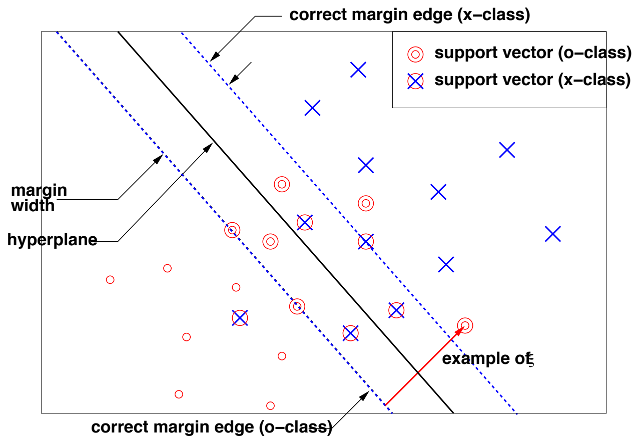
University of Cambridge
Engineering Part IIB

Paper 4F10: Statistical Pattern Processing

Handout 9: Support Vector Machines



Mark Gales
mjfg@eng.cam.ac.uk
Michaelmas 2013

Linear Decision Boundaries

A number of criteria have been presented for training linear decision boundaries:

- Gaussian class-conditional distributions (tied covariances);
- perceptron algorithm;
- logistic regression/classification
- Fisher's discriminant analysis.

A linear decision boundary is characterised by

- direction: normally written $\boldsymbol{w}$;
- bias: normally written b.

For the binary (two class) problem the decision rule is

$$\hat{\omega} = \begin{cases} 1, & \boldsymbol{w}'\boldsymbol{x} + b \geq 0 \\ -1, & \boldsymbol{w}'\boldsymbol{x} + b \leq 0 \end{cases}$$

In the literature it is common to write the dot-product as

$$\boldsymbol{w}'\boldsymbol{x} = \langle \boldsymbol{w}, \boldsymbol{x} \rangle$$

This will be the notation used in the next two lectures.

Training Data

The linear decision boundaries examined in these lectures are:

- static: the observations are of fixed length;
- binary: there are only two classes.

Supervised training data will be used. This consists of pairs training *samples* of the form

$$\{\{\boldsymbol{x}_1, y_1\}, \ldots, \{\boldsymbol{x}_m, y_m\}\}$$

where

- $\boldsymbol{x}_i$ is the observation for the i^{th} sample;
- y_i is the class label for the i^{th} sample ($y_i \in \{-1, 1\}$)

Correct classification of the training will satisfy

$$y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b\right) \geq 0$$

This form of classification will be use for SVM training.

Generalisation Revisited

The concept of generalisation and its importance was described at the start of the course.

For a classifier, $f(\boldsymbol{x}, \boldsymbol{\theta})$, trained with parameters $\boldsymbol{\theta}$, the expected test error may be expressed as

$$R(\boldsymbol{\theta}) = \sum_y \int \frac{1}{2} |y - f(\boldsymbol{x}, \boldsymbol{\theta})| \, p(\boldsymbol{x}, y) d\boldsymbol{x}$$

$R(\boldsymbol{\theta})$ is called the expected risk (sometimes called actual risk).

Normally it is not possible to find $p(\boldsymbol{x}, y)$, so the empirical risk $R_{\text{emp}}(\boldsymbol{\theta})$ is used

$$R_{\text{emp}}(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} |y_i - f(\boldsymbol{x}_i, \boldsymbol{\theta})|$$

We know this is a *lower bound*, so

$$R(\boldsymbol{\theta}) \geq R_{\text{emp}}(\boldsymbol{\theta})$$

Though interesting, an upper bound would be more useful.

One famous upper bound is based on the use of the VC dimension. With probability $1 - \eta$ $(0 \leq \eta \leq 1)$

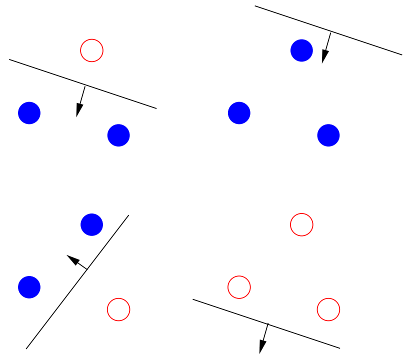
$$R(\boldsymbol{\theta}) \leq R_{\text{emp}}(\boldsymbol{\theta}) + \sqrt{\left(\frac{h(\log(2m/h) + 1) - \log(\eta/4)}{m}\right)}$$

h is the VC dimension. If the empirical error rate is zero then a classifier is selected that minimises the right-hand expression.

VC Dimension

For the binary case the VC dimension is defined as

> The VC dimension for a learning machine is defined as the maximum number of training points that can be shattered by that learning machine.



The above diagram shows an example of three points in $\mathcal{R}^2$ being shattered by a linear decision boundary. The higher the dimension the higher the number of points that can be shattered.

The bound on the previous page varies as:

- the number of training examples, m, increases so the bound is closer to $R_{\text{emp}}(\boldsymbol{\theta})$;
- as the VC dimension, h, increases so the RHS of the bound gets larger.

Support Vector Machine

The support vector machine is one approach to training linear decision boundaries.

The training of an SVM aims is related to attempting to minimising the RHS of the bound of expected risk.

It has some other attractive properties:

- unique solution (compare to perceptron algorithm);
- possible to kernelise training and classification;
- they work!

The training criterion for SVMs can be summarised as:

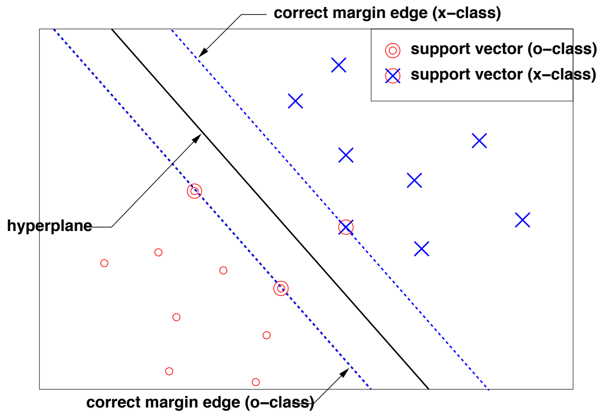
> find the optimal decision boundary distinguished by the maximum margin of separation between any point and the decision boundary.

If you are interested in SVMs, and other related kernel-based schemes, information is available on the web at

`http://www.kernel-machines.org`

Maximum Margin Classifier

An illustration of a maximum margin classifier is below.



It is simple to see that the unique hyperplane that maximises the margin is the one shown.

This may be expressed mathematically as

$$\max_{\boldsymbol{w},b} \min \{||\boldsymbol{x} - \boldsymbol{x}_i||; \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0, i = 1, \ldots, m\}$$

There are various theoretical arguments why specifying a hyperplane in this fashion will yield good generalisation.

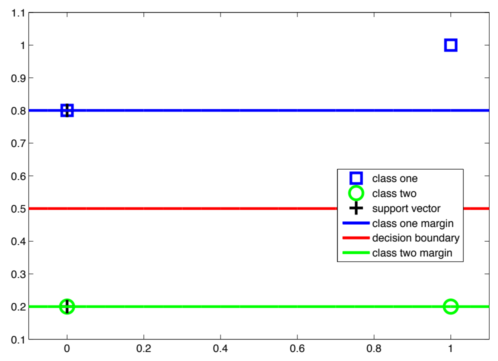
In support vector machines the points that lie on the margin are called support vectors. We will see that the decision boundary can be specified in terms of only these points. This yields a sparse representation.

Simple Example

Consider the following set of points:

$$\left\{\left\{\begin{bmatrix}1\\1\end{bmatrix}, 1\right\}, \left\{\begin{bmatrix}0\\0.8\end{bmatrix}, 1\right\}, \left\{\begin{bmatrix}1\\0.2\end{bmatrix}, -1\right\}, \left\{\begin{bmatrix}0\\0.2\end{bmatrix}, -1\right\}\right\}$$

The decision boundary (and margins) are given below:



There are 2 support vectors (often vectors on the margin which do not contribute to the decision boundary are ignored). These lie on the margin and determine the decision boundary.

For this problem one solution is

$$\boldsymbol{w} = \begin{bmatrix}0\\1\end{bmatrix}; \quad b = -0.5$$

Size of the Margin

For a linear classifier, the shortest distance from the origin to the decision hyperplane is

$$d = \frac{|b|}{||\boldsymbol{w}||}$$

There are two margins to consider the x class (+) and the o class (-). These will be labelled d_+ and d_- respectively.

For linear classifiers $\boldsymbol{w}$ and b can be scaled arbitrarily without affecting discrimination. To avoid this problem constants are introduced to fix the scaling. Typically

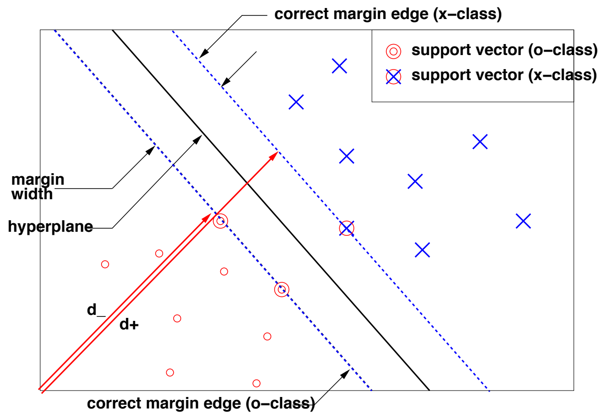
$$\begin{aligned} \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \geq 1, &\qquad \text{for } y_i = +1 \\ \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \leq -1, &\qquad \text{for } y_i = -1 \end{aligned}$$

The use of 1 in the constraint is an arbitrary choice. Any positive real number will do.

Note: the shortest distance from the line $\boldsymbol{w}'\boldsymbol{x} + b$ to the origin is given by the dot-product of a point on the line, $\boldsymbol{x}$, with the perpendicular unit vector:

$$\frac{|\langle \boldsymbol{w}, \boldsymbol{x} \rangle|}{||\boldsymbol{w}||} = \frac{|b|}{||\boldsymbol{w}||}$$

Size of the Margin (cont)



The respective distances to the origin for each of the margins, which are a unit distance from the hyperplane are

$$\begin{aligned} d_+ &= \frac{|1 - b|}{||\boldsymbol{w}||} \\ d_- &= \frac{|-1 - b|}{||\boldsymbol{w}||} \end{aligned}$$

The margin is therefore

$$|d_+ - d_-| = \frac{2}{||\boldsymbol{w}||}$$

SVM Training

The maximum margin optimisation may be rewritten as

$$\min_{\boldsymbol{w},b} \left\{ E(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \right\}$$

subject to

$$y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \right) \geq 1$$

for all $i = 1, \ldots, m$.

This is a standard constrained optimisation problem. As usual introduce a positive Lagrange multiplier, α_i, for each of the m constraints. The Lagrangian to minimise

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{m} \alpha_i \left(y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \right) - 1 \right)$$

From this primal the following KKT conditions can be stated.

$$\begin{aligned}
\frac{\partial}{\partial b} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) &= -\sum_{i=1}^{m} \alpha_i y_i = 0 \\
\frac{\partial}{\partial \boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) &= \boldsymbol{w} - \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i = \mathbf{0} \\
y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \right) &\geq 1 \\
\alpha_i &\geq 0 \\
\alpha_i \left(\left(y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \right) - 1 \right) \right) &= 0
\end{aligned}$$

The last condition results from α_i being non-zero only for samples that lie exactly on the margin. For the points on the margin $(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) - 1 = 0$.

SVM Training (cont)

These conditions lead to

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i$$

The KKT conditions are satisfied at the solution. For convex problems (such as SVM optimisation) finding a solution to the KKT are necessary and sufficient for $\boldsymbol{w}$, b and $\boldsymbol{\alpha}$ to be a solution to the primal.

The dual optimisation problem is commonly solved in practise. Here the following expression is maximised

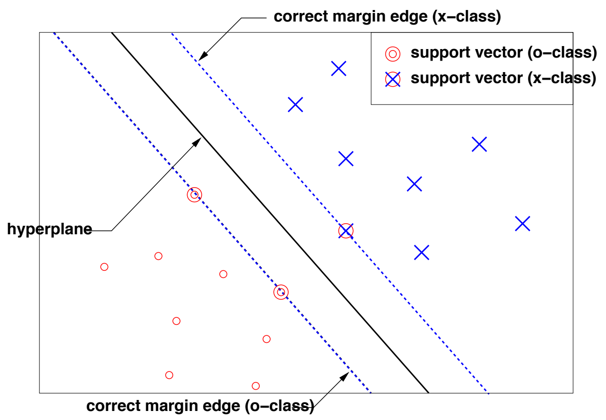
$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$

subject to

$$\begin{aligned}
\alpha_i &\geq 0 \\
\sum_{i=1}^{m} \alpha_i y_i &= 0
\end{aligned}$$

It is interesting that in this dual formulation the b is not explicitly estimated.

Determining the Bias



The bias term b is not explicitly obtained in the optimisation process. From the previous conditions

$$\alpha_i \left(\left(y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \right) - 1 \right) \right) = 0$$

the points may be split into two groups

- correctly classified points that lie beyond the margin

$$\alpha_i = 0$$

- correctly classified points that lie on the margin (the support vectors)

$$\alpha_i \geq 0$$

Selecting a value of i for which α_i is non-zero will simply yield the value of b. A more accurate estimate is found from averaging all such values.

Training

The optimisation of the dual is a quadratic programming problem. There are a number of approaches that have been adopted to this.

If the support vectors (SVs) are known then the problem is quite simple. These can sometimes be obtained by using arguments of symmetry by inspection (see XOR and simple example). However in practise it is often not possible (see Iris data).

One elegant aspects of SVM training is that there is a unique global solution to the problem. The optimisation is strictly concave.

There are a number of toolkits for training SVMs. For a list of software see

`http://www.support-vector-machines.org`

SVM Classification

The standard linear classification rule is

$$\hat{\omega} = \begin{cases} 1, & \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \geq 0 \\ -1, & \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \leq 0 \end{cases}$$

From the previous slide the dual form of the decision boundary where

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i$$

it is possible to write

$$\begin{aligned} \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b &= \left\langle \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i, \boldsymbol{x} \right\rangle + b \\ &= \sum_{i=1}^{m} y_i \alpha_i \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle + b \end{aligned}$$

Note:

- classification is only a function of the support vectors
- classification requires dot products between the support vectors and $\boldsymbol{x}$.

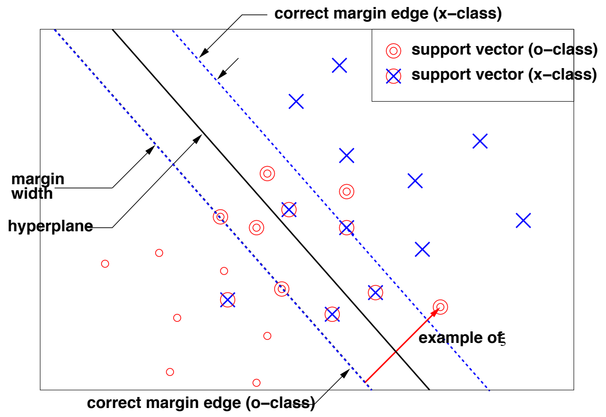
The importance of being able to write the classification and training in this form will become clear in the next slides.

Non-Separable Case

The derivation so far has assumed that the data is linearly separable. In many real-world applications this is not the case.

To allow for training data errors, slack variables, $\xi_i \geq 0$, are introduced. This relaxes the previous constraint to be

$$y_i \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \right) \geq 1 - \xi_i$$



All the points that lie outside the required margin (of +/-1) will have $\xi = 0$. For a classification error of the training data to occur $\xi > 1$, so a simple upper bound on the training errors is given by

$$N \text{error} \leq \sum_{i=1}^{m} \xi_i$$

Non-Separable Case (cont)

There are now two aspects of the optimisation

1. maximise the margin;
2. minimise the number of training errors.

The balance of the two is commonly controlled by a variable, C.

The soft-margin classifier is obtained by minimising

$$E(\boldsymbol{w},\boldsymbol{\xi}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{m}\xi_i$$

subject to

$$\begin{aligned} y_i\left(\langle \boldsymbol{w},\boldsymbol{x}_i\rangle + b\right) &\geq 1-\xi_i \\ \xi_i &\geq 0 \end{aligned}$$

The dual optimisation problem for this may also be set-up, now

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \left\langle \boldsymbol{x}_i,\boldsymbol{x}_j\right\rangle$$

subject to

$$\begin{aligned} 0 &\leq \alpha_i \leq C \\ \sum_{i=1}^{m}\alpha_i y_i &= 0 \end{aligned}$$

Note the slack variable no longer occurs in the optimisation.

Determining the Bias

In a similar fashion to the separable case the bias, b has not been determined. One of the conditions that is satisfied is

$$\alpha_i\left(y_i\left(\langle \boldsymbol{w},\boldsymbol{x}_i\rangle + b\right) - 1 + \xi_i\right) = 0$$

This has a sensible feel about it

- for correctly classified points that lie beyond the margin

$$\alpha_i = 0, \quad \xi_i = 0$$

- for correctly classified points that lie on the margin

$$0 \leq \alpha_i \leq C, \quad \xi_i = 0$$

- points that lie “within” the margin

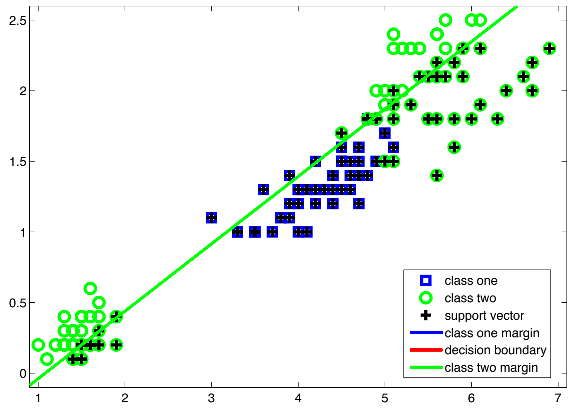
$$0 \leq \alpha_i \leq C, \quad \xi_i > 0$$

Thus to determine b select a correctly classified point that lie on the margin.

The “Kernel Trick”

Only linear hyperplanes have been considered so far. This limits the ability of the classifier.

Consider some standard test data, the Iris data.



A linear decision boundary was trained using an SVM and the result is shown above. Here $C = 5000$.

For this data there is clearly no linear decision boundary that will simply partition the data. One solution to this problem is to use a classifier with a non-linear decision boundary, for example a multi-layer perceptron, or Gaussian mixture model.

An alternative approach is to expand the feature space, so the the points become separable - use a kernel

The XOR problem

A classic problem is the `XOR` problem. The training data for this is:

$$\left\{\left\{\begin{bmatrix}1.0\\-1.0\end{bmatrix},1\right\},\left\{\begin{bmatrix}-1.0\\1.0\end{bmatrix},1\right\},\left\{\begin{bmatrix}1.0\\1.0\end{bmatrix},-1\right\},\left\{\begin{bmatrix}-1.0\\-1.0\end{bmatrix},-1\right\}\right\}$$

This data is clearly not separable with a linear decision boundary. Consider the following mapping:

$$\boldsymbol{x} = \begin{bmatrix}x_1\\x_2\end{bmatrix} \rightarrow \boldsymbol{\Phi}(\boldsymbol{x}) = \begin{bmatrix}1\\\sqrt{2}x_1\\\sqrt{2}x_2\\\sqrt{2}x_1x_2\\x_1^2\\x_2^2\end{bmatrix}$$

Each point is now mapped from a 2-dimensional space to a 5-dimensional space. The data is separable in this high dimensional space.

The optimisation and constraints may be expressed as

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \left\langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}_j)\right\rangle$$

subject to $\alpha_i \geq 0$ and $\Sigma_{i=1}^{m}\alpha_i y_i = 0$. The direction of the decision boundary is given by

$$\boldsymbol{w} = \sum_{i=1}^{m}\alpha_i y_i \boldsymbol{\Phi}(\boldsymbol{x}_i)$$

Gram Matrix

One way to view the data is use the Gram matrix, $\mathbf{K}$. This is defined as

$$k_{ij} = \langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}_j) \rangle$$

The optimisation is then

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j k_{ij}$$

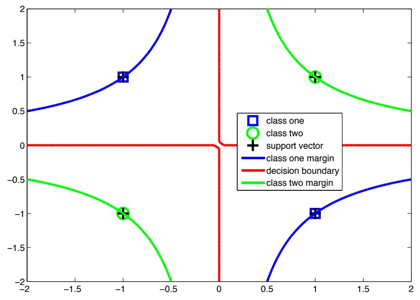
Producing the Gram matrix for the `XOR` problem

$$\mathbf{K} = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

This matrix encapsulates the information about the "points" required for estimating the SVM decision boundary.

For the `XOR` example, the solution is

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{8}$$



Gram Matrix (Cont)

This is interesting - it is not necessary to explicitly work in the feature-space it is only necessary to work with the Gram matrix.

The process has so far been described in terms of mapping a point from the input-space to feature-space

$$\boldsymbol{x} \to \boldsymbol{\Phi}(\boldsymbol{x})$$

and then taking dot-products to form the Gram matrix.

This is not necessary. The Gram matrix can be directly generated by defining a kernel function, $k()$. The elements of the matrix are then found as

$$k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

We will see that some kernels have an infinite feature-space.

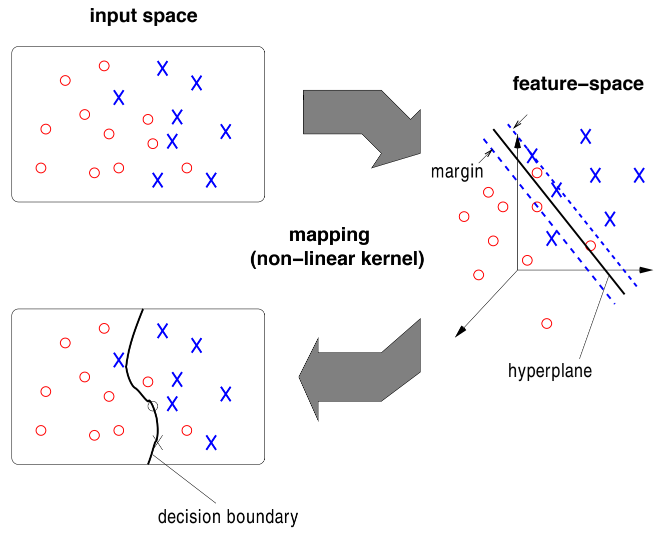
Classification is performed as

$$\hat{\omega} = \begin{cases} 1, & \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle + b \geq 0 \\ -1, & \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle + b \leq 0 \end{cases}$$

From the previous slide the dual form of the decision boundary is

$$\begin{aligned} \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle + b &= \sum_{i=1}^{m} y_i \alpha_i \langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}_i) \rangle + b \\ &= \sum_{i=1}^{m} y_i \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b \end{aligned}$$

Kernel Functions



The diagram above summarises the use of Kernels for SVMs.

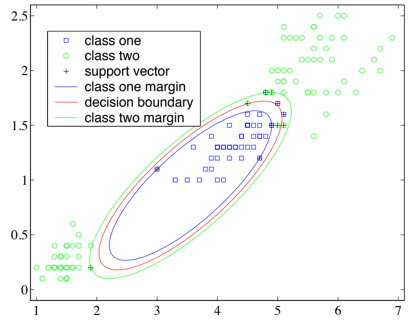
A range of kernel functions are possible - common ones are:

- linear:
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$
- polynomial, order d:
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1)^d$$
- Gaussian, width σ:
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{2\sigma^2}\right)$$

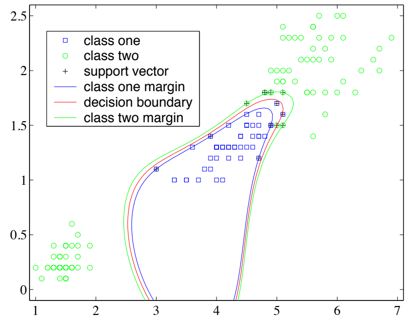
Iris Data

The Iris data is a standard set of data for classification. For this problem it is split into 2 classes and the original observations are in 2-dimensional space.

The linear kernel SVM was previously shown.



2nd order polynomial



Gaussian ($\sigma^2 = 0.5$)

The figures above show the non-linear decision boundaries resulting from altering the kernel.

Kernel	Number SV	% Correct
Linear	112	66.7
Polynomial order 2	13	97.3
Gaussian ($\sigma^2 = 0.5$)	14	98.0

The performance and number of SVs are dramatically altered by choice of the kernel.

Dimensionality of the Feature Space

The choice of Kernel dramatically alters the nature of the decision boundary. Consider a order-2 homogeneous (no offset by 1) polynomial kernel. Thus

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^2$$

What is the effective dimensionality of the feature space?

For the kernel specified this is easy to find. Consider

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

This yields the required polynomial kernel. The effective dimensionality is 3. Thus the use of the kernel maps the data from $\mathcal{R}^2$ to $\mathcal{R}^3$.

The mapping is not unique. For example

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \frac{1}{\sqrt{2}} \begin{bmatrix} (x_1^2 - x_2^2) \\ 2x_1x_2 \\ (x_1^2 + x_2^2) \end{bmatrix}$$

The same kernel can be obtained with different mappings.

Infinite Feature Spaces

The rank of the Gram matrix, corresponds to the dimensionality of the feature space (provided that the number of training samples is at least greater than the dimensionality of the feature space).

Consider the class of kernels where

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) \rightarrow 0$$

for $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ as

$$||\boldsymbol{x}_1 - \boldsymbol{x}_2|| \rightarrow \infty$$

The dimensionality of the feature space under these conditions will become infinite as the amount the training data becomes infinite.

One kernel that can satisfy these constraints is the Gaussian kernel. This can be achieved by setting the value of σ^2 small enough.

Mercer's Condition (reference)

The kernels have been described in two forms $\boldsymbol{\Phi}(\boldsymbol{x})$ and $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. When does there existing a mapping for

$$k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

The answer is Mercer's condition.

This states that a mapping exists if and only if, for any $g(\boldsymbol{x})$ such that

$$\int g(\boldsymbol{x})^2 d\boldsymbol{x} \quad \text{is finite}$$

then

$$\int k(\boldsymbol{x}, \boldsymbol{y}) g(\boldsymbol{x}) g(\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} \geq 0$$

Kernels of any form could be used but if the above condition is not satisfied then the Hessian may not be positive semi-definite and the optimisation will have no solution (the dual can become arbitrarily large).

Kernelising Classifiers

It is possible to apply the kernel trick to other classifiers. If the training of the classifier and its use involves dot products (and linear operations) then it can be simply kernelised.

This trick has been applied (and will be further applied) to various algorithms have been kernelised, for example Kernel Principal Component Analysis. Here we will examine the kernel perceptron algorithm.

In previous lectures the perceptron algorithm was presented. The training algorithm may be written as

```
Initialise $\boldsymbol{w} = \boldsymbol{0}$, $k = 0$ and $b = 0$;
Until all points correctly classified do:
    k=k+1;
    if $\boldsymbol{x}_k$ is misclassified then
        $\boldsymbol{w} = \boldsymbol{w} + y_k \boldsymbol{x}_k$
        $b = b + y_k$
```

To increase the classification ability of the perceptron algorithm it would be useful to kernelise it.

Kernelised Perceptron Algorithm

The kernelised version of the algorithm may be described as

```
Initialise $\alpha_i = 0,\ i = 1, \ldots, m$, $k = 0$ and $b = 0$;
Until all points correctly classified do:
      k=k+1;
      if $\boldsymbol{x}_k$ is misclassified then
            $\alpha_k = \alpha_k + 1$
            $b = b + y_k$
```

The equivalent of the Lagrange multipliers here represent the number of times that a particular point has been misrecognised.

Classification is then performed based on (as for the SVM)

$$\sum_{i=1}^{m} y_i \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b$$

This allows the Gram matrix to be used during the classification process, so there is no need to work in the, possibly, high dimensional, feature space.

Just because a training algorithm can be kernelised does not mean that it will perform well. The number of model parameters will increase. For the SVM generalisation is good because of the use of a maximum margin classifier.

Summary

The last two lectures have examined the use of SVM as a classifier and the use of kernels. In particular:

- bounds on generalisation;
- maximum margin classifiers;
- SVM training and the use of the dual optimisation;
- SVM classification;
- non-separable data;
- kernels and their dimensionality;
- kernelising the perceptron algorithm.