$P = \begin{pmatrix} v_1^T \\ v_2^T \\ v_m^T \end{pmatrix}$ where $m < n$ is the number of components we keep

Given an *n-dimensional* input pattern x we can construct an *m-dimensional* pattern z

$z = P(x - \mu)$

Then use z are our new pattern

k) Reconstruction

$\hat{x}_k = \mu + \sum_{i=1}^{m} z_i^k v_i$ Where $z_i^k = v_i^T (x_k - \mu)$ and eigenvectors are normalised

Reconstruction error $\|x_k - \hat{x}_k\|^2$ can be used to measure how much information has been lost

PCA finds a subspace with the smallest reconstruction error

7 PCA in practice

a) Images

When processing real images there are too many features to look at so it creates a very large covariance matrix

Mega pixel image → million wide matrix

P images span at most a *P-dimensional* subspace

This will be much smaller that the *NxN* space described by the images with standard PCA

The Dual Matrix 'trick' can be used to over come this

b) Dual Matrix

Typical covariance matrix is $C = XX^T$ consider the matrix $D = X^T X$

Suppose v is an eigen vector of D

$Dv = \lambda v$

$X^T X v = \lambda v$

$XX^T Xv = \lambda Xv$

$(XX^T) Xv = \lambda Xv$

$CXv = \lambda Xv$ let $u = Xv$

$Cu = \lambda u$

u is an eigen vector of C

c) Single Value Decomposition

Any *NxP* matrix X can be written as $X = USV^T$

Where:

U is an *NxN* orthogonal matrix

V is a *PxP* orthogonal matrix

S is an *NxP* diagonal matrix of singular values

A matrix M can also be described in terms of its eigenvectors and eigenvalues

$M = V \Lambda V^T$

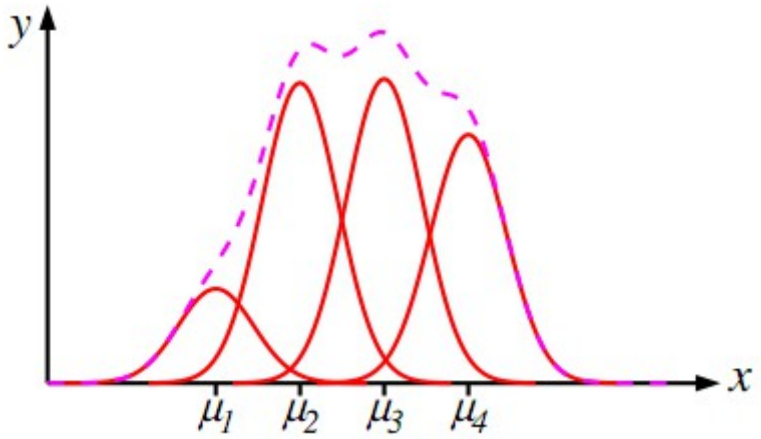


Illustration 2: Example output of a radial basis function. Note the height of the functions is the weight.

This can be combined further to produce a network of nodes and functions with multiple outputs.

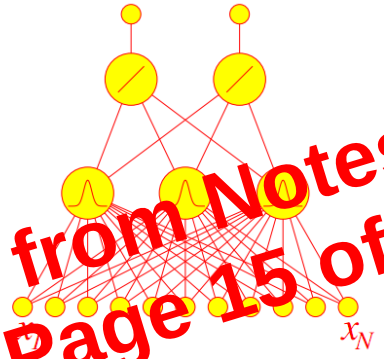


Illustration 3: Radial Basis network. Note the top 2 nodes are simply linear perceptrons.

d) Finding centers

Radial basis functions depend on the distance to centres
Finding these centres and how many are needed can be hard
They are normally found in clusters

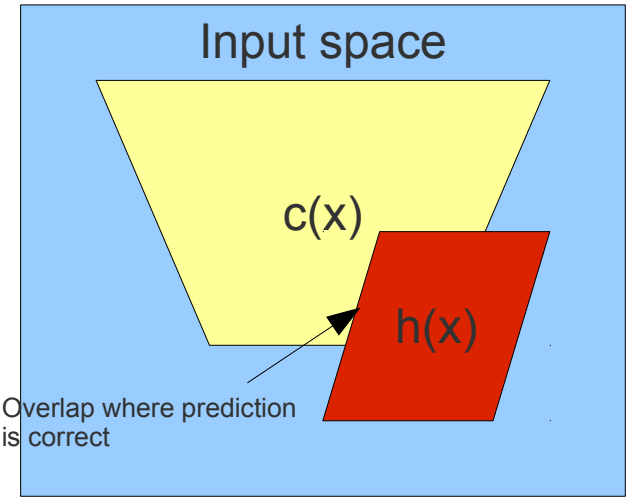
e) k-Means Clustering

Find *k* clusters in the data
Can be tricky to find the right *k*
Easy algorithm and popular

Takes a set of unlabelled data $D=\{x_j\}_1^N$

Algorithm:

1. Choose K
2. Randomly assign each data point to a cluster C_i



d) PAC-Learnable

A concept class C is PAC learn-able if for all concepts in it $c \in C$ and a probability distribution p a learner choosing a hypothesis $h \in H$ can:

achieve a generalisation performance of $E(h|p) < \epsilon$ for $0 < \epsilon < \frac{1}{2}$

with a probability of $1-\delta$ and $0 < \delta < \frac{1}{2}$

using P training examples

where P is a polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$

The idea is to restrict the number of hypothesis's that can be learnt by eliminating those with a generalisation error greater than ϵ. $1-\delta$ refers to the certainty that the machine hasn't learnt a hypothesis with an error greater than ϵ.

e) Finite Consistent Hypothesis Spaces

Suppose the hypothesis space is finite and contains the true answer

Suppose our learning machine is able to correctly categorise all the training examples

$$E(h|p) = 0$$

Consider examples $D = \{(x_k, c(x_k))\}_{k=1}^{P}$

Consider a hypothesis with an errror $E(h|p) > \epsilon$ this means that the probability of it getting all the training examples correct is: $(1-\epsilon)^P$

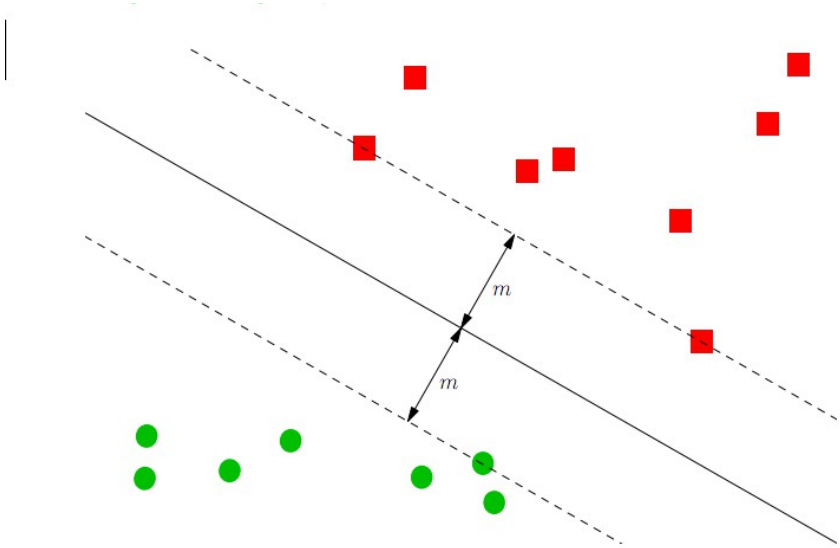
There will be $k < |H|$ hypothesises with $E(h|p) > \epsilon$

The probability of any of these being correct means marginalising over all the k hypothesis

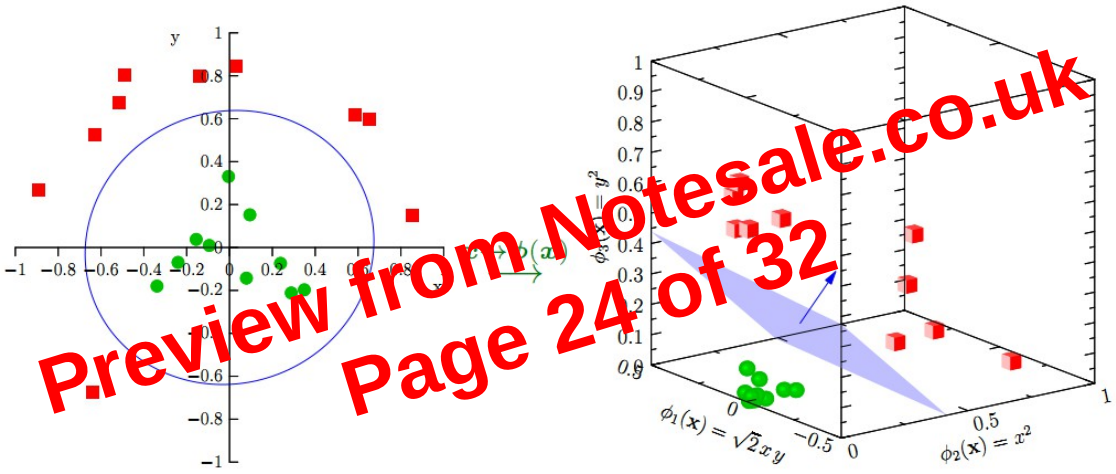
$$k(1-\epsilon)^P < |H|(1-\epsilon)^P \leq |H|e^{-\epsilon P}$$

Using $\delta \geq |H|e^{-\epsilon P}$ the number of training examples needed to get a generalisation error better than ϵ with a probability of $1-\delta$ is given by:

$$\delta \geq |H|e^{-\epsilon P}$$
$$\ln(\delta) - \ln(|H|) \geq -\epsilon P$$
$$\ln(|H|) - \ln(\delta) \leq \epsilon P$$
$$\frac{1}{\epsilon}\ln(|H|) - \ln(\delta) \leq P$$



Can increase the likelihood of finding a separating plane by projecting into high dimensional space using the Kernel trick (see later)



SVMs have the ability to separate almost any data set
However, by choosing the largest margin it selects the most robust solution
Thus giving a good generalisation performance despite having a large capacity to learn data
This is termed *structural risk minimisation*

b) Maximal Margins

Consider a linearly separable data set

$\{(x_k, y_k)\}_{k=1}^{P}$ and $y_k \in \{-1, 1\}$

Task is to find the vector of weights *w* and bias *b* such that

$y_k(\frac{w^T x_k}{||w||} - b) \geq m$ where *m* is the margin

In the image below, we want to maximise the distance from the central separating plane to the nearest data points
The vector from the line to these points is known as *Support Vectors* and these need to have their length maximise

$\lambda_i \geq 0 \forall i$ otherwise the 'distance' between points in the extended space can be negative

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{y})$$

This is the same as saying:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$$

So if you like a kernel is the sum of multiple eigenfunctions and their eigenvalues

For SVMs to work the kernels must be projecting into euclidean space so that distances are always positive

Therefore we must use Positive Semi-definite Kernels ($\lambda_i \geq 0 \forall i$) which can always be decomposed into as sum of positive functions

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{y})$$

d) Integratal of Kernels

$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{y})$ And is positive semi-definite

$$\int \int f(x) K(x, y) f(y) \, dx \, dy \geq 0$$

$$\int \int f(x) \sum_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{y}) f(y) \, dx \, dy$$

$$\sum_i \int f(x) \phi_i(\boldsymbol{x}) \, dx \int \phi_i(\boldsymbol{y}) f(y) \, dy$$

$$\int f(x) \phi_i(\boldsymbol{x}) \, dx = \int \phi_i(\boldsymbol{y}) f(y) \, dy \quad \text{therefore}$$

$$\sum_i \left(\int f(x) \phi_i(\boldsymbol{x}) \, dx \right)^2 = \sum_i \left(\int \phi_i(\boldsymbol{y}) f(y) \, dy \right)^2 \geq 0$$

e) Combining Kernels

Adding

$K_1(x, y)$ and $K_2(x, y)$ are valid kernels

So is $K_3(x, y) = K_1(x, y) + K_2(x, y)$

Multiplied

$K_3(x, y) = K_1(x, y) * K_2(x, y)$ is valid

$K_2(x, y) = K_1(x, y)^n$ is valid too

Exponential

$K_2(x, y) = e^{K_1(x, y)}$ is valid

An important kernel is :

$$K(x, y) = e^{\frac{-\|x - y\|^2}{2}}$$

As it a an infinite number of eigenvalues so if you like its projecting the data into an infinite dimensional space. This is because *e* is an irrational number which can be estimated by an infinite sequence of sums