# Advanced Topics in Statistics

*~ Support Vector Machines ~*

7

UNIVERSITY OF EXETER

# Introduction.

Support vector machine is an approach for classification developed by the computer science community in the 1990s.

They perform well in a variety of settings. Even though they are mainly used for binary classification problems, extensions for cases with multiple classes do exist.

The support vector machine is an extension of the support vector classifier, which in turn is an extension of the maximal margin classifier.

- Maximal margin classifier is a quite simple and elegant method. But at the same time, since it requires the classes to be linearly separable, it can rarely be applied to real datasets.
- Support vector classifier is not so restrictive, but only works well in cases where a linear decision boundary is suitable.
- Support vector machines extend support vector classifiers by allowing non-linear decision boundaries. For this they use a so-called kernel trick, essentially mapping the inputs into a higher dimensional kernel space. (See later).

Here we will discuss all three cases.

# HYPERPLANES.

Before we start discussing support vector machines, we should first understand what hyperplanes are.

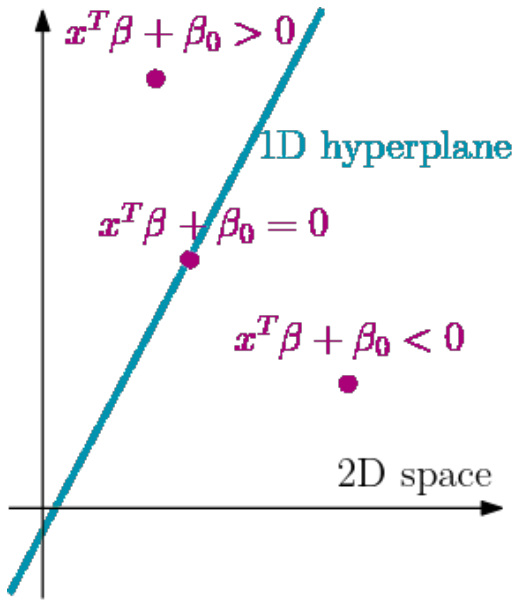
In a $p$-dimensional space, a hyperplane is essentially a $p-1$ dimensional subspace.

In a plane, which is a 2 dimensional space, a hyperplane is a line, which is a 1 dimensional space. While in a 3 dimensional space, a hyperplane is a (2D) plane.

In any dimension the points $(X_1, \ldots, X_p)^T$ of the hyperplane satisfy the equation

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0.$$

E.g. for $p = 2$ we have $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$, which is the equation of a line.




Furthermore

- if $\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p > 0$ then the point lies on one side of the hyperplane, while if
- $\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p < 0$, then the point lies on the other side of the hyperplane.

*Thus a hyperplane is dividing a p-dimensional space into two halves.*

# MAXIMAL MARGIN CLASSIFIER.

Imagine a situation where you have a two class classification problem with two predictors $X_1$ and $X_2$.

Suppose that the two classes are 'linearly separable' i.e. one can draw a straight line in which all points on one side belong to the first class and points on the other side to the second class.

Then a natural approach is to find the straight line that gives the biggest separation between the classes i.e. the points are as far from the line as possible. The line with the biggest separation is what gives the smallest test error.

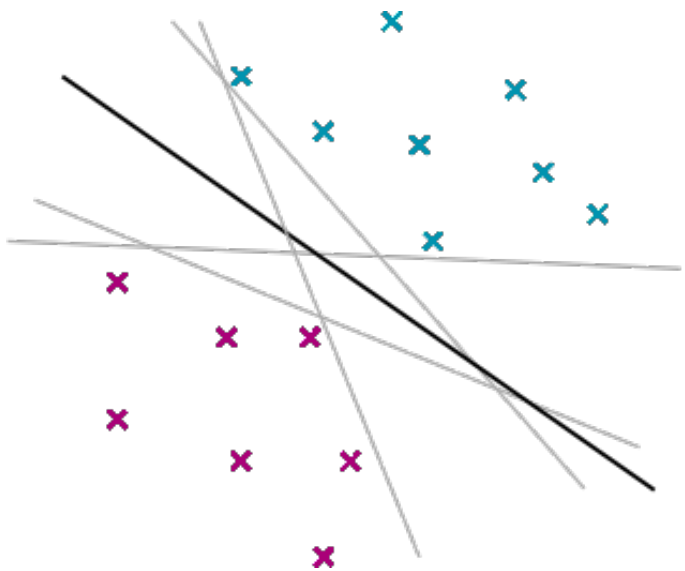
This is the basic idea of a support vector classifier.

The figure shows two classes that are separable.

In this case there are an infinite number of lines that separate these two classes. The figure shows just a couple.

In order to minimise the test error rate we should choose the one that's furthest away from all datapoints.

This is called the maximal margin hyperplane.



# Maximal margin hyperplane example.

Consider the following example that shows two separable classes and a separating hyperplane.

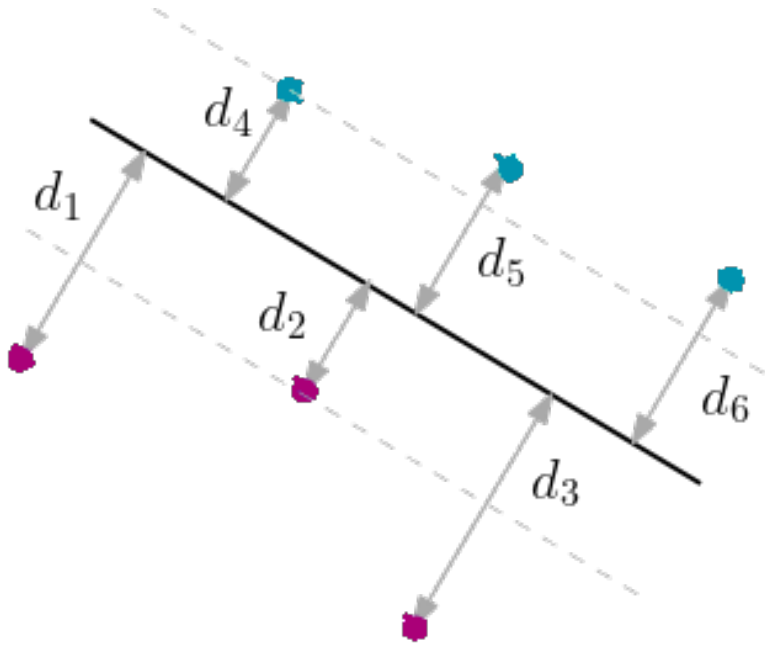
We can measure the perpendicular distance between each point and the separating line. These are $d_1, d_2, \ldots, d_6$.

We will denote the minimum of these distances by $d^*$, that is

$$d^* = \min\{d_1, d_2, d_3, d_4, d_5, d_6\}.$$

This minimal distance is called the margin.

The idea of the maximum margin classifier is to choose the line for which $d^*$ is the largest.




Or in other words, choose the separating hyperplane for which this margin is the largest, we call this line the maximal margin hyperplane.

Assume the given line is the maximal margin hyperplane, and $d_2 = d_4$. Then the vectors $d_2$ and $d_4$ are called support vectors, since they 'support' the 'optimal separating hyperplane'.

# Prediction.

Once we have found the hyperplane that gives the maximal separation, classification is easy.

- A new observation is assigned to a class depending on which side of the line it falls on.
- We have seen that the hyperplane divides the space into two halves, and we can use the defining expression to decide on which side the point falls on.
- Thus, when we want to predict the label of a new observation we just have to plug the coordinates of the new point into the defining expression of the optimal separating hyperplane.
- The sign of the outcome determines the class we predict.

Our example was in 2 dimension, which corresponds to the two-predictor case. But this idea works just as well with more than two predictors.

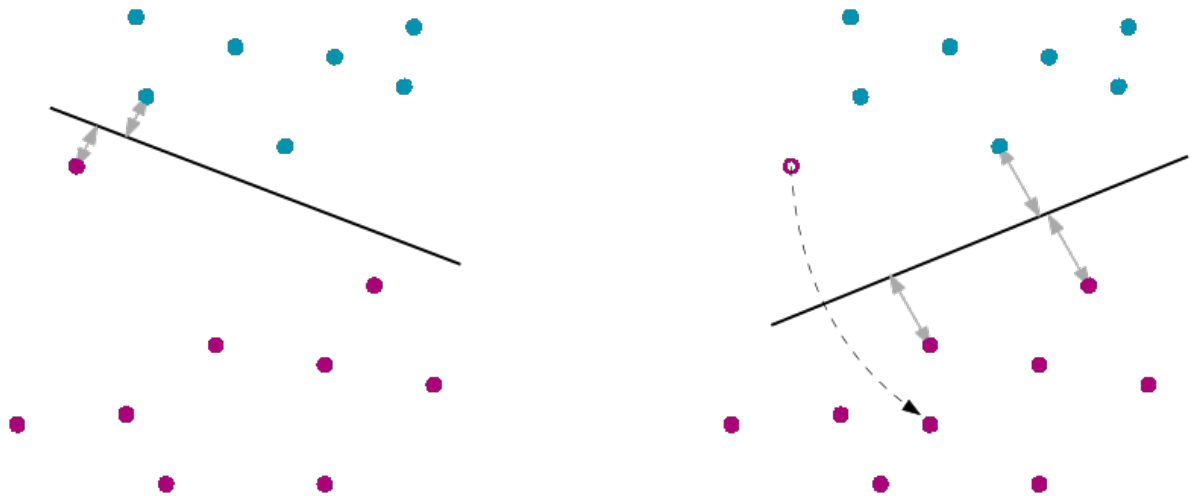
For example, with three predictors you want to find the plane that produces the largest separation between the classes.

With more than three dimensions it becomes hard to visualise a hyperplane but it still exists.

# PROBLEMS WITH THE MAXIMAL MARGIN CLASSIFIER.

1. Even when the classes are separable, the maximum margin classifier overfits the data.

   The maximal margin hyperplane only depends on the support vectors, but not on the other observations. So it's quite sensitive to a small change in one of these support vectors.



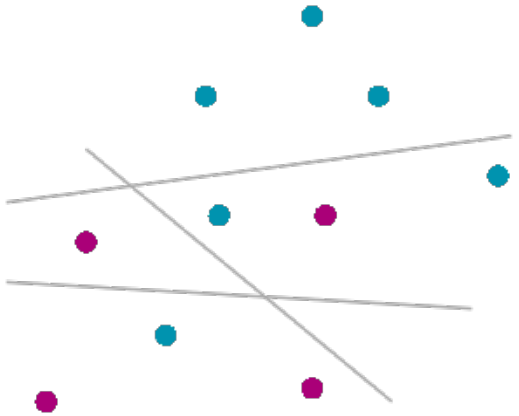
2. The main issue however is that in practice it is not usually possible to find a hyperplane that perfectly separates the two classes.

   In other words, for any straight line or plane that I draw there will always be at least some points on the wrong side of the line.

# Non-separable classes.

The following figure shows a two-class example where the classes are not separable.

In this situation we try to find the plane that gives the best separation between the points that are correctly classified *subject to the points on the wrong side of the line not being off by too much*.



The idea is that we introduce some so called slack variables, that allow observations to be on the wrong side of the margin.

- If the given observation is on the right side of the margin, then the slack variable corresponding that observation is zero.
- If the observation is on the wrong side of the margin, but on the correct side of the hyperplane then the slack variable is somewhere between 0 and 1.
- And finally, if the observation is on the wrong side of the hyperplane, which will result in incorrect classification, then the slack variable is greater than 1.

# SUPPORT VECTOR CLASSIFIER.

Here the slack vector associated to observation $i$ is denoted by $\xi_i$.

We can set up an optimisation problem using these slack variables.

We still want to maximise the margin, but now this maximisation is subject to some constraints.

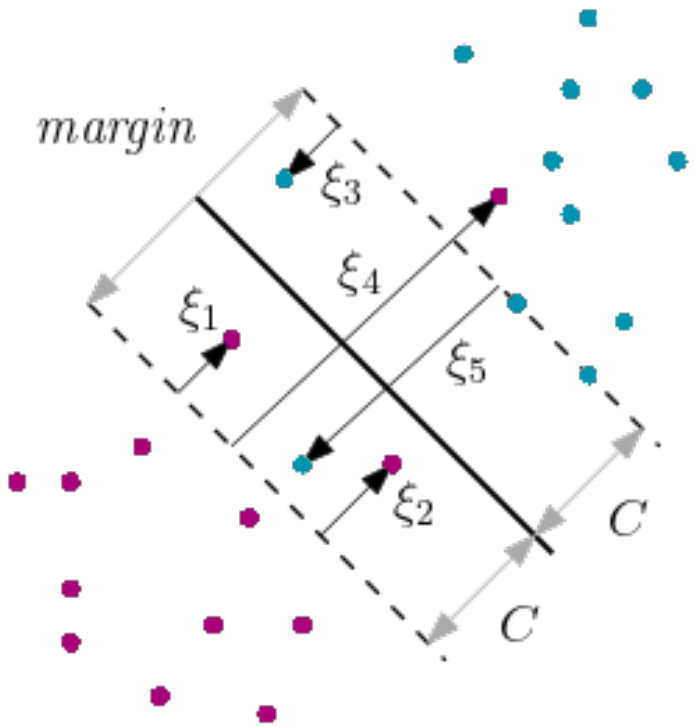
That is, we want to maximise $C$ (half the margin) subject to

$$\frac{1}{C}\sum_{i=1}^{n}\xi_i \leq \text{constant},$$

where the constant is a tuning parameter that we can choose.

E.g. if the constant is one, that allows at most $C$ observations to be on the wrong side of the separating hyperplane.

It's this constant that controls the bias-variance trade-off.




The solution of this optimisation problem is going to be the equivalent of the separating hyperplane. We call the margins soft margins.

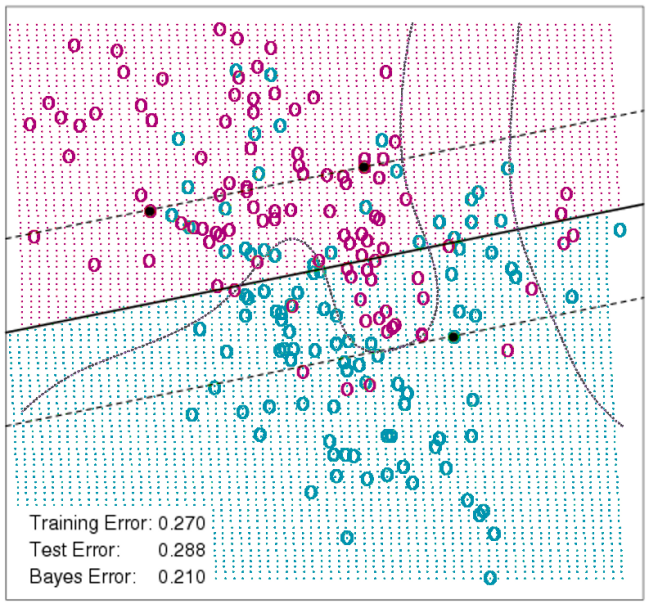
# Support Vector Classifier Example.

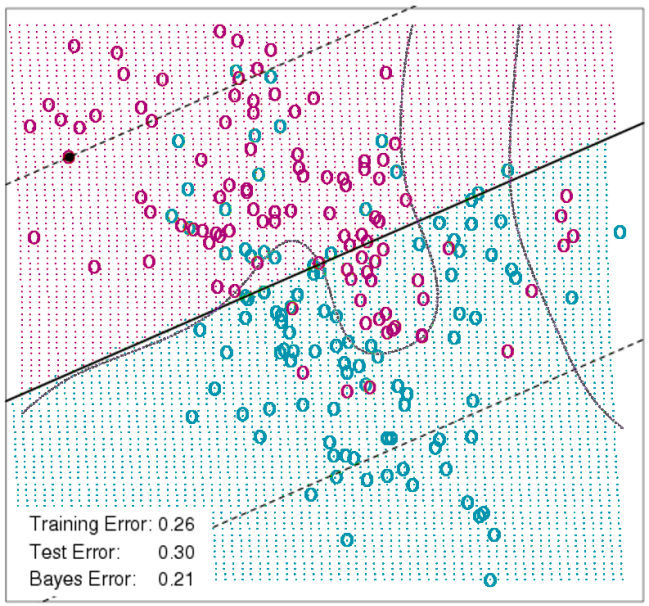
Consider the following simulated example. Here it is quite clear that the two classes are not separable.

The distance between the dashed lines in the margin.

We used a larger margin to create the support vector classifier on the right hand side. This results in a greater margin, and creates a slightly different classifier.

Notice that the decision boundary of a support vector classifier must always be linear.



# Relaxing linearity.

Support vector machines relax the linearity of the support vector classifiers, and thus have the potential to create non-linear decision boundaries.

The idea is similar to how GAMs extend linear regression to cases where the relationship between the predictors and the outcome is non-linear.

The linear case:

- Linear regression takes the linear combination of the predictors $\boldsymbol{X} = (X_1, \ldots, X_p)$,

$$Y_i = \alpha_0 + \alpha_1 X_{1i} + \cdots + \alpha_p X_{pi} + \varepsilon_i, \quad i = 1, \ldots, n,$$

  that is it finds an optimal solution in the space spanned by these predictors.

The non-linear case:

- GAMs enrich the feature space by considering functions of the predictors, called basis functions,

$$Y_i = \beta_0 + \beta_1\, b_1(\boldsymbol{X}_i) + \beta_2\, b_2(\boldsymbol{X}_i) + \cdots + \beta_M\, b_M(\boldsymbol{X}_i) + \varepsilon_i, \quad i = 1, \ldots, n.$$

  That is GAMs find the optimal solution not in the space spanned by the predictors, but in the space that's spanned by the basis functions.

# SUPPORT VECTOR MACHINES.

The linear case of the classification setting:

- The support vector classifier finds the optimal hyperplane in the space spanned by the predictors $\boldsymbol{X} = (X_1, \ldots, X_p)$.

The non-linear case of the classification setting:

- Conceptually, the support vector machine classifier transforms the predictors, and find the optimal hyperplane in the space spanned by $b_1(\boldsymbol{X}), \ldots, b_M(\boldsymbol{X})$.

This approach produces a linear plane in the transformed space but a non-linear decision boundary in the original space.

While conceptually the basis approach is how the support vector machine works, in reality we don't actually end up choosing basis vectors.

But instead we choose something called a kernel function. We can think of the kernel as a function that quantifies the similarity of two observations.

Common kernel choices include:

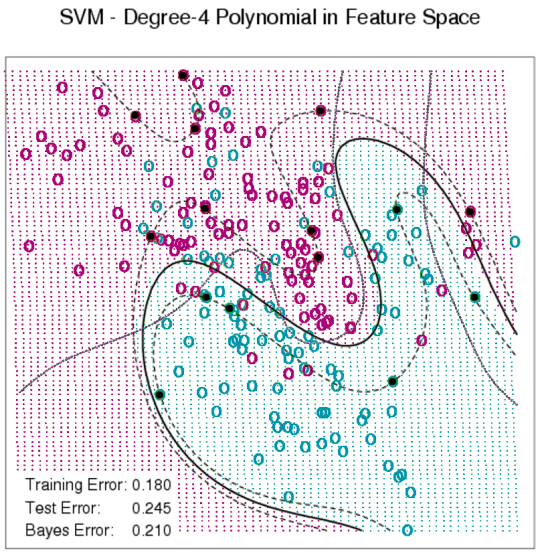
- Linear kernel (same as the support vector classifier)
- Polynomial kernel
- Sigmoid kernel
- Radial Basis kernel

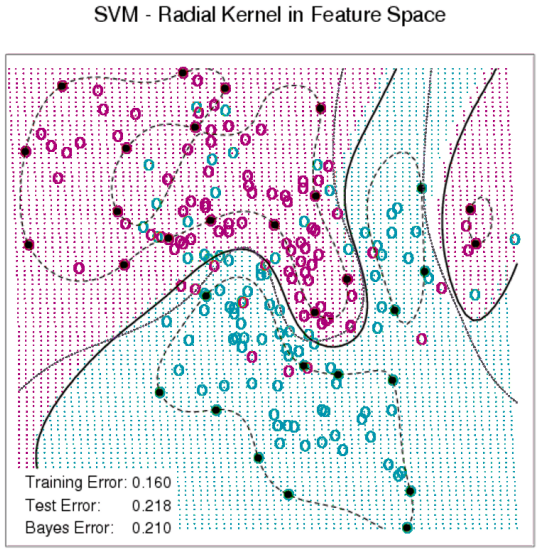
# Support Vector Machine Example.

Recall that for the previous example the support vector classifier gave a training error rate of 0.26 and a test error rate of 0.3.

Using a polynomial kernel on the previous example gives a much lower error rate than the error rates produces by the support vector classifier.

Replacing the polynomial kernel by a radial basis kernel reduces the error rate even futher.




The support vector machine is quite a robust method. When we are optimising the performance we not only have to choose the right kernel but also optimise the associated tuning parameters.