple linear SVM where the data are separable. We will then generalize with Kernels and with soft margins.

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

Hard-Margin SVMs - simple linear classifier.

The simplest case is a simple linear classifier trained from separable data.

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

Where the decision rule is: IF $\vec{W}^T \vec{X} + b > 0$ THEN C_1 else C_2

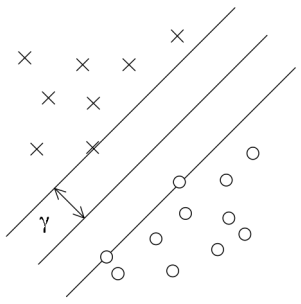
For a hard margin SVM we assume that the two classes are separable for all of the training data:

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

We will use a subset S of the training samples, $\{\vec{X}_s\} \subset \{\vec{X}_n\}$ composed of N_s training samples to define the "best" decision surface $g(\vec{X}) = \vec{W}^T \vec{X} + b$. The number of support vectors depends on the number of features ($N_s \geq D+1$). The N_s selected training samples are called the support vectors. For example, in a 2D feature space we need only 3 training samples to serve as support vectors.

Thus to define the classifier we will look for the subset if S training samples that maximizes the separation between the two classes.

The separation is defined using the margin: , γ.



Assume that we have normalized the coefficients of the hyperplane such at

$$\|\vec{W}\| = 1$$

Then the distance of any sample point $\vec{X}_n$ from the hyper-plane, $\vec{W}$.

$$d = \; y_n(\vec{W}^T \vec{X}_n + b)$$

The margin is the minimum distance

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

A D dimensional decision surface is defined by at least D points. However, we will seek a surface such that that the closest samples are as far away as possible. This is equivalent to a pair of parallel surfaces a distance γ from the decision surface. Thus we will need $N_s \geq D+1$ points to define the pair of parallel surfaces.

Because a D dimensional decision surface is defined by $N_s=D+1$ points there will be at least N_s training sample on margin, such that $d_s=\gamma$.

We will use these samples as support vectors. For all other training samples:

$$d_m \geq \gamma$$

To find the sample points that serve as support vectors, we can arbitrarily define the margin as $\gamma = 1$ and then renormalize $\|\vec{W}\|$ once the support vectors have been discovered. We will look for two separate hyper-planes that "bound" the decision surface, such that for points on these surfaces:

$$\vec{W}^T\vec{X} + b = 1 \text{ and } \vec{W}^T\vec{X} + b = -1$$

The distance between these two planes is $\frac{2}{\|\vec{W}\|}$

We will add the constraint that for all training samples that are support vectors

$$\vec{W}^T\vec{X}_m + b \geq 1$$

while for all other samples:

$$\vec{W}^T\vec{X}_m + b \leq -1$$

This can be written as: $y_n(\vec{W}^T\vec{X}_n + b) \geq 1$

This gives we have an optimization algorithm that looks for minimizes $\|\vec{W}\|$ subject

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

If we note that minimizing $\|\vec{W}\|$ is equivalent minimizing $\frac{1}{2}\|W\|^2$, we can set this up as a quadratic optimization problem, and use Lagrange Multipliers.

So our problem is to find $\arg\min_{W,b}\{\|\vec{W}\|^2\}$ such that $y_n(\vec{W}^T\vec{X}_n + b) \geq 1$

We look for a surface, ($\vec{W}$, b) such that

$$\arg\min_{\vec{W},b}\left\{\frac{1}{2}\|\vec{W}\|^2\right\} \qquad \text{subject to} \qquad y_n(\vec{W}^T\vec{X}_n + b) \geq 1$$

by searching for :

$$\arg\min_{\vec{W},b}\{\max_{\alpha_n}\{\frac{1}{2}\|W\|^2 - \sum_{n=1}^{N}\alpha_n[y_n(\vec{W}^T\vec{X}_n + b) - 1]\}\}$$

for a subset of $N_s \geq D+1$ samples, $\alpha_n \geq 0$. These are the samples on the margins.

For all other samples where ($\alpha_n < 0$) we set $\alpha_n = 0$.

The normal of the decision surface is then:

$$\vec{W} = \sum_{n=1}^{N}\alpha_n y_n \vec{X}_n$$

and the offset can be found by solving for:

$$b = \frac{1}{N_S}\sum_{n\in S}\vec{W}^T\vec{X}_n - y_n$$

The solution can be generalized for use with non-linear decision surfaces using kernels.

Kernel Functions

A Kernel function transforms the training data so that a non-linear decision surface is transformed to a linear equation in a higher number of dimensions.

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

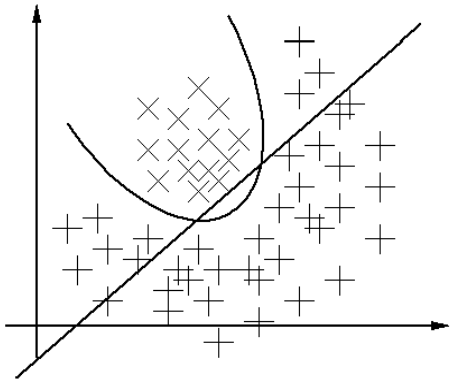
For many domains, it is easier to separate the classes with a linear function if you can transform your feature data into a space with a higher number of dimensions.

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

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



Formally, a "kernel function" is any function that satisfies the Mercer condition.
A function, *k(x, y)*, satisfies Mercer's condition if for all square, integrable functions *f(x)*,

$$\iint k(x,y) f(x) f(y) dx dy \geq 0$$

This condition is satisfied by inner products.

Inner products are of the form $\langle \vec{W}, \vec{X} \rangle = \sum_{d=1}^{D} w_d x_d = \vec{W}^T \vec{X}$

Thus $k(\vec{W},\vec{X}) = \langle \vec{W},\vec{X} \rangle$ is a valid kernel function.

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

and in this case $\vec{W} = \varphi(\vec{Z})$
We can learn the discriminant in an inner product space $k(\vec{Z},\vec{X}) = \langle \varphi(\vec{Z}),\varphi(\vec{X}) \rangle$
where the vector $\vec{z}$ will be learned from the training data.

This will give us a discriminant function of the form:

$$g(\vec{X}) = \sum_{n=1}^{N} a_n y_n \langle \phi(\vec{X}_n),\phi(\vec{X}) \rangle + b = \vec{W}^T \vec{X} + b$$

where $\vec{W} = \sum_{n=1}^{N} a_n y_n \phi(\vec{X}_n)$

The Mercer function can be satisfied by many other functions.
Popular kernel functions include:

- Polynomial Kernels
- Radial Basis Functions
- Fisher Kernels

Kernel functions provide an implicit feature space. We will see that we can learn in the kernel space, and then recognize without explicitly computing the position in this implicit space!

This will allow us to use Kernels for infinite dimensional spaces as well as non-numerical and symbolic data!

Polynomial Kernel Functions

The Polynomial kernel is defined as

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

where n is the "order" of the kernel, and c is a constant that allows to trade off the influence of the higher order and lower order terms.

Second order or quadratic kernels are a popular form of Polynomial kernel, widely used in Speech Recognition. Higher order kernels tend to "overfit" the training data and thus do not generalize well.

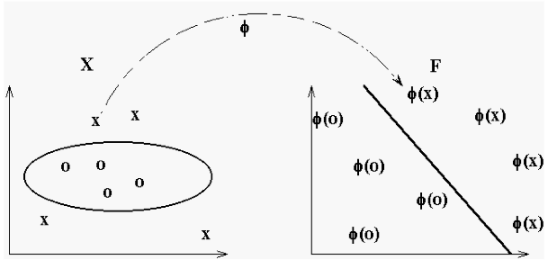
The quadratic kernel can be expressed as:

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

For example for a feature vector with two components d=2,

$$\vec{\varphi}(\vec{X}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1c \\ \sqrt{2}x_2c \end{pmatrix} \text{ for } \vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

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.



In general when for a d dimensional feature vector:

$$\vec{\varphi}(\vec{X}) = \begin{pmatrix} x_1^2 \\ \vdots \\ x_d^2 \\ \sqrt{2}x_1x_2 \\ \vdots \\ \sqrt{2}x_dx_{d-1} \\ \sqrt{2}x_1c \\ \vdots \\ \sqrt{2}x_dc \\ c \end{pmatrix}$$

Radial Basis Function (RBF)

Also known as a Gaussian Kernel, Radial Basis Function (RBF) kernels are often used in Computer Vision. The RBF Kernel function is:

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

Where $\|\vec{x}-\vec{z}\|$ is the Euclidean distance, and $\frac{\|\vec{x}-\vec{z}\|^2}{2\sigma^2}$ is the squared Mahalanobis distance (Squared Euclidean distance normalized by the variance, σ^2).

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 parameter, σ acts as a smoothing parameter that determines the influence of each of the points, $\vec{z}$, derived from the training data.

The zero-crossings in the sum of Gaussians 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.

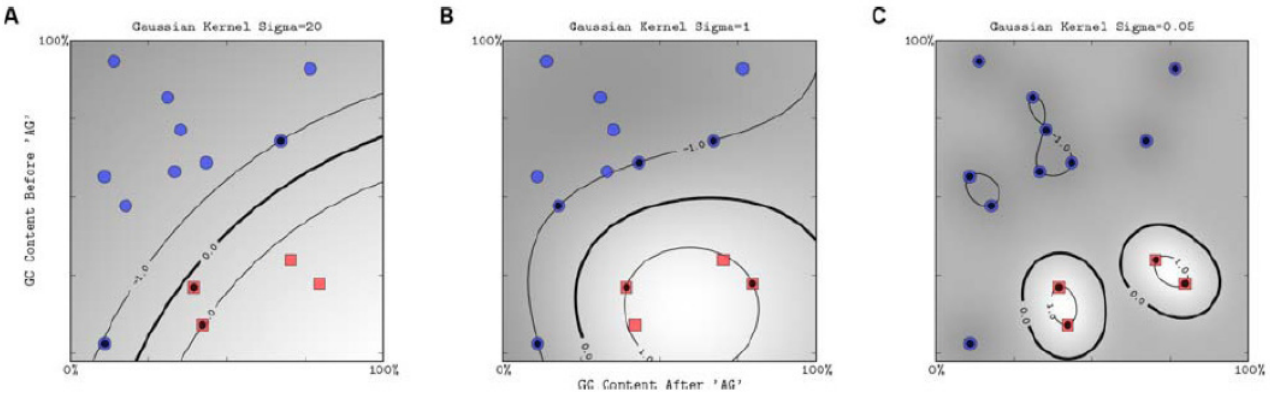


Figure from lecture "A Computational Biology Example using Support Vector Machines" Suzy Fei, 2009 (on line).

Among other properties, the feature vector can have infinite number of dimensions. RBF's are widely used in Computer Vision, as well as biology.

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 {W}.
We can select a subset of discriminant words {S} ⊂ {W}

Now given a set of words (a probe), {A} ⊂ {W}

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

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

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

Kernels for Bayesian Reasoning

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

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

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

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

We can extend this for conditional probabilities to

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

where $\vec{Z}$ is a hidden latent variable.

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