Kernels

Kernel Approach:

Choose a kernel function k which measures the similarity between raw data vectors $ k(x_m,x_n) $ Learn parameters using the values and class labels. When making a prediction for a test data vector use the learned parameters to make a prediction.

Design Matrix = Data after performing feature mapping. where the nth row is the nth data point.

$$w = \Phi^T a \qquad a = \text{N-dimensional parameter vector} \qquad \Phi = \text{Design Matrix}$$

$$y(x) = w^T \phi(x) = a^T \Phi \phi(x)$$

$$k(x, x') = \phi(x)^T \phi(x')$$

The kernel function represents the degree of similarity between its two arguments (since its the dot product of two feature vectors).

In addition to this most of the parameters in a will be close enough to zero that they can be discarded.

Prediction:

$$y(x) = a^T \begin{bmatrix} k(x_1, x) \\ k(x_2, x) \\ k(x_3, x) \end{bmatrix} = a_1 k(x_1, x) + a_2 k(x_2, x) + a_3 k(x_3, x) x = \text{data point}$$

We must learn the parameter vector a without having to evaluate the feature vectors or we lose the benefit of using kernels. Kernel methods are 'non-parametric' because there are the same number of parameters as data points. In addition to this, we only have to care about parameters that have non-zero values, these are the dual coefficients and the corresponding data points are the support vectors.

The Gram Matrix

$$\text{Gram Matrix: } K = \Phi \Phi^T$$

$$\text{Learning: } a = (K + \lambda I_N)^{-1} t$$

Classification:

During classification, there will be many different hyperplanes that will linearly separate the data. We can use a maximum margin classifier to choose the hyperplane that leads to the largest distance between the closest data point to the plane. This will help to reduce overfitting.

$$argmax_{w,b}\{\frac{1}{||w||}\min[t_n(w^T\phi(x_n)+b)]\}$$