Lecture 13:

Data Modelling and Distributions

Why data distributions?

It is a well established fact that many naturally occurring data sets have values distributed normally.

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

If we can find the correct distribution and the parameters we have a compact model that we can use for classification or prediction.

Distributions can also be used in mathematical proofs.

Uniform Distributions

Distributions are usually constructed to represent probability, so the area under them integrates to 1.

The numeric ranges of the data are set by the parameters:

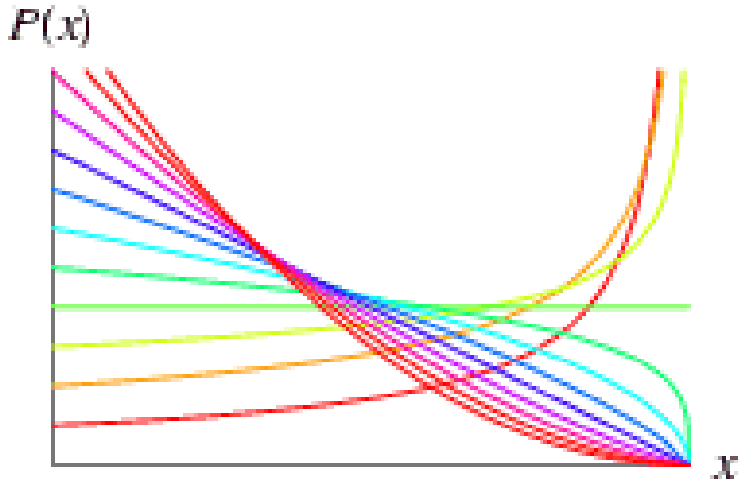
eg Uniform distribution:

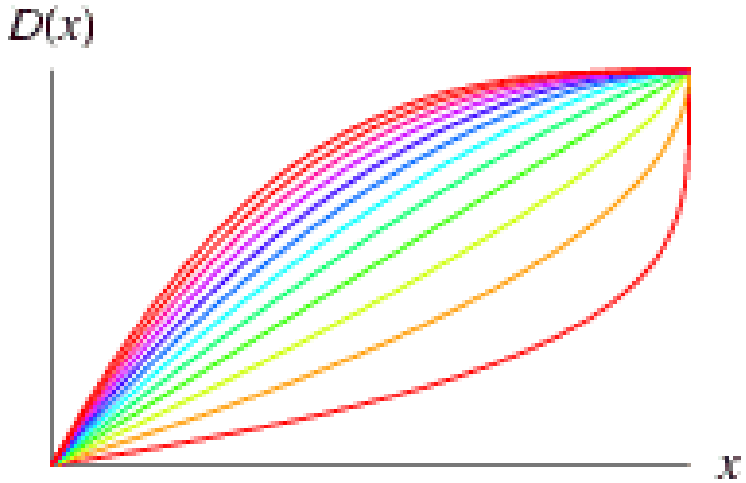
$$p(y) = \begin{cases} 1/(\beta - \alpha) & \text{for} \alpha < y < \beta \\ 0 & \text{otherwise} \end{cases}$$

Other Distributions

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/)





Cumulative form

The Multi-dimensional Gaussian distribution

Multi-variate joint normal (Gaussian) distribution.

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

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

The covariance matrix is symmetric with variances on the diagonal:

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

Covariance is defined as follows:

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

The covariance matrix

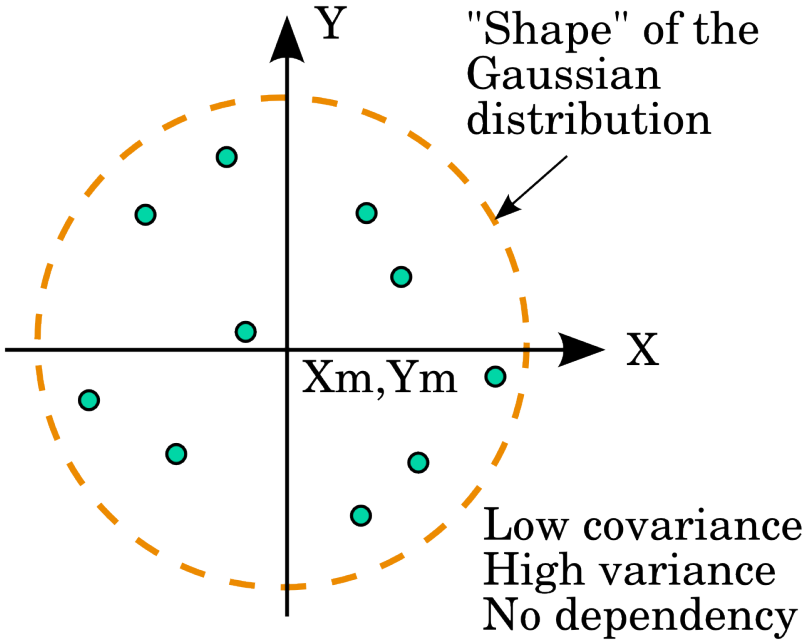
Diagonal elements are individual variances,

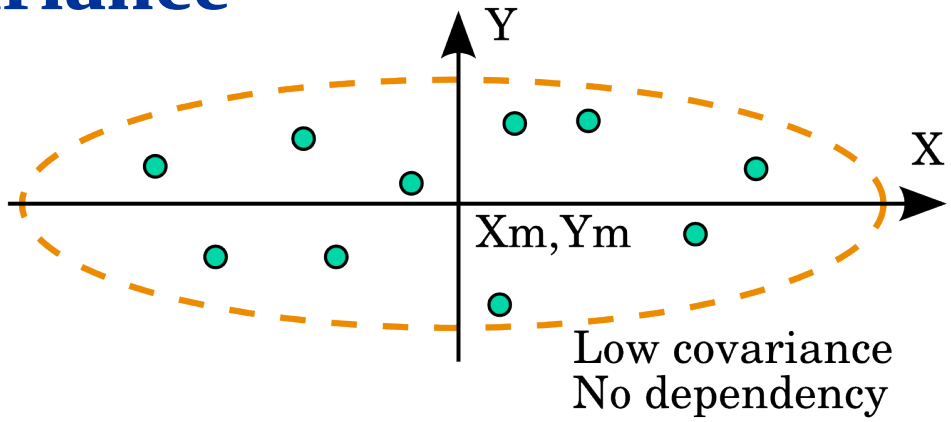
Off diagonal elements are co-variances indicating data dependencies.

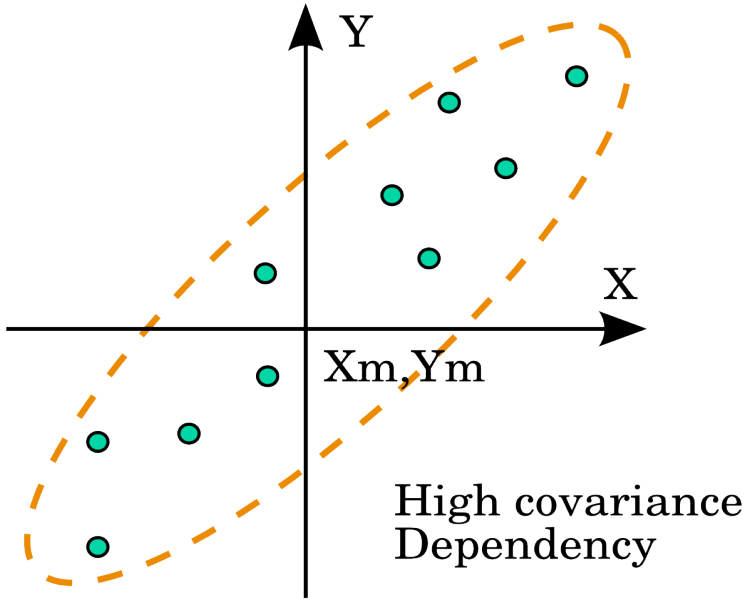
Recall that we used the correlation coefficient as a dependency measure:

$$Dep(x, y) = Corr(x, y) = \sigma_{x,y} / \sqrt{(\sigma_{xx} \sigma_{yy})}$$

2D Illustration of covariance

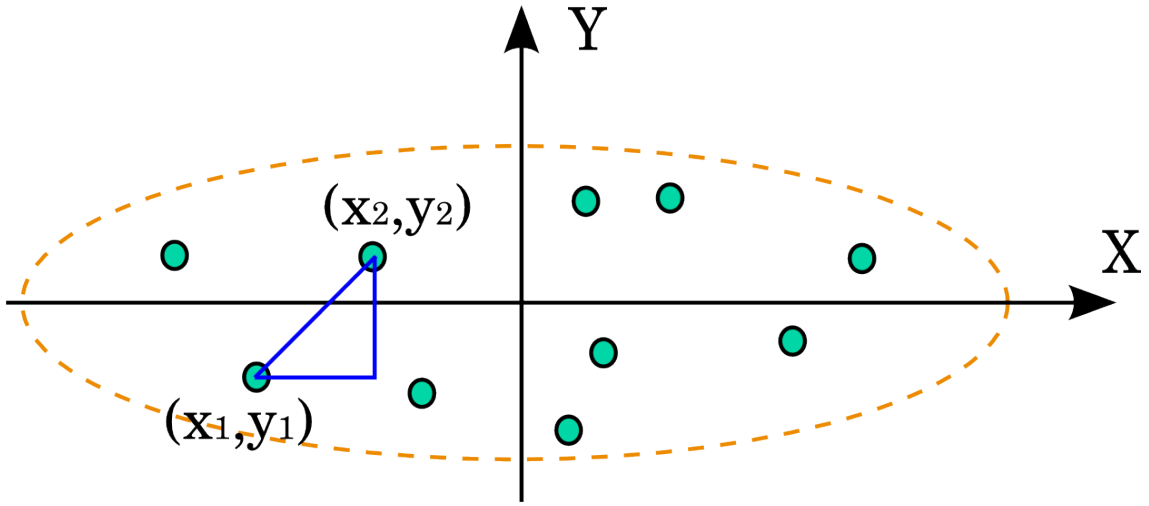






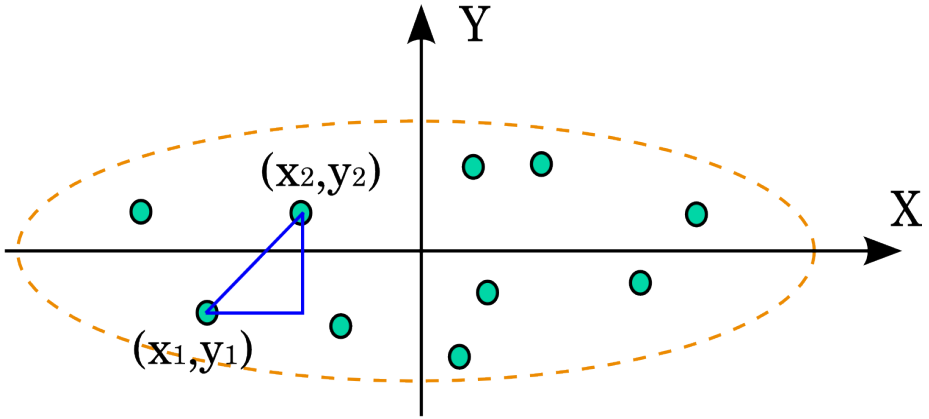
Mahalanobis Distance

Euclidian distance weights all dimensions (variables) equally, however, statistically they may not be the same:



The Euclidian distance tells us that $(x_2-x_1) = (y_2-y_1)$, however statistically $(x_2-x_1) < (y_2-y_1)$.

Mahalanobis Distance



It is easy to see that for low (or zero) covariance we can can normalise the distances by dividing by the variance:

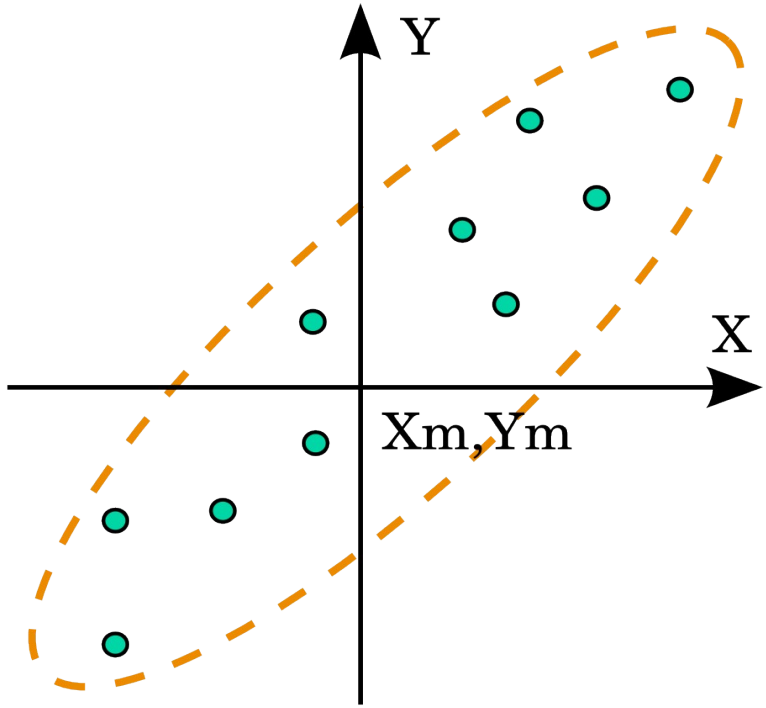
$$\Delta' x = (x_2 - x_1)/\sqrt{\sigma_{xx}}$$
$$\Delta' y = (y_2 - y_1)/\sqrt{\sigma_{yy}}$$
$$\text{Mahalanobis Distance} = \sqrt{(\Delta' x^2 + \Delta' y^2)}$$

In general the Mahalanobis distance between two points can be written as:

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

Mahalanobis Distance

The Mahalanobis distance also works for high co-variance.



For the 2D case the inverse of the covariance matrix is:

$$\Sigma^{-1} = \quad 1/(\sigma_{xx}\sigma_{yy} - \sigma_{xy}^{2}) \begin{pmatrix} \sigma_{yy} & -\sigma_{xy} \\ -\sigma_{xy} & \sigma_{xx} \end{pmatrix}$$

Likelihood and Probability

Likelihood and Probability are dual concepts:

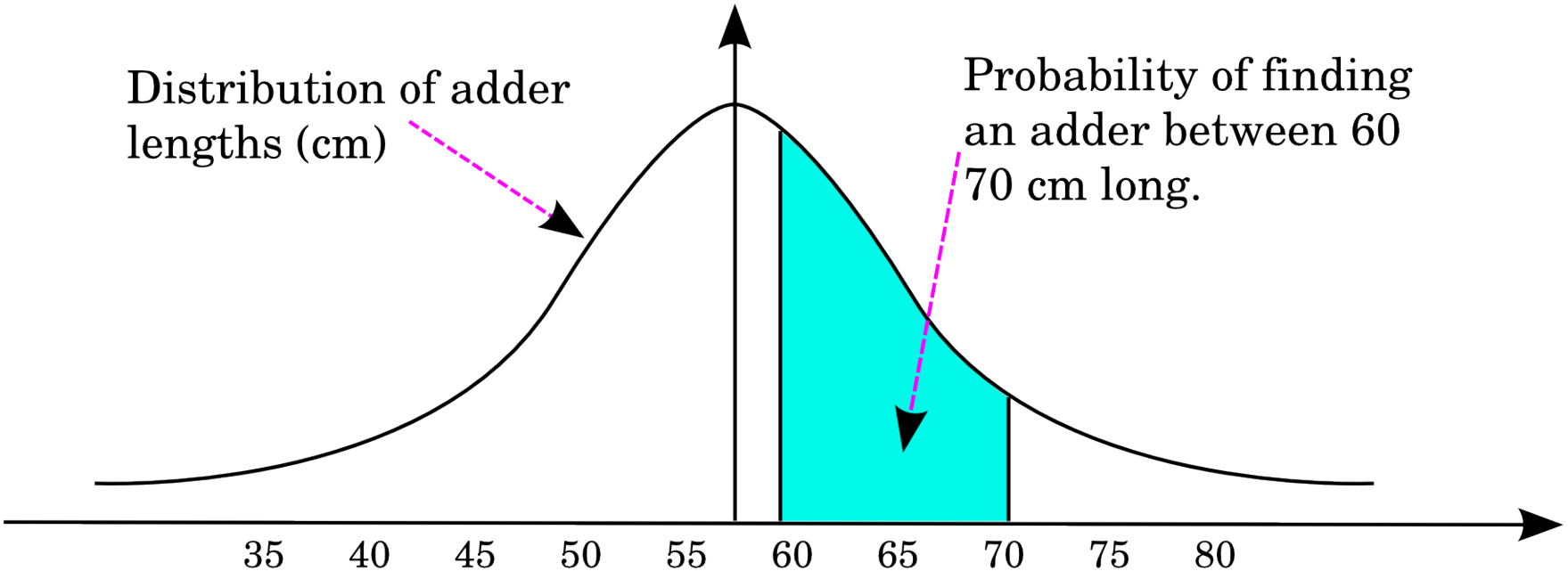
Given a distribution we can work out the probability of a point.

Given a set of data points we work out the likelihood of a distribution

Probability

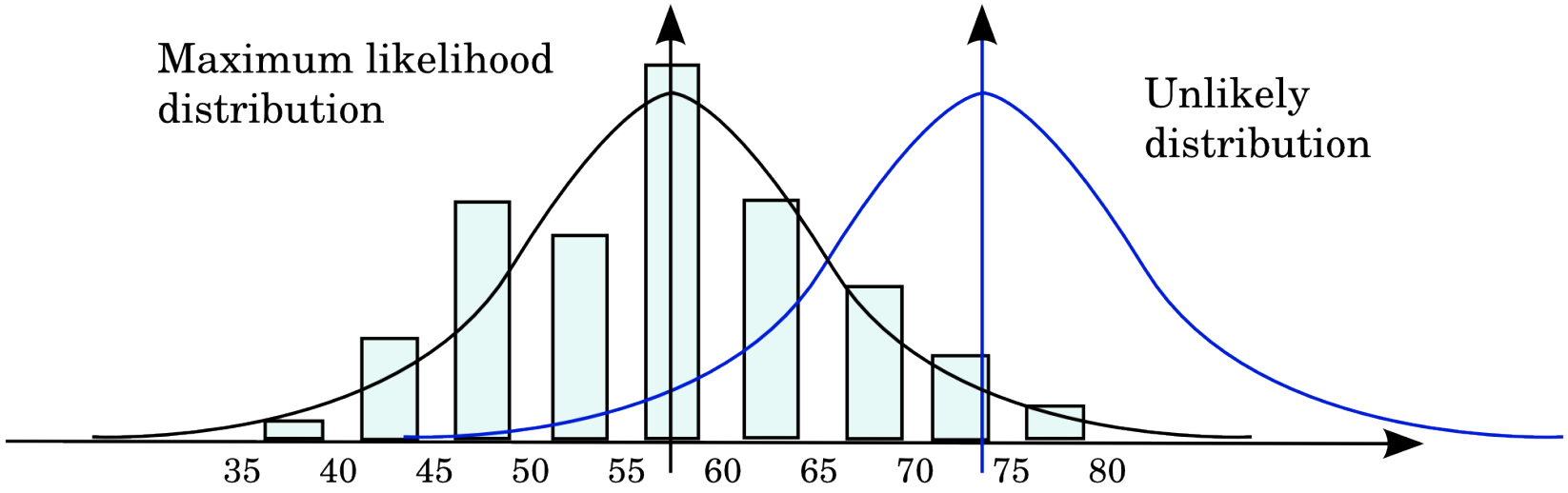
Given a distribution we can calculate a probability,

eg: What is the probability of finding an adder between 65 and 75 cm long?



Likelihood

Given a data set we can ask:
"How likely is a model"



The likelihood of a distribution is the product of the probability of each data point in the distribution - likelihood is not probability.

Maximum Likelihood

The maximum likelihood can be found for a given distribution by differentiating the likelihood function with respect to the parameters (eg mean and variance) and setting the result to zero.

For the Gaussian this produces the well known formulae for μ and Σ, which, when calculated from a data set define the maximum likelihood distribution given that data.

Log Likelihood

Computationally is is often more convenient to find the log likelihood (since the product will underflow for large data sets):

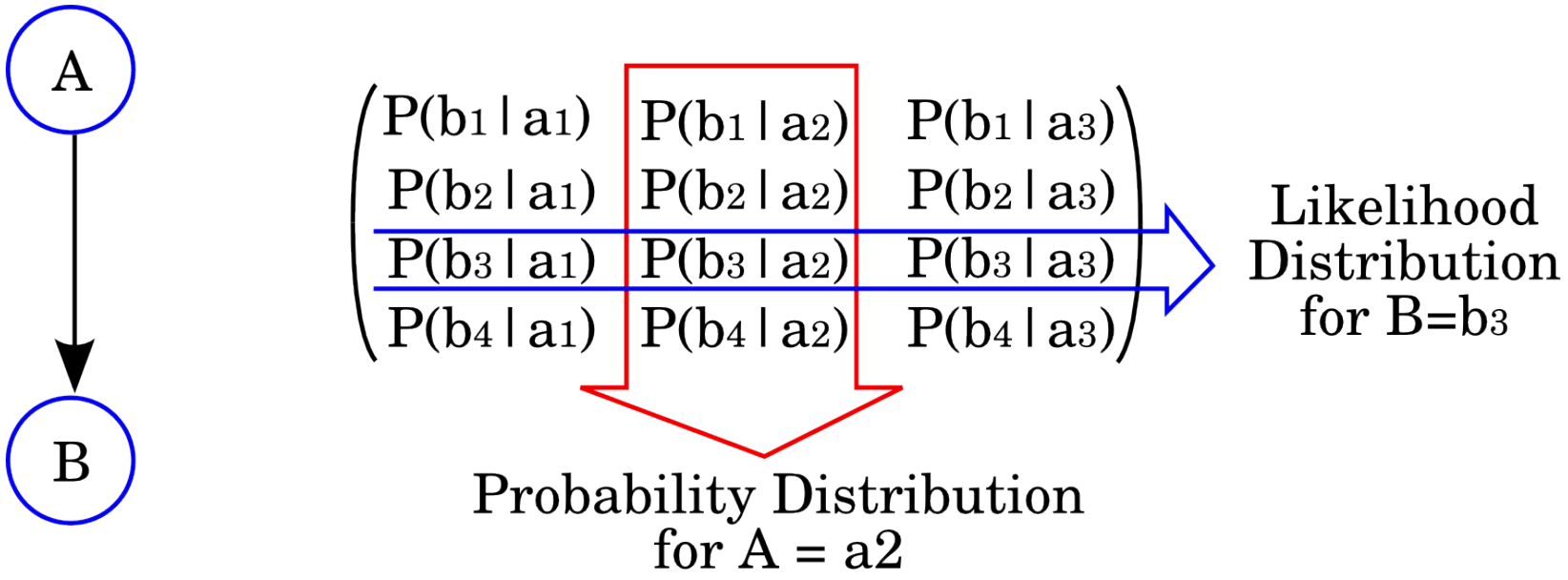
$$log(L(\mu, \sigma|X) = log \prod_{data} exp(-(x-\mu)^2/(2\sigma^2))/\sqrt{(2\pi\sigma^2)}$$

$$= \sum_{data} [-(x-\mu)^2/(2\sigma^2)) - log\sqrt{(2\pi\sigma^2)}]$$

The maximum likelihood distribution can also be found by maximising the log likelihood.

Probability/Likelihood duality in BNs

In Bayesian networks the Probability-Likelihood duality is shown in the conditional probability matrices



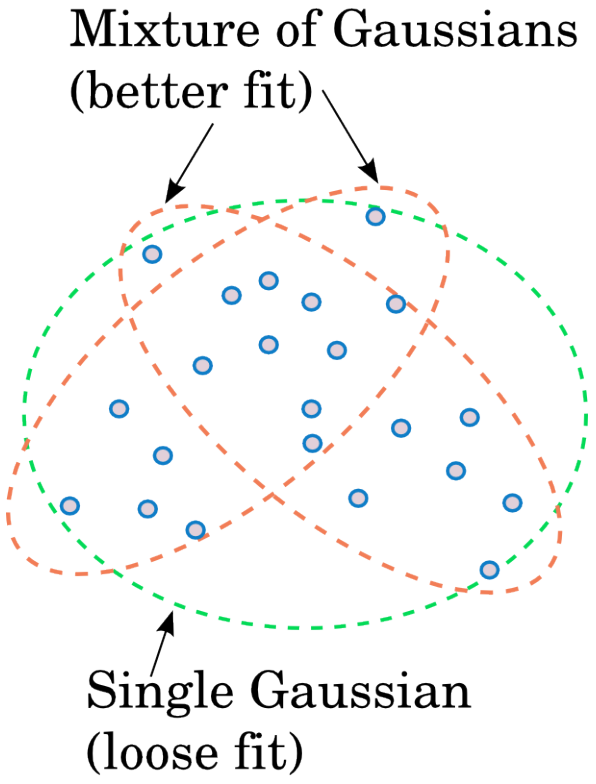
Mixture Models

Gaussian classifiers have class boundaries that are hyper-ellipsoids. If the boundary is more complex we may need a mixture model.

A mixture of M Gaussians has pdf:

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

where $\Sigma_M \alpha = 1$

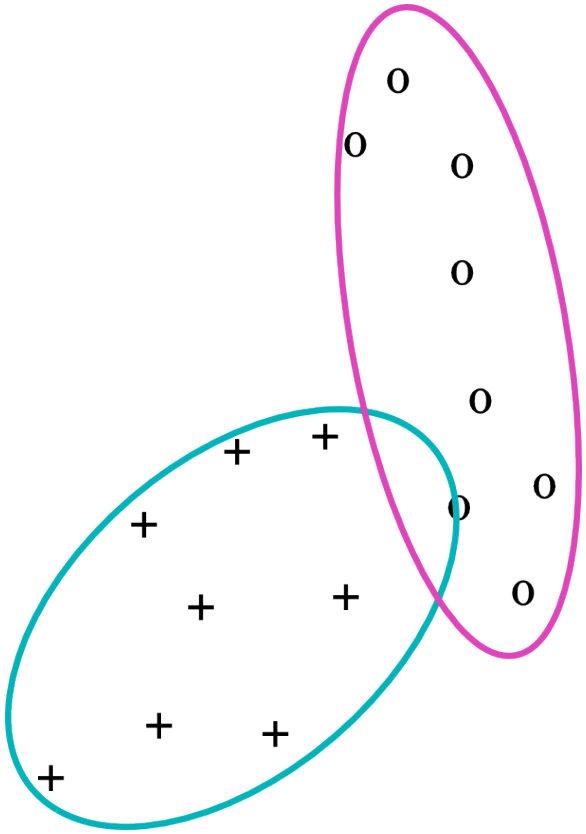


Problems with Mixture Models

Maximum likelihood estimates of single distributions can be calculated directly from the data. (mean and covariance).

Mixture models are data dependent, and maximum likelihood estimates can only be found numerically. This means a large optimisation problem.

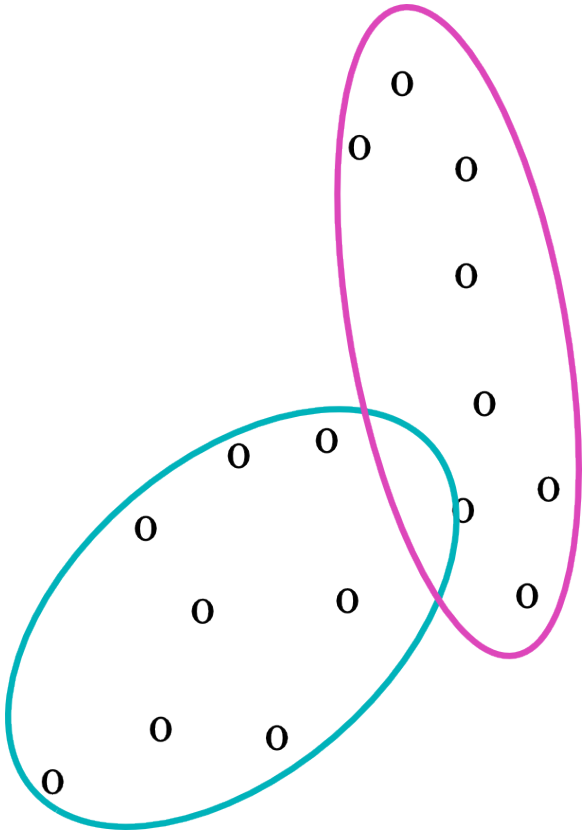
Problems in estimating mixture models



If we knew which points belong to which class it would be easy to fit a mixture distribution. Using the standard formulae.

However this is rarely the case

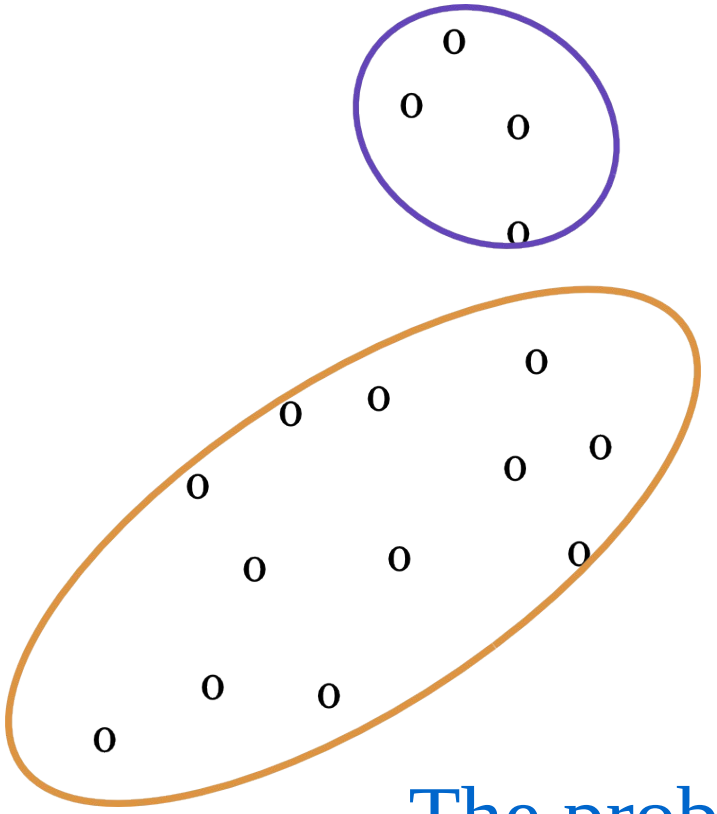
Problems in estimating mixture models



In general we don't know which points belong to which class, or even how many classes there are.

There may be many possible mixture models

Problems in estimating mixture models



In general we don't know which points belong to which class, or even how many classes there are.

There may be many possible mixture models

The problems are normally addressed using the EM Algorithm

The Expectation Maximisation (EM) Algorithm

As before, let us define a mixture model as:

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

where the individual distributions are all Gaussians. Our problem is to choose the parameters (α_j, μ_j Σ_j) that maximise the log likelihood:

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

Where there are *N* data points.

The EM Algorithm - the E step

The EM algorithm starts by choosing a mixture model at random.

Suppose each point has a class label y_i which indicates which component of the mixture model it belongs to.

In the EM algorithm the y_i values are unknown. They are estimated from the current mixture parameters.

This is called the Expectation (E) step

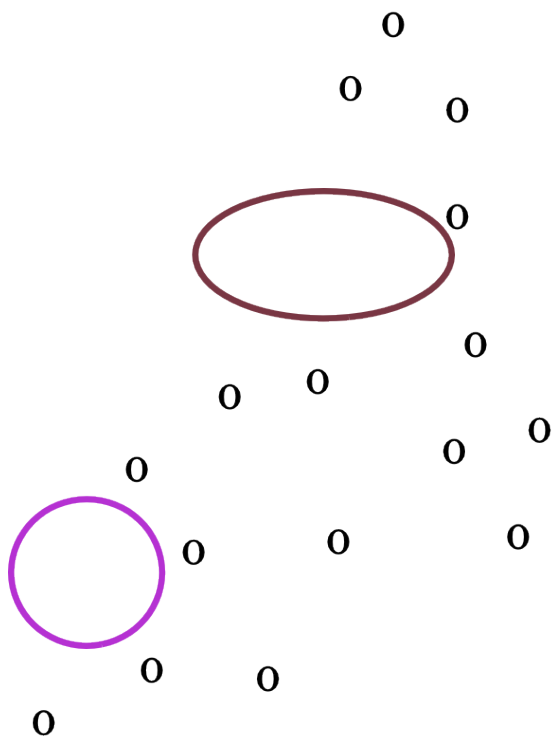
The EM Algorithm - The M step

In the M step we take the current estimate of which point belongs to which class and use it to find the means and covariances of each class.

This can be done analytically.

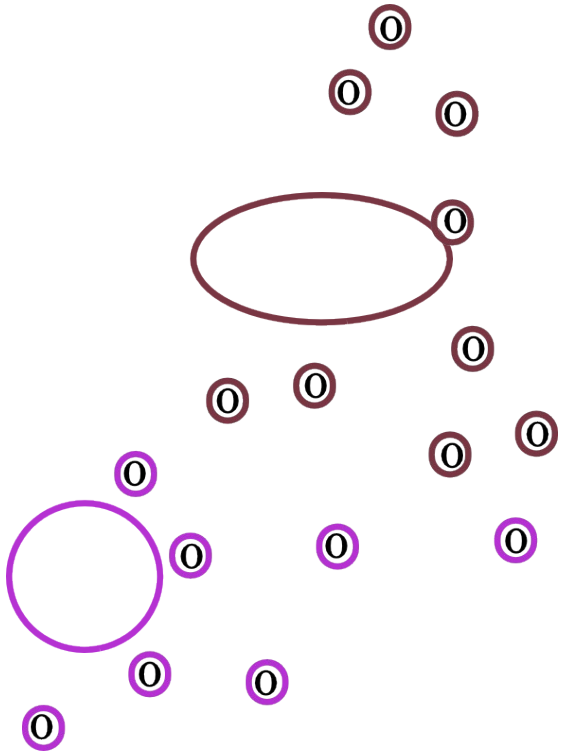
The E and M steps are repeated until convergence.

Illustration of the EM Algorithm



The first step is to take a random guess at the class means and covariances.

Illustration of the EM algorithm



The probability that each point belongs to each class is then found. The diagram illustrates the most likely class each point belongs to.

This is the expectation (E) step.

The EM algorithm - the E step

Given that the t^{th} step estimate of the j^{th} Gaussian in the mixture is written $f(x_i, \mu_j^t, \Sigma_j^t)$, we can calculate the probability that point x_i belongs to class j as:

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

We could allocate each point to its most probable class, but it is better to retain all the membership probabilities.

The outcome of the E step

The E step uses the values of P($x_i \in j$) to create an expectation function. This in essence is an estimate of the log likelihood function from which we can calculate the maximum likelihood estimate of the parameters of the distributions in the mixture.

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

The maximum likelihood estimates can be found analytically.

Calculating the M step

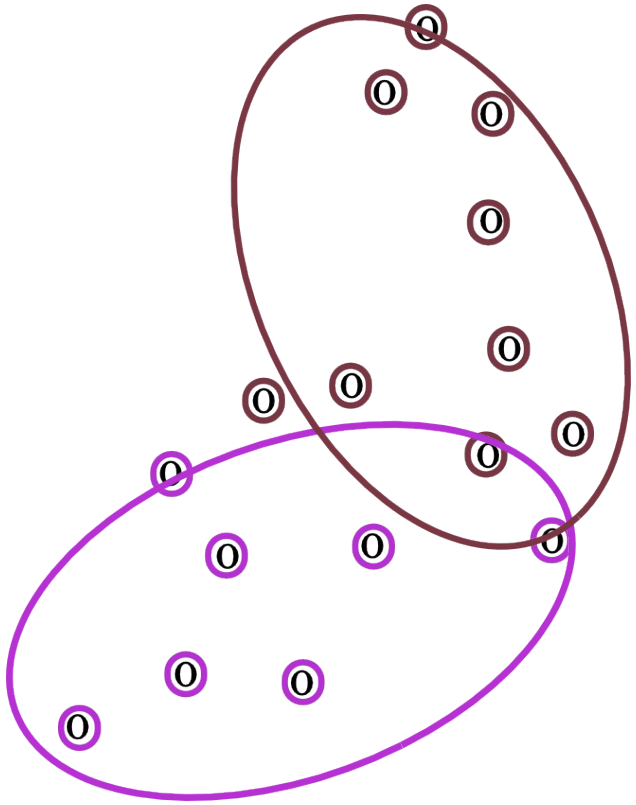
The M step makes the maximum likelihood extimate using the following formulae:

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

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

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

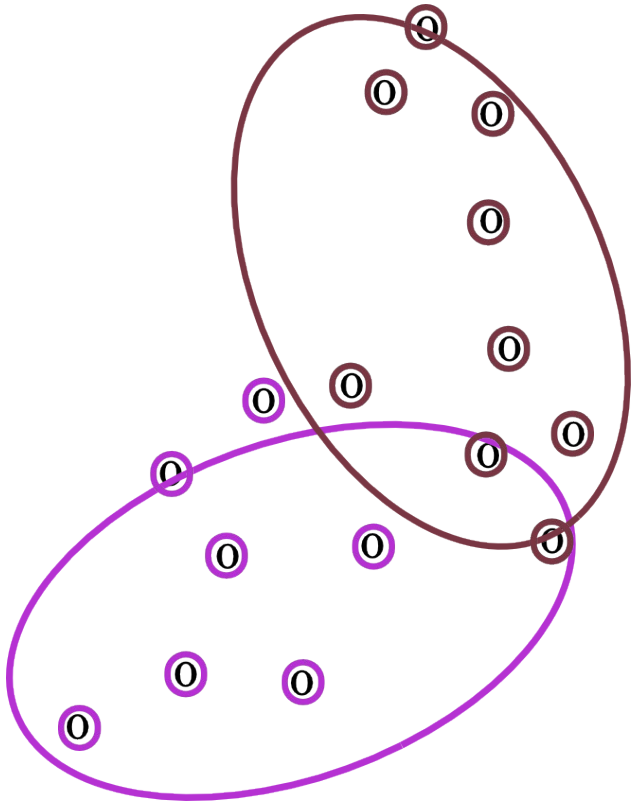
Illustration of the EM algorithm



The class means and covariances are calculated using the probabilities of class membership to weight the formulae

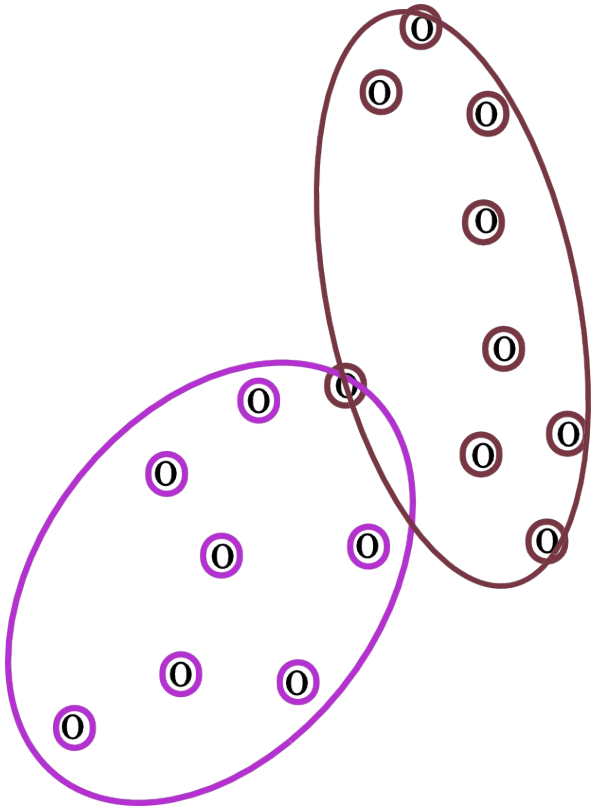
This is the maximisation (M) step

Illustration of the EM Algorithm



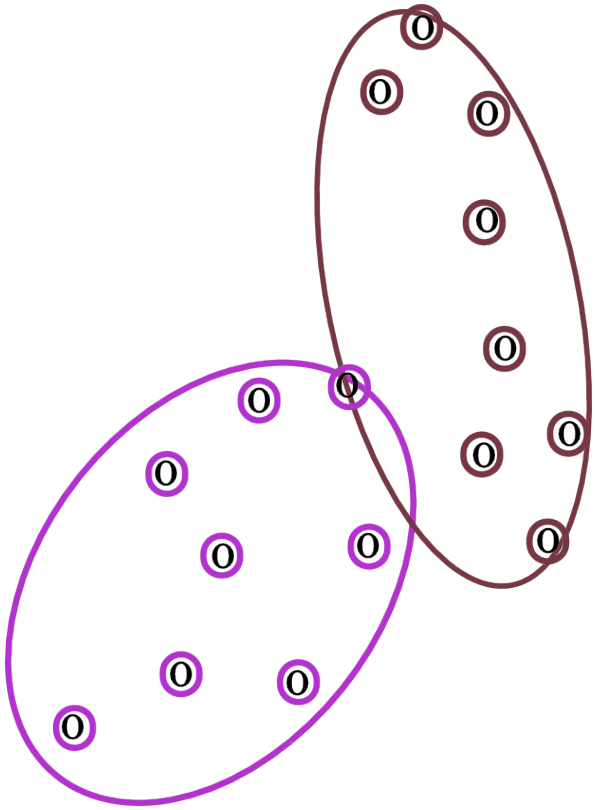
We carry out the E step again. The probabilities of class membership for each point are updated.

Illustration of the EM Algorithm



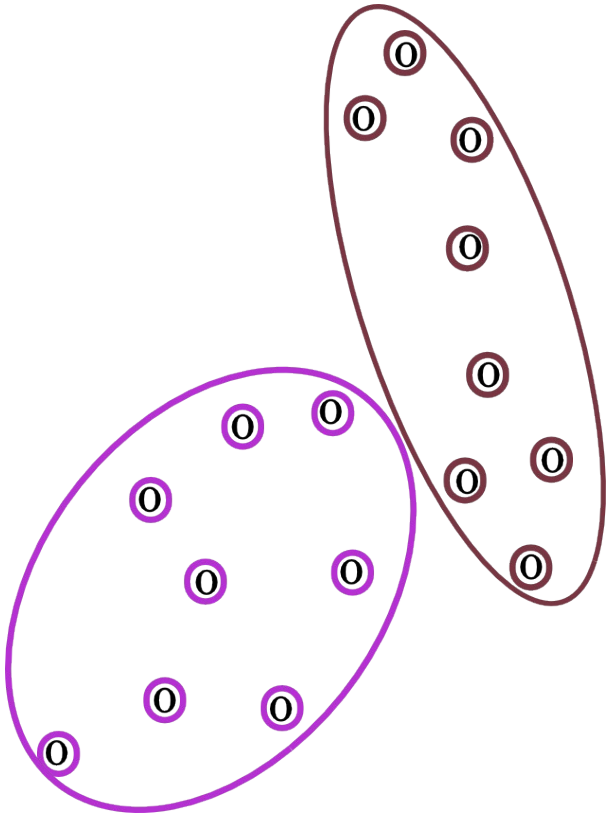
The M step is then repeated, and the process continues.

Illustration of the EM algorithm



The membership probabilities begin to converge

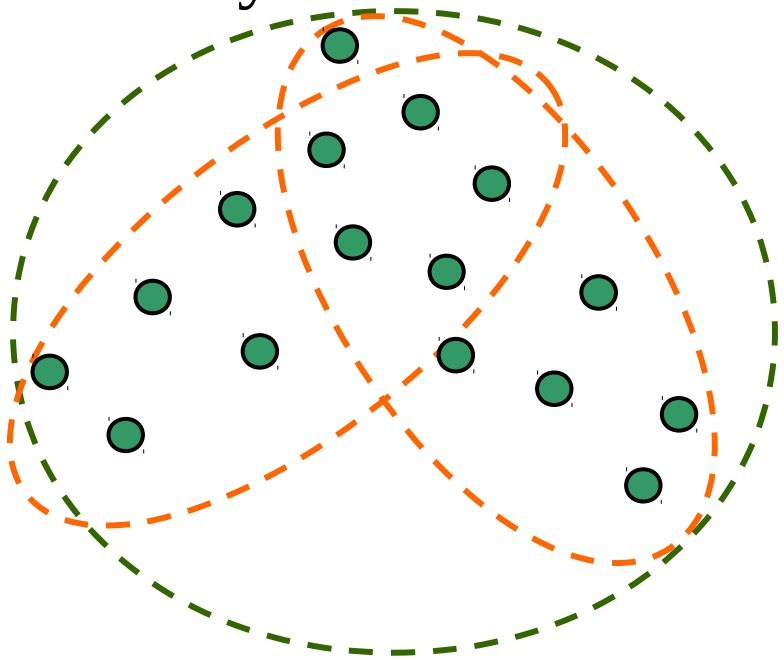
Illustration of the EM Algorithm



Finally the algorithm has converged to a stable state and terminates.

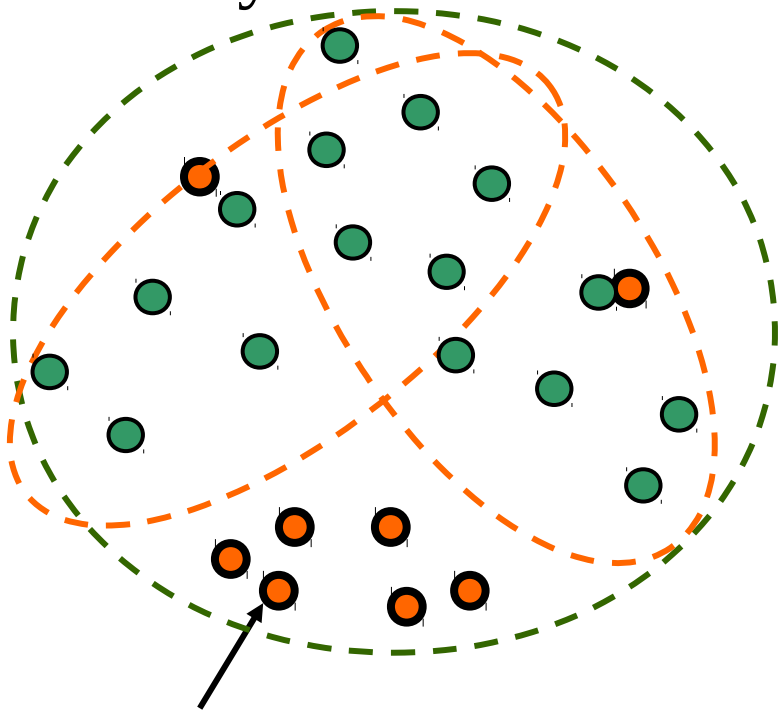
Overfitting

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



Overfitting

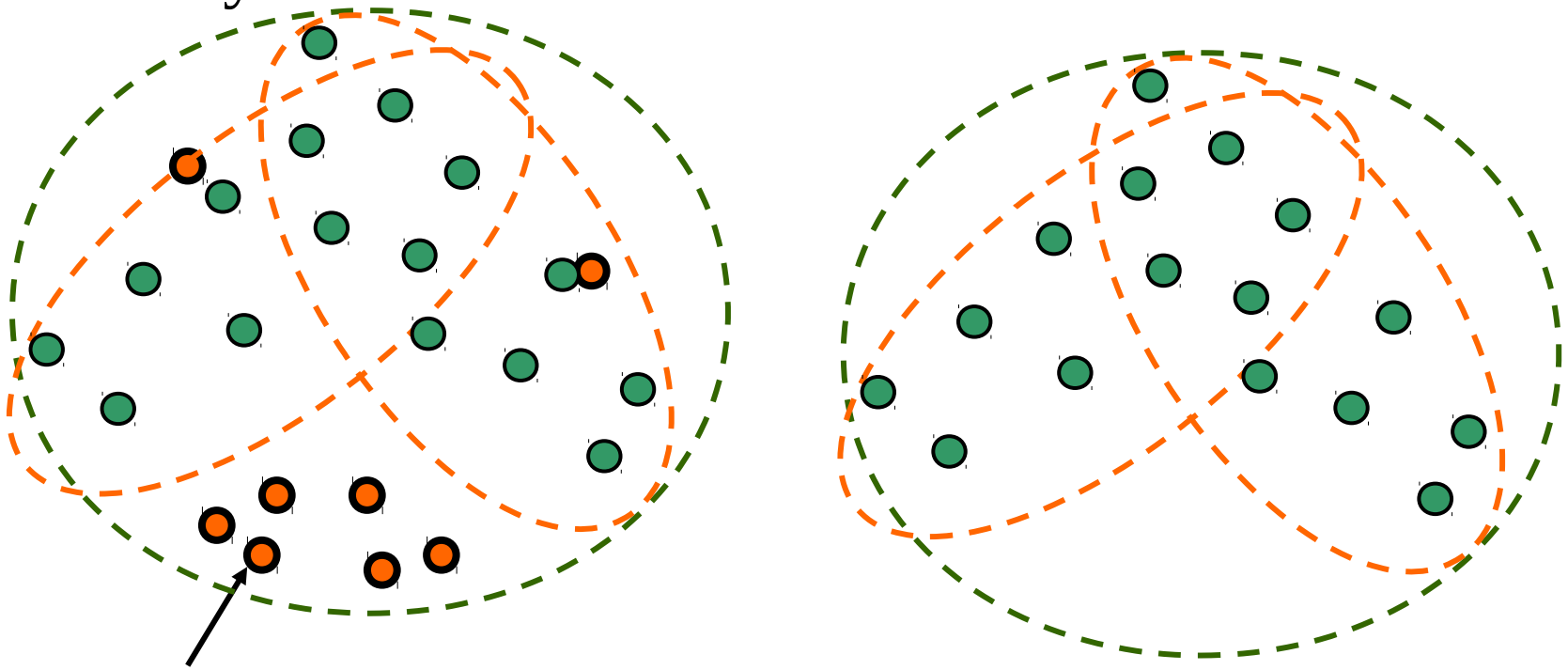
If our classifier is trained on a small data set it is inadvisable to build too complex a classification boundary



More data in favour of a loose boundary implies overfitting

Overfitting

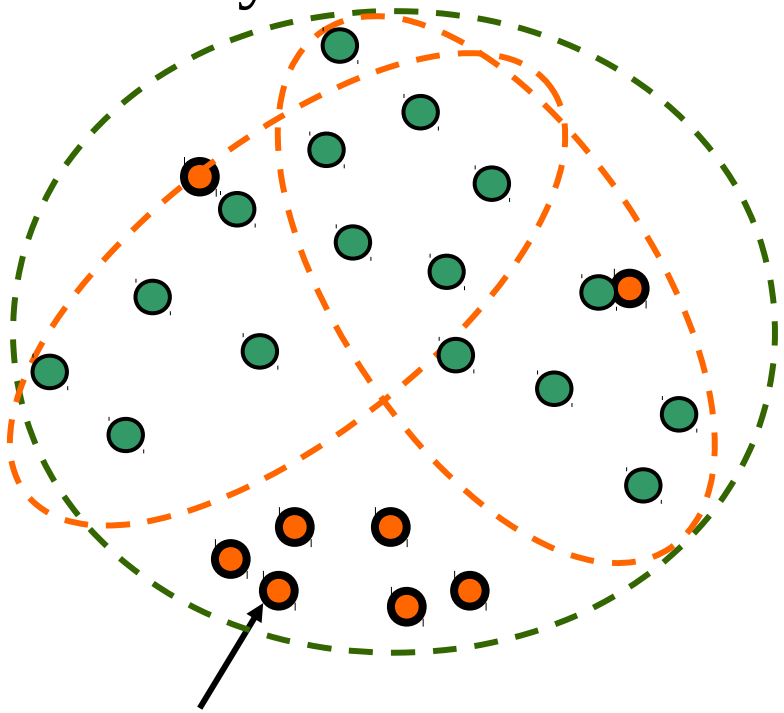
If our classifier is trained on a small data set it is inadvisable to build too complex a classification boundary



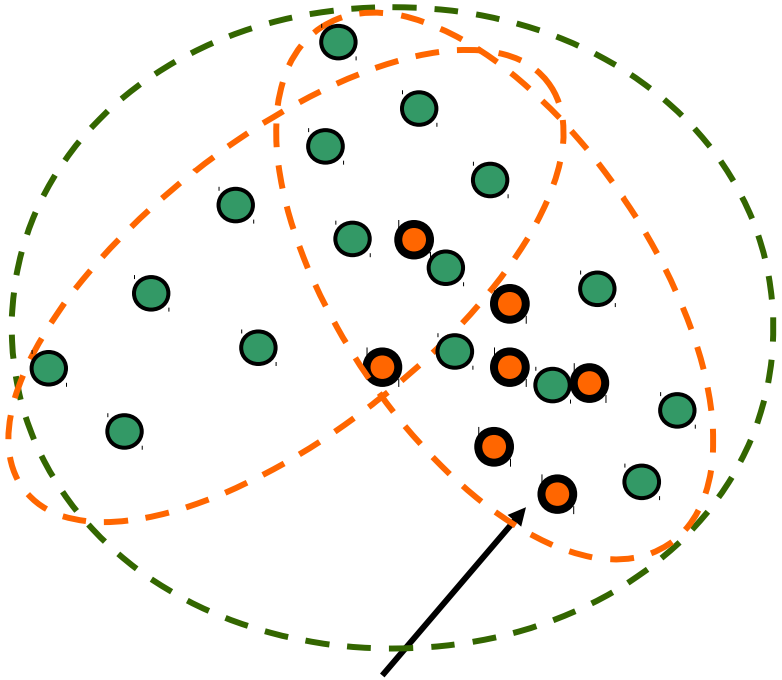
More data in favour of a loose boundary implies overfitting

Overfitting

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



More data in favour of a loose boundary implies overfitting

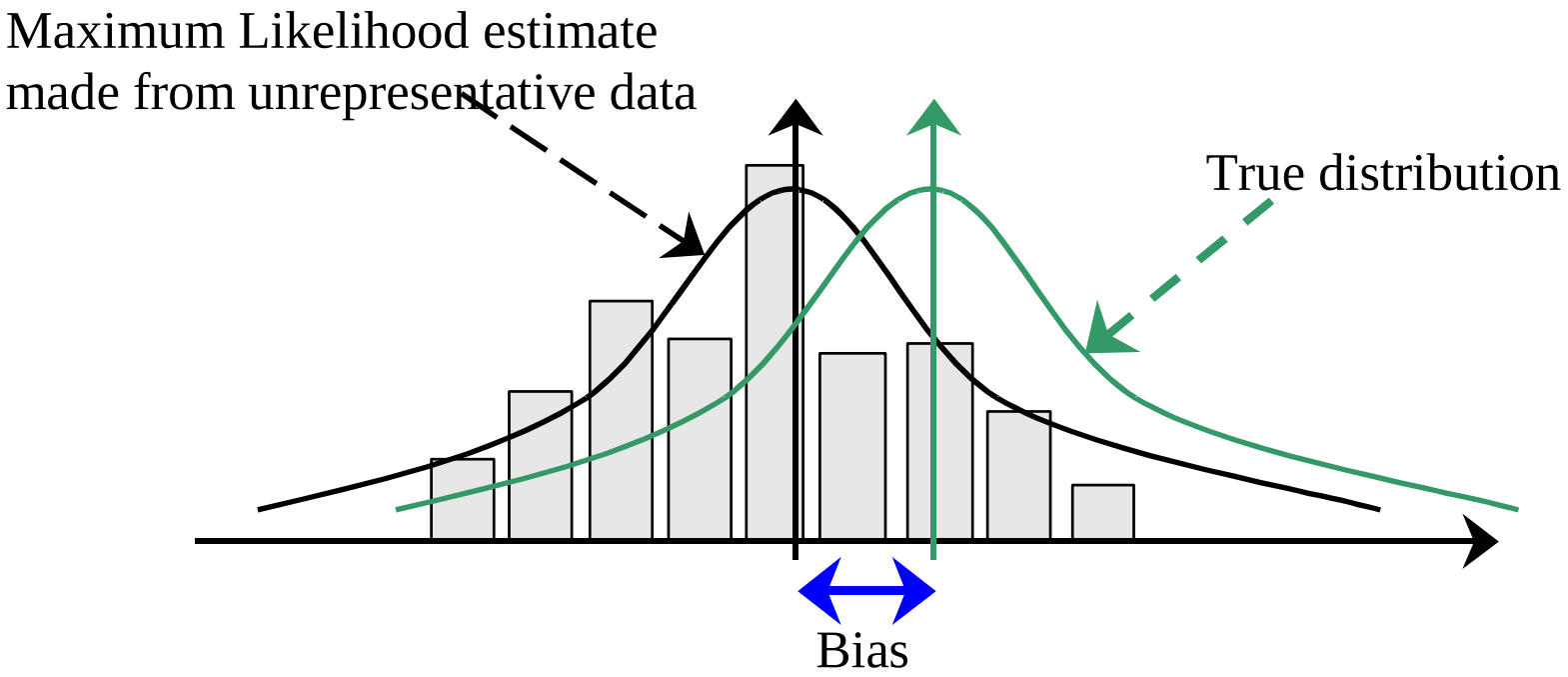


More data could favour the mixture

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

In most of our classification examples the state of a hypothesis variable Y is predicted by a set of random variables $\mathbf{X} = \{X1, X2, \ldots Xn\}$

Regression models estimate a numeric value for Y from the values of discrete or continuous variables.

eg $Y \sim N(B_0 + \mathbf{B.X}, \sigma^2)$ - linear regression model

we can write this as $Y \sim N(E(Y|\mathbf{X}), \sigma^2)$

Regression Models

The regression model need not be linear, it can be any function of X.

Higher order models may fit the data better: Consider:

Linear $E(Y|X) = B_0 + B_1X$

Quadratic $E(Y|X) = B_0 + B_1X + B_2X^2$

Cubic $E(Y|X) = B_0 + B_1X + B_2X^2 + B_3X^3$

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 $B = \{B_0, B_1, \ldots B_n\}$

we write $Y = f(\mathbf{X}|T)$ where T is the training data set.

We can distinguish two types of error, if the correct estimate of $Y = f(X)$ then we can define the reducible prediction error as:

$$(f(\mathbf{X}) - f(\mathbf{X}|T))^2$$

Irreducible Error

The reducible part of the error is only reducible by finding a better model. (or more representative data)

There is also irreducible error which is caused by the natural variation in y.

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

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

Nothing can be done about irreducible error

Reducible Error

We can further divide the reducible error to:

$(f(\mathbf{X}) - E[f(\mathbf{X}|T)])^2$ - The error due to bias

and

$E[(f(\mathbf{X}|T) - E[f(\mathbf{X}|T)])^2]$ - The error due to variance

where $E[f(\mathbf{X}|T)]$ 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

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.

Bias and variance in re-sampling

As noted previously, Bagging and Boosting are both found to reduce the variance component of prediction error in simulation studies.

Boosting is also claimed to reduce bias, though the degree to which this happens is highly data dependent.