Lecture 13: Data Modelling

So far we have almost exclusively been considering discrete data. This has the advantage of generality. We can represent any relationship between two variables - however complex - using a discrete link matrix. However, the representation is not compact. If the relationship is highly complex then we are going to need many states to represent it accurately, and this in turn means that the link matrices are very large and therefore difficult to estimate from data. Data distributions enable us to represent relationships between variables in a compact and simple form. It is a well established fact that many naturally occurring data sets have values distributed normally.

$$p(x) = exp(-(x-\mu)^2/(2\sigma^2))/\surd(2\pi\sigma^2) = N(\mu, \sigma^2) \tag{1}$$

If we can find the correct distribution and its parameters we have a compact model that we can use for classification or prediction. We can also use conditional distributions in the place of link matrices. For example if we extend the normal distribution to two variables x, y with equal variance:

$$p(x, y) = exp(-((x-\mu_x)^2 + (y-\mu_y)^2)/(2\sigma^2))/\surd(2\pi\sigma^2) \tag{2}$$

it expresses a continuous joint probability distribution. Knowing a value for say x gives us a probability distribution over the possible values of y. Distributions can also be used in closed form mathematical proofs, though we will not concern ourselves with this in this course. Distributions are usually constructed to represent probability, so the area under them must integrate to 1. The numeric ranges of the data can be infinite (as in the normal (Gaussian) distribution) or set by the parameters, eg the uniform distribution takes the form:

$$\begin{array}{rll} p(y) = & 1/(\beta - \alpha) & \text{for} \alpha < y < \beta \\ & 0 & \text{otherwise} \end{array}$$

There are many other distributions offering different shapes of probability density function. The β distribution is one such with two shape parameters: (see http://mathworld.wolfram.com/)

The multi-variate joint normal (Gaussian) distribution is used when we have two or more variables:

$$p(\boldsymbol{x}) = \frac{exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}))}{(2\pi)^{n/2}\surd|\Sigma|} \tag{3}$$

where:
n is the number of variables
$\boldsymbol{x} = [x_1, x_2, ..x_n]$ is a vector of variables
$\boldsymbol{\mu} = [\mu_1, \mu_2, ..\mu_n]$ is a vector of variable means
Σ is a covariance matrix, Σ^{-1} is its inverse and $|\Sigma|$ its determinant.

The covariance matrix is symmetric with variances on the diagonal:

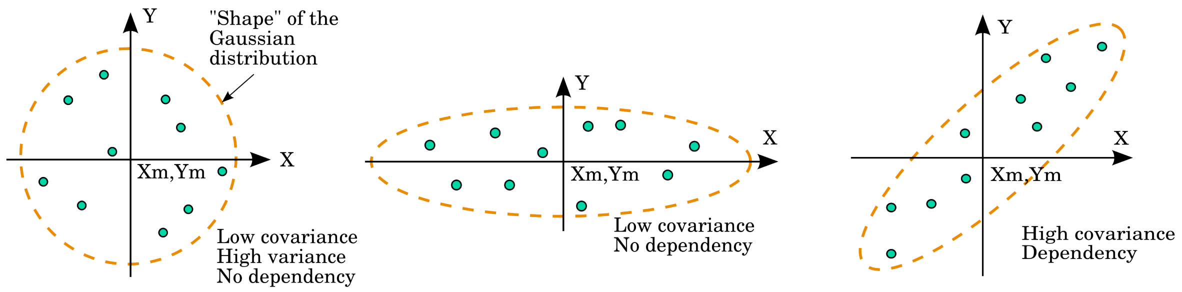
$$\begin{pmatrix} \sigma_{x_1x_1} & \sigma_{x_1x_2} & \sigma_{x_1x_3} & .. & \sigma_{x_1x_n} \\ \sigma_{x_2x_1} & \sigma_{x_2x_2} & \sigma_{x_2x_3} & .. & \sigma_{x_2x_n} \\ \sigma_{x_3x_1} & \sigma_{x_3x_2} & \sigma_{x_3x_3} & .. & \sigma_{x_3x_n} \\ & & & & \\ \sigma_{x_nx_1} & \sigma_{x_nx_2} & \sigma_{x_nx_3} & .. & \sigma_{x_nx_n} \end{pmatrix}$$

The individual elements in the matrix are defined as follows:

$$\sigma_{x,y} = \frac{1}{N-1}\sum_{data}(x_i - \mu_x)(y_i - \mu_y) \tag{4}$$

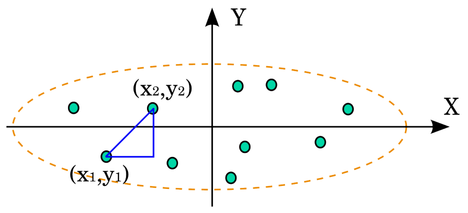
The off diagonal elements are co-variances and indicate data dependencies. Recall that we used correlation as a dependency measure: $Dep(x, y) = Corr(x, y) = \sigma_{x,y}/\surd(\sigma_{xx}\sigma_{yy})$

The concept of co-variance can be illustrated graphically in two dimensions. Suppose that two variables X and Y have similar high variance and low co-variance. Then the Gaussian distribution will be the shaped like a Chico Marx hat placed at the mean point as indicated by the circle in the figure.



On the other hand, if the variances differ greatly, but the co-variances are still small the Gaussian is squashed on one axis, taking on an ellipse shape. As long as the major and minor axes line up with the X and Y axes there is still no dependence between X and Y. If, as shown in the third picture the co-variance becomes large then the ellipse is rotated about its centre. As soon as it is off centre we see a dependence between X and Y which can be modelled using a linear equation.

Mahalanobis Distance



The Euclidian distance treats all dimensions equally, however, statistically they may not be the same. In the picture the variance in the X direction is much bigger than the variation in the Y direction. Let us suppose that this Gaussian function is defining the likely membership of a class. Then moving away from the mean in the Y direction is far more significant than moving the same distance in the X direction. For low (or zero) covariance, as shown in the picture, we can compensate for this effect by dividing the distances on each axis by the variance of the same axis:

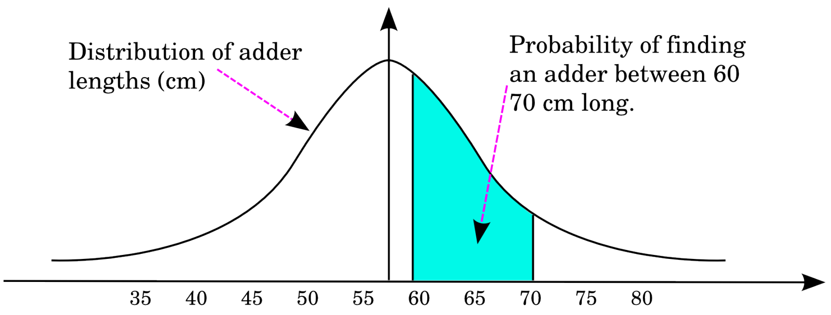
$$\Delta' x = (x_2 - x_1)/\sqrt{\sigma_{xx}}$$
$$\Delta' y = (y_2 - y_1)/\sqrt{\sigma_{yy}}$$
$$\text{Mahalanobis Distance} = \sqrt{(\Delta' x^2 + \Delta' y^2)}$$

In the general case, when the covariance is not zero the Mahalanobis distance between two points can be written:

$$\sqrt{((x_2 - x_1, y_2 - y_1)\Sigma^{-1}(x_2 - x_1, y_2 - y_1))} \quad (5)$$

More on Likelihood and Probability

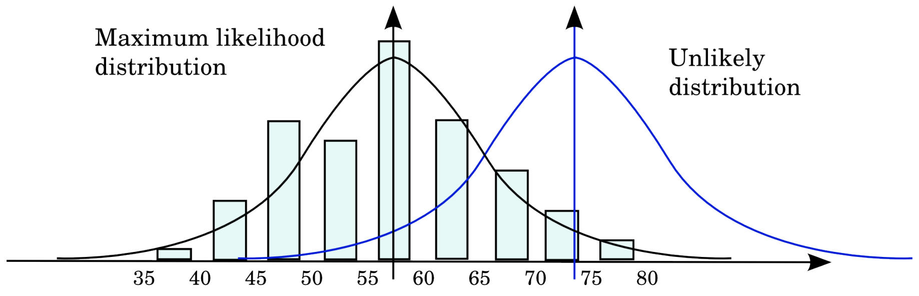
Likelihood and Probability are dual concepts. Given a distribution we can work out the probability of a point whereas given a set of data points we work out the likelihood of a distribution. For example we can simply integrate to find the probability of an adder being between 65 and 75 cm long from an assumed normal distribution of the lengths.


Distribution of adder lengths (cm)
Probability of finding an adder between 60 70 cm long.
35
40
45
50
55
60
65
70
75
80

On the other hand likelihood allows us to consider how likely a particular distribution is, given some data. The likelihood is the product of the individual probabilities of each data point. For the Gaussian distribution the log likelihood function is:

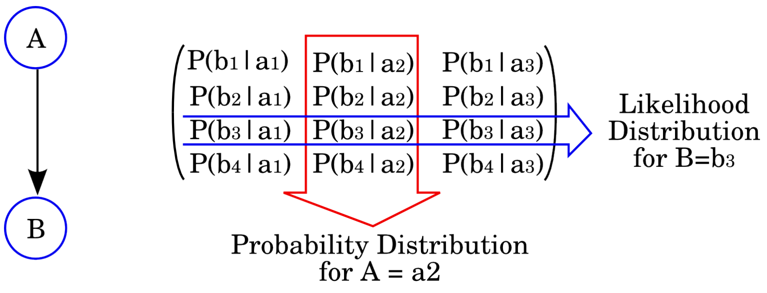
$$log(L(\mu, \sigma|X) = log \prod_{data} exp(-(x-\mu)^2/(2\sigma^2))/\sqrt{(2\pi\sigma^2)} = \sum_{data} [-(x-\mu)^2/(2\sigma^2)) - log\sqrt{(2\pi\sigma^2)}] \quad (6)$$

We find the maximum likelihood values of the parameters (μ, σ) by differentiating the above equation, and setting the result to zero. Analytically this leads to the well known formulae for mean and variance that can be calculated from the data.



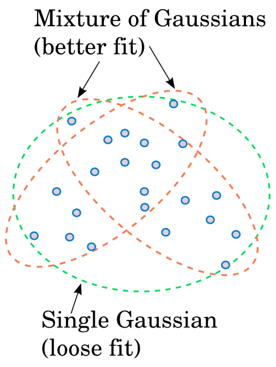
The Probability-Likelihood duality in Bayesian Networks

In Bayesian networks the Probability-Likelihood duality is shown in the conditional probability matrices. The rows are likelihood distributions dependent on a measured value $(B = b_3)$. The columns are a probability distribution based on an hypothesis $(A = a_2)$.



Mixture Models and the EM Algorithm

Gaussian classifiers have class boundaries that are hyper-ellipsoids. The class boundary is the point at which the probability of membership, based on the Gaussian falls below a certina threshold. If the boundary is more complex we may need to define a mixture model. If a mixture is made from a mixture of Gaussians the probability associated with a region of our space will be a weighted average of each Gaussian making up the model. Since the mixture model represents a probability density it must integrate to 1.


Mixture of Gaussians (better fit)
Single Gaussian (loose fit)

Maximum likelihood estimates of the parameters (mean and covariance) of single Gaussian distributions can be calculated directly from the data. Mixture models are data dependent and maximum likelihood estimates can only be found numerically. This means a large optimisation problem which in practice is often solved using the EM Algorithm. Suppose we define a mixture model as follows:

$$p(x|\theta) = \sum_{j=1}^{M} \alpha_j p_j(x|\theta_j)$$

where there are M probability distributions in the mixture and α_i are the mixing weights that have the property $\sum_{j=1}^{M} \alpha_j = 1$.

To fit a mixture model to some data we need, as in the case of the single Gaussian, to find the maximum likelihood estimate of the parameters θ_j of the individual distributions. So, if the individual components of the mixture are multivariate Gaussians, we need to find the values of (μ_j, Σ_j) that maximise the (log) likelihood function:

$$log(L(\theta|x)) = log \prod_{i=1}^{N} p(x_i|\theta) = \sum_{i=1}^{N} log \sum_{j=1}^{M} \alpha_j p_j(x_i|\theta_j) \quad (7)$$

where there are N data points labelled x_i. Unfortunately this equation is very difficult to optimise since there are many possible ways of fitting distributions to the data.

The problem would become simple if we knew which ditribution was responsible for each point. If we think of the data being a set of samples drawn from the mixture, then each data point x_i would have a corresponding index value y_i which is simply the index of the particular distribution from which it was drawn. If this were known then the distributions become independent of each other and their parameters could be found directly. Unfortunately however the values of y_i are unknown.

The EM algorithm is a general algorithm for solving estimation problems of this kind in which there are unknown data. There are two steps:

The E (expectation) step, in which an estimate of the unknowns (y_i values in our problem) is made based on the current estimate of the mixture parameters. This then is used to define the likelihood function.

The M (maximation) step, which, given the likelihood function found above, finds the maximum liklihood estimate of the paramters from the known data.

In the case of the mixture model the outline algorithm is:

1. Choose initial values for the mixture model parameters: $[\alpha_1, \alpha_2, ..\alpha_M, \mu_1, \mu_2, ..\mu_M, \Sigma_1, \Sigma_2, ..\Sigma_M]$ either at random or using some prior information.

2. E Step: For each data point x_i, for each multivariate Gaussian component j, estimate $P(x_i \epsilon j)$, which is the probability that x_i was drawn from the j^{th} distribution. This forms a probability distribution over the possible values of y_i, and is our estimate of the unknown data. If we write the t^{th} step estimate of the j^{th} multivariate Gaussian function as $f(x_i, \mu_j^t, \Sigma_j^t)$ then we can write:

$$P(x_i \epsilon j) = P(y_i = j | x_i, \mu_j, \Sigma_j) = \frac{\alpha_j^t f(x_i, \mu_j^t, \Sigma_j^t)}{\sum_{k=1}^{M} \alpha_k^t f(x_i, \mu_k^t, \Sigma_k^t)} \tag{8}$$

The expectation function, which is the likelihood function that we wish to maximise, can now be written as:

$$Q(\theta, \theta^t) = \sum_{data} logL(x, y, \theta) = \sum_{i=1}^{N} \sum_{j=1}^{M} P(x_i \epsilon j) log(\alpha_j f(x_i, \mu_j, \Sigma_j)) \tag{9}$$

3. M Step: Find the maximum liklihood estimate of the mixture model parameters. Maximising this expression is now straightforward since it has a linear form. The optimum for the α values can be found using:

$$\alpha_j^{t+1} = \frac{\sum_{data} P(x_i \epsilon j)}{\sum_{data} \sum_{j=1}^{M} P(x_i \epsilon j)} = \frac{1}{N} \sum_{data} P(x_i \epsilon j) \tag{10}$$

Each mixture component can be maximised individually since they are combined linearly. This is done using a weighted version of the standard formulae.

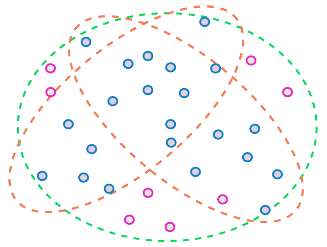
$$\mu_k^{t+1} = \frac{\sum_{i=1}^{N} P(x_i \epsilon k) x_i}{\sum_{i=1}^{N} P(x_i \epsilon k)} \tag{11}$$

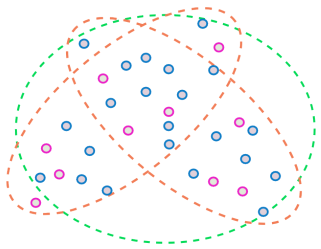
$$\Sigma_k^{t+1} = \frac{\sum_{i=1}^{N} P(x_i \epsilon k)(x_i - \mu_k^{t+1})^T (x_i - \mu_k^{t+1})}{\sum_{i=1}^{N} P(x_i \epsilon k))} \tag{12}$$

4. repeat 2 and 3 until convergence.

Overfitting

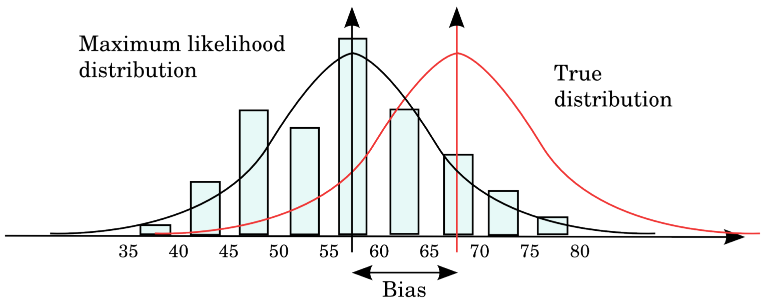
If our classifier is trained on a small data set it is inadvisable to build too complex a classification boundary.


New data suggesting overfitting by the mixture model


New data supporting the mixture model-

Bias

Parameters of a distribution are normally computed from a data set. The difference between a true mean and an estimated mean is termed bias.



Regression Models

Most of our examples have considered classification where the state of a hypothesis variable Y is predicted by a set of random variables $\boldsymbol{X} = [X_1, X_2, ..X_n]$. Regression models estimate a numeric value for Y from the values of discrete or continuous variables. For example the linear regression model is:

$$Y \sim N(E(Y|\boldsymbol{X}), \sigma^2) = N(B_0 + \boldsymbol{B}.\boldsymbol{X}, \sigma^2) \tag{13}$$

where $\boldsymbol{B}$ is a set of constants which are estimated from data. The regression model need not be linear, it can be any function of X. Higher order models may fit the data better. Consider the case of a single variable X. We have:

$$\begin{array}{rl} \text{Linear} & E(Y|X) = B_0 + B_1X \\ \text{Quadratic} & E(Y|X) = B_0 + B_1X + B_2X^2 \\ \text{Cubic} & E(Y|X) = B_0 + B_1X + B_2X^2 + B_3X^3 \end{array}$$

The higher order models encapsulate the lower order models, and therefore fit the data better. However higher order models are susceptible to overfitting if the data is not sufficiently representative.

Prediction Error

Suppose we use an estimation method to calculate the parameters of a regression model $\boldsymbol{B} = [B_0, B_1, ...B_n]$. We write: $Y = f(\boldsymbol{X}|T)$ where T is the training data set. We can distinguish two types of error, if the correct estimate of Y is $f(\boldsymbol{X})$ then we can define the reducible prediction error as: $(f(\boldsymbol{X}) - f(\boldsymbol{X}|T))^2$

Prediction error is divided into reducible and irreducible parts. The reducible part of the error is only reducible by finding a better model. The irreducible error is caused by the natural variation in Y.

$$\text{Irreducible error} = E[(y - f(\boldsymbol{X}))^2]$$

where E stands for the expected value (mean), and $f(\boldsymbol{X})$ is (as before) the true estimate of Y.

Nothing can be done about irreducible error which is data dependent. We can further divide the reducible error to:

	$(f(\boldsymbol{X}) - E[f(\boldsymbol{X}|T)])^2$ - The error due to bias
and	$E[(f(\boldsymbol{X}|T) - E[f(\boldsymbol{X}|T)])^2]$ - The error due to variance
where	$E[f(\boldsymbol{X}|T)]$ is the mean over the training set

Trade off between bias and variance

Statistically it turns out that there is a trade off between bias and variance: low order models (eg linear) tend to be biased, whereaas high order models have greater variance. For large training sets variance tends to decrease for a fixed bias, hence higher order models may be more appropriate. We also can note that the re-sampling techniques, Boosting and Bagging, discussed in the last lecture will reduce the variance of the result. Boosting is also claimed to reduce bias, though the degree to which this happens is highly data dependent.