Intelligent Systems: Reasoning and Recognition

James L. Crowley

EMSIMAG 2 Winter Semester 2021
Lesson 6 26 February 2021

Support Vector Machines and Kernel Methods

Outline

Notation

x_d	A feature. An observed or measured value.
$\vec{X}$	A vector of D features.
D	The number of dimensions for the vector $\vec{X}$
$\{\vec{X}_m\}$	Training samples for learning.
$\{y_m\}$	The indicator variable for each training sample,
	$y_m = +1$ for examples of the target pattern (class 1)
	$y_m = -1$ for all other examples (class 2)
M	The number of training samples.

$\vec{w}^T\vec{X}+b=0$	Set of points, $\vec{X}$, on a hyperplane that divides a space in D dimensions
$\vec{w}^T\vec{X}+b>0$	Points, $\vec{X}$, that are on the positive side of the hyperplane
$\vec{w}^T\vec{X}+b<0$	Points, $\vec{X}$, that are on the negative side of the hyperplane

$\vec{w}=\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{pmatrix}$ The coefficients for a hyperplane.

b The perpendicular distance of the hyperplane from the origin.

$\text{sgn}(z)$ The sign function $\text{sgn}(z)=\begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$

$\gamma = \min\left\{ y_m \cdot \left(\vec{w}^T\vec{X}_m + b \right) \right\}$ margin for a classifier:

$\langle \cdot, \cdot \rangle$ inner product operator $\left\langle \vec{X}, \vec{Z} \right\rangle = \vec{X}^T\vec{Z} = \sum_{d=1}^{D} x_d z_d$

The Margin for a Linear Classifier

Linear Discriminant Functions

The equation $\vec{w}^T\vec{X}+b=0$ is a hyper-plane in a D-dimensional space,

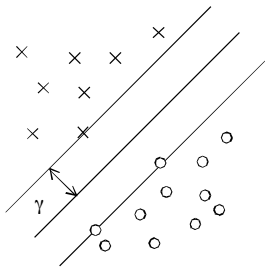
$\vec{w}=\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{pmatrix}$ is the normal to the hyperplane and b is proportional to the perpendicular distance to the origin.

For any point, $\vec{X}$ if $\vec{w}^T\vec{X}+b>0$ then the point is above the hyperplane.
if $\vec{w}^T\vec{X}+b=0$ then the point is on the hyperplane.
if $\vec{w}^T\vec{X}+b<0$ then the point is below the hyperplane.

We can use the sign to build a linear classifier: IF $\vec{w}^T\vec{X}+b\geq 0$ THEN C_1 else C_2
The hyperplane is called a "decision surface".
$g(\vec{X})=\vec{w}^T\vec{X}+b$ is a linear discriminant function.

If the coefficients are normalized ($\|\vec{w}\|=1$) then b becomes the perpendicular distance to the origin and $d=\vec{w}^T\vec{X}+b$ is the signed distance of the point $\vec{X}$ from the plane. Normalization allows us to compare different decision surfaces based on the distance of training data from the surface.

Margin



For a 2-Class classifier, the "margin", γ, is the smallest separation between the two classes.

Assume a training set of M observations $\{\vec{X}_m\}$ $\{y_m\}$ where
$y_m=$ +1 for examples of the target class (class 1)
$y_m=$ −1 for all others (class 2)

If there exists a decision surfaces $\vec{w}$ with bias b, such that for all training data, $\{\vec{X}_m\}$ a training sample is correctly classified by:

$$y_m \cdot \left(\vec{w}^T \vec{X}_m + b\right) \geq 0$$

then the training data is said to be **separable**. Separable training data was an important criteria for many early classifiers (including perceptrons).

The margin, γ_m, for each sample, m, is the distance from the decision surface.

$$\gamma_m = y_m \cdot \left(\vec{w}^T \vec{x}_m + b\right)$$

The margin for a linear classifier is the minimum margin for the training data.

$$\gamma = \min_m \left\{ y_m \cdot \left(\vec{w}^T \vec{X}_m + b\right) \right\}$$

Support Vector Machines

Support Vector Machines (SVM) are a form of maximum margin classifier used for classification, regression and novelty detection.

A Support Vector Machine is defined by the small subset of the training samples that are closest to the decision surface. The selection of the subset of training samples that defines the model corresponds to a convex optimization problem that can require a substantial computational cost. While the computational load for training an SVM can be high, the runtime computation can be extremely light-weight, making SVMs ideal for embedded systems.

The subset of the training data that define the classifier are referred to as the “support vectors). The support vectors define an "optimal" decision surface between two classes, where the optimality criteria is defined as the largest margin.

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 very high dimensional feature vectors to classify the data.

In the following, we will use a two class problem, K=2, with separable training data to illustrate the principle. We will then generalize to non-separable training data by

showing an SVM with soft margins. We will show how a Kernel function can be used to provide a powerful form of general purpose classifier.

Hard-Margin SVMs - a Simple Binary Classifier.

Assume that we have M training samples $\{\vec{X}_m\}$ with an indicator variable, $\{y_m\}$, where , y_m is -1 or +1, with +1 representing P and -1 representing N.

The linear classifier trained from separable data is

$g(\vec{x}) = \vec{w}^T\vec{x} + b$ with the decision rule : IF $g(\vec{x}) \geq 0$ THEN P else N

This is sometimes written using the sgn() function as $\hat{y} = \text{sgn}\left(\vec{w}^T\vec{X} + b\right)$

where the sgn() function is:

$$\text{sgn}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

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

$$\exists\{\vec{w}, b\} : \forall m : \; y_m(\vec{w}^T\vec{X}_m + b) \geq 0$$

We will use a subset $\{\vec{X}_s\}$ of the training samples, $\{\vec{X}_s\} \subset \{\vec{X}_m\}$ composed of M_s training samples to define the "best" decision surface $\vec{w}^T\vec{X} + b = 0$.

The M_s selected training samples $\{\vec{X}_s\}$ are called the support vectors. The number of support vectors depends on the number of features, D.

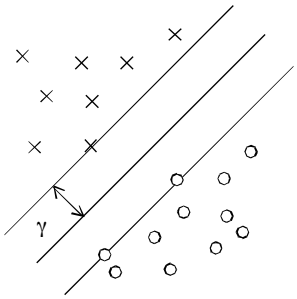
In general:

$$M_s = D+1$$

D features are required to define the decision surface and 1 additional feature is required to define the margin. For example, in a 2D feature space we need only 3 training samples to serve as support vectors.

To define the classifier we will look for the subset if M_s training samples that maximizes the margin (γ) between the two classes.

Margin: $\gamma = \min\left\{ y_m \cdot \left(\vec{w}^T \vec{x}_m + b \right) \right\}$



This is equivalent to a search for a pair of parallel surfaces a distance γ from the decision surface.

Finding the Support Vectors

Classically, the support vectors are selected using Lagrange Multipliers.

For each possible decision surface, $\vec{w}^T \vec{X} + b$, assume that we have normalized the coefficients of the hyperplane such at

$$\left\| \vec{w} \right\| = 1$$

Then it becomes possible to compare margins for different decision surfaces.

The learning algorithm must choose the subset of training samples that provide a decision surface with the largest margin. This is equivalent to a search for a pair of parallel surfaces a distance γ from the decision surface.

The distance from the decision surface for any training point, m, is

$$d_m = y_m \cdot \left(\vec{w}^T \vec{X}_m + b \right)$$

For the D+1 support vectors $\vec{X}_m \in \{\vec{X}_s\}$: $d_m = \gamma$

For all other training samples $\vec{X}_m \notin \{\vec{X}_s\}$: $d_m \geq \gamma$

The scale of the margin is determined by $\|\vec{w}\|$. To find the support vectors, we can arbitrarily define the margin as $\gamma = 1$ and then renormalize $\|\vec{w}\| = 1$ once the support vectors have been discovered.

With $\gamma = 1$, we will look for two separate hyper-planes that "bound" the decision surface, such that for points on the surfaces: $\vec{w}^T\vec{x}_m + b = 1$ and $\vec{w}^T\vec{x}_m + b = -1$

The distance between these two planes is $\frac{2}{\|\vec{w}\|}$. The inverse of the distance is $\frac{1}{2}\|\vec{w}\|$

Maximizing the distance is the same as minimizing $\frac{1}{2}\|\vec{w}\|$.

If we note that minimizing $\|\vec{w}\|$ is equivalent minimizing $\frac{1}{2}\|\vec{w}\|^2$,

We will add the constraint that for all support vectors $y_m\left(\vec{w}^T\vec{X}_m + b\right) = 1$

while for all other samples: $y_m\left(\vec{w}^T\vec{X}_m + b\right) \geq 1$

Thus the problem is to maximize $\arg\min_{\vec{w},b}\left\{\frac{1}{2}\|\vec{w}\|^2\right\}$ while minimizing the number of samples, M_s. This can be solved as a quadratic optimization problem using Lagrange Multipliers.

For a subset of $M_s \geq D+1$ samples, $a_m > 0$. These are the support vectors. For these points, we can reset mutiplier a_m to 1: $a_m = 1$.

For all other samples $a_m \leq 0$.

For these points the algorithm will set the mutiplier a_m to 0: $a_m = 0$

The Lagrangian function is: $L(\vec{w},b,\vec{a}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{m=1}^{M} a_m\left\{y_m(\vec{w}^T\vec{x}_m + b) - 1\right\}$

Setting the derivatives to zero, we obtain:

$$\frac{\partial L}{\partial \vec{w}} = 0 \Rightarrow \quad \vec{w} = \sum_{m=1}^{M} a_m y_m \vec{X}_m \qquad\qquad \frac{\partial L}{\partial b} = 0 \Rightarrow \quad b = \frac{1}{M_s}\sum_{x_s \in \{x_s\}} \vec{w}^T\vec{X}_s - y_s$$

The normal of the decision surface is then: $\vec{W} = \sum_{m=1}^{M} a_m y_m \vec{X}_m$

where most of the a_m are zero. The M_s nonzero a_m select the support vectors.

and the offset can be found by solving for: $b = \frac{1}{M_S}\sum_{m \in S} \vec{W}^T\vec{X}_m - y_m$

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

Note that only S=D+1 of the coefficients a_m are non-zero. These S non-zero coefficients select the support vectors from within the complete set of training data.

The terms a_m are used to compose a set $\{\vec{X}_S\}$ of M_s support vectors.

The bias should the same for any support vector.

Thus we can use any of the support vectors to compute the bias:

$$b = \vec{w}^T \vec{x}_S - y_s \quad \text{for any} \quad \vec{x}_S \in \{\vec{X}_S\}$$

Alternatively we can compute b as the average bias for all the support vectors.

$$b = \frac{1}{M_s} \sum_{x_m \in S} \vec{w}^T \vec{X}_m - y_m$$

This solution can be generalized for use with non-separable training data using Soft margin Support Vector Machines and non-linear decision surfaces using kernels.

Soft Margin SVMs

We can extend the SVM to the case where the training data are not separable by introducing the "hinge loss" function.

hinge loss: $L_m = \max\{0, (1 - y_m(\vec{w}^T \vec{x}_m + b))\}$

This loss is 0 if the data is separable, and non-zero if the data is non-separable.

Using the hinge loss, we can minimize

$$\left[\frac{1}{M} \sum_{m=1}^{M} \max\{0, (1 - y_m(\vec{w}^T \vec{x}_m + b))\}\right] + \lambda \|\vec{w}\|^2$$

To set this up as an optimization problem, we note that

$L_m = \max\{0, 1 - y_m(\vec{w}^T \vec{x}_m + b)\}$ is the smallest non-negative that satisfies

$$y_m(\vec{w}^T\vec{x}_m + b) \geq 1 - L_m$$

we rewrite the optimization as

$$\text{minimize } \frac{1}{M}\sum_{m=1}^{M} L_m + \lambda\|\vec{w}\|^2 \text{ subject to } y_m(\vec{w}^T\vec{x}_m + b) \geq 1 - L_m \text{ and } L_m \geq 0 \text{ for all } m.$$

This is classically solved as a quadratic optimization problem with Lagrange multipliers.

Maximize $\sum_{m=1}^{M} a_m - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} y_i a_i (\vec{x}_i^T\vec{x}_j) y_j a_j$

subject to $\sum_{m=1}^{M} a_m y_m = 0$ and $0 \leq a_m \leq \frac{1}{2M\lambda}$ for all m

The resulting Normal for the decision surface is: $\vec{w} = \sum_{m=1}^{M} a_m y_m \vec{x}_m$

We can compute b as the average bias for all the support vectors.

$$b = \frac{1}{M_s}\sum_{x_m \in S} \vec{w}^T\vec{X}_m - y_m$$

To obtain the linear classifier $g(\vec{X}) = \vec{w}^T\vec{X} + b$

However, modern approaches use Gradient Descent to solve for $\vec{w}$ and b, offering important advantages for large data sets.

Kernel Methods

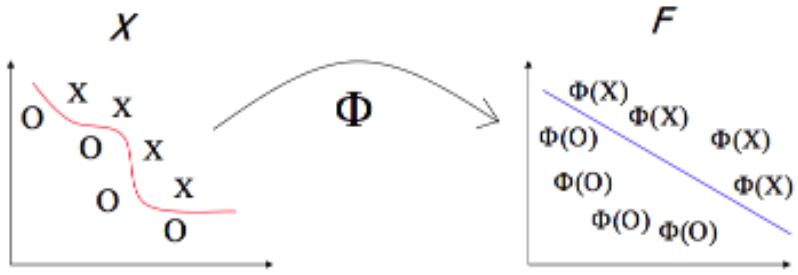
Support Vector Machines can be generalized for use with non-linear decision surfaces using kernel functions. This provides a very general and efficient classifier. The idea of for use of a kernel functions first appeared in the perceptron literature in the 1960's. However, Kernel methods were not widely used until the mid 1990's when Support Vector Machines using quadratic kernels were shown to outperform multi-layer perceptrons in both accuracy and computational cost for problems such as handwriting analysis.

Technically, a kernel function is a weighted sum. For example, the filters learned in neural networks are sometimes called kernels.

The kernel is the inner product $K(\vec{X},\vec{X}_s)=\left\langle \vec{f}(\vec{X}),\vec{f}(\vec{X}_s)\right\rangle=\sum_{n=1}^{N} f_n(\vec{X})f_n(\vec{X}_s)$

Where $f_n(\vec{X}_s)$ are the N components of the vector function $\vec{f}(\vec{X}_s)$.

Kernel functions are sometimes explained as projecting the training data into higher dimensional space where the classes are separable.



Instead of a decision function: $g(\vec{x})=\text{sgn}(\vec{w}^T\vec{x}+b)$ we obtain a decision surface $g(\vec{x})=w_s\vec{f}(\vec{x}_s)^T\vec{f}(\vec{x})+b$ where $\vec{f}(\vec{x})$ is an N component vector function.

Quadratic Kernels

Quadratic kernels are as special case for Polynomial kernels. A polynomial kernel is defined as

$$K(\vec{X},\vec{Z})=\left(\vec{X}^T\vec{Z}+c\right)^N$$

Where $\vec{X}$ and $\vec{Z}$ d-dimensional vectors (e.g. feature vectors) and c ≥ 0 is a free parameter used to trade off the influence of higher-order versus lower-order terms in the polynomial. When c = 0, the kernel is called homogeneous.

A Quadratic kernel is the special case where N=2. In this case we can show that

$$K(\vec{X},\vec{Z}) = \left(\vec{X}^T\vec{Z} + c\right)^2 = \sum_{d=1}^{D}(x_d^2)(z_d^2) + \sum_{i=2}^{D}\sum_{j=1}^{i-1}\left(\sqrt{2}x_i x_j\right)\left(\sqrt{2}z_i z_j\right) + \sum_{d=1}^{D}(\sqrt{2c}x_d)(\sqrt{2c}z_d) + c^2$$

This was sometimes referred to as the "Kernel Trick" and explained as mapping the features to a higher dimensional space where the data was separable.

In the late 1990's Quadratic kernels were popular, and quadratic kernels continue to be used in natural language processing (NLP).

When used with Support Vector Machines, it can be more convenient to think of kernel methods as instance-based learners in which the kernel acts as a similarity function, and the training algorithm learns by example.

A small representative set $\left\{\vec{X}_s\right\}$ of the training data $\left\{\vec{X}_m\right\}$ is selected and 0used to construct a discriminant function as a weighted sum of vector products between the observation and the selected examples. An observation is classified based on a weighted sum of similarity scores.

For instance, a kernalized binary classifier is defined as

$$\hat{y} = \operatorname{sgn}\left(\sum_{s=1}^{M_s} w_s y_s k(\vec{X},\vec{X}_s)\right)$$

Where the weights, w_s, are learned in training.

In computer vision, Radial Basis Functions have emerged as a popular choice for kernel function.

Radial Basis Function Kernels

A radial basis function (RBF) is a real-valued function whose value depends only on the distance from the origin.

For example, the Gaussian function $g(z) = e^{-\frac{z^2}{2\sigma^2}}$

is a popular Radial Basis Function with $z = \| \vec{X} - \vec{x}_s \|$ defined by samples from training data. In this case:

$$K(\left\| \vec{X} - \vec{x}_s \right\|) = e^{-\frac{\left\| \vec{X} - \vec{x}_s \right\|^2}{\sigma^2}}$$

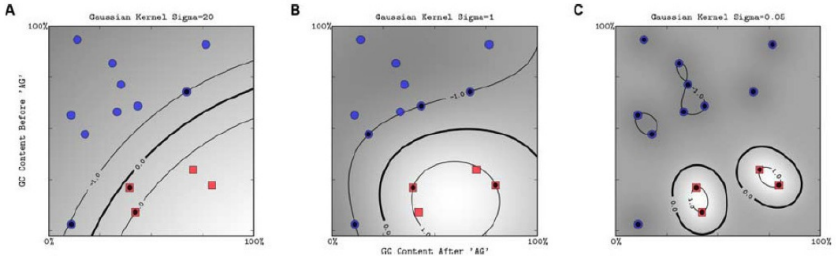
Where each center point, $\vec{x}_s$, is one of the support vectors.

We can use a sum of M_s radial basis functions to define a discriminant function, where the support vectors are drawn from the M training samples.
This gives a discriminant function of the form:

$$g(\vec{X}) = \sum_{i=1}^{M_s} w_i y_i K(\| \vec{X} - \vec{x}_i \|) + b_i \ ,$$

Where the training samples $\vec{x}_i \in \{\vec{X}_s\}$ are the support vectors.

The standard deviation, σ, provides a free variable that constrains the rigidity (or curvature) of the decision surface. The standard deviation, σ, provides a compromise between generality and over-fitting and is chosen according to the distance between the classes. If σ is small compared to the distance between classes, the surface will over-fit the samples. If σ is large compared to the distance between the classes, then the decision surface is overly flat and may not capture the special cases in the training data. The standard deviation, σ is generally chosen to be the estimated distance between the closest members of the two classes.



(images from "A Computational Biology Example using Support Vector Machines", Suzy Fei, 2009)

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}

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.