Machine Learning Module

Week 6

Lecture Notes 11 & 12

Probability Density Estimation

Mark Girolami
girolami@dcs.gla.ac.uk
Department of Computing Science
University of Glasgow

February 14, 2006

1 Probability Density Estimation

In Week 4 we saw that devising a classifier based on the discriminative principle required an estimate of the class-conditional probabilities $p(\mathbf{x}|C = k)$ for each class. The two examples of class-conditional probabilities we considered were the multivariate Gaussian which produced a quadratic discriminant function and the Naive Bayes classifier based on Bernoulli probabilities for the binary variables representing the presence or absence of words in a document. Now in both of these cases we have assumed that the class-conditional distributions are of a specific parametric form i.e. multivariate Gaussian and multiple independent Bernoulli distributions. Each of these have a set of parameters which, given our data which is available for training, has to be estimated in some way. This then leads us onto the first and most straightforward method of Probability Density (or distribution if we are dealing with discrete random variables) which is referred to as the Parametric form of density estimation.

If, given our data, we are justified in making specific assumptions regarding the parametric form of the distribution of our available features then our task simply requires us to obtain estimates of the corresponding parameters of the distribution.

2 Parametric Probability Density Estimation

As the class-conditional probability $p(\mathbf{x}|C = k)$ will take on a specific parametric form with a set of parameters $\boldsymbol{\theta}_k$ with the subscript k denoting that these parameters are associated with class k we can write $p(\mathbf{x}|\boldsymbol{\theta}_k)$ to denote our class-conditional probability density. So how do we obtain estimates for each set of $\boldsymbol{\theta}_k$? In Week 2 we were introduced to the Maximum Likelihood (ML) Principle and it is exactly this that we employ in obtaining our parameter estimates.

2.1 Maximum Likelihood Estimation

A couple of examples will illustrate the method of parameter estimation based on ML.

2.1.1 Multivariate Gaussian

If we have N_k examples from class k in our training data and we have sufficient justification to assume that the D features we have are distributed as a Multivariate Gaussian then the likelihood under this assumption is given as

$$\begin{aligned}\mathcal{L}_k = \prod_{n=1}^{N_k} p(\mathbf{x}_n|\boldsymbol{\theta}_k) &= \prod_{n=1}^{N_k} p(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ &= \prod_{n=1}^{N_k} \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}_k|}} \exp\left\{-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)\right\}\end{aligned}$$

and as before we can work with the logarithm of the likelihood in which case

$$\log \mathcal{L}_k = -\frac{N_k D}{2}\log 2\pi - \frac{N}{2}\log|\boldsymbol{\Sigma}_k| - \frac{1}{2}\sum_{n=1}^{N_k}(\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

as before we require to obtain the stationary points of the likelihood and solve for our parameters in which case we take derivatives with respect to $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$.

Now we can expand the quadratic term on the right hand side of the above expression and drop all terms which are not functions of $\boldsymbol{\mu}_k$ in which case

$$\frac{\partial}{\partial \boldsymbol{\mu}_k}\log \mathcal{L}_k = \frac{\partial}{\partial \boldsymbol{\mu}_k}\left(\frac{1}{2}\sum_{n=1}^{N_k}\left\{2\boldsymbol{\mu}_k^\mathsf{T}\boldsymbol{\Sigma}_k^{-1}\mathbf{x}_n - \boldsymbol{\mu}_k^\mathsf{T}\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k\right\}\right)$$

Now remembering the vector derivatives from Week 2 (refer to the Matrix Cookbook) we can obtain the required derivative as

$$\frac{\partial}{\partial \boldsymbol{\mu}_k}\log \mathcal{L}_k = \sum_{n=1}^{N_k}\left\{\boldsymbol{\Sigma}_k^{-1}\mathbf{x}_n - \boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k\right\}$$

Setting the gradient to zero we then obtain

$$\sum_{n=1}^{N_k}\boldsymbol{\Sigma}_k^{-1}\mathbf{x}_n = \sum_{n=1}^{N_k}\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k = N_k\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k$$

Now we can multiply both sides by the matrix $\boldsymbol{\Sigma}_k$ to obtain $\sum_{n=1}^{N_k} \mathbf{x}_n = N_k \boldsymbol{\mu}_k$ finally given the Maximum-Likelihood estimate for the mean of the class-conditional Multivariate Gaussian as

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \mathbf{x}_n$$

This is an intuitively appealing result as the Maximum-Likelihood estimate of the mean is simply the sample mean estimate.

Now we can do the same for the ML estimate of the required covariance matrix $\boldsymbol{\Sigma}_k$. We need to exploit some standard results from the Matrix Cookbook to proceed as we need the partial derivatives of the logarithm of the determinant of a symmetric matrix i.e. from the expression of the log-likelihood above

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \frac{N_k}{2} \log |\boldsymbol{\Sigma}_k|$$

Now from Section 2.1.2 of the Matrix Cookbook we have the following equality

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k} |\boldsymbol{\Sigma}_k| = |\boldsymbol{\Sigma}_k| \left(\boldsymbol{\Sigma}_k\right)^{-1}$$

we no longer require the explicit matrix transformation as $\left(\boldsymbol{\Sigma}_k\right)^{-1}$ is symmetric. Now we know that the derivative of the logarithm of a scalar value is one over the scalar value then we see that

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \frac{N_k}{2} \log |\boldsymbol{\Sigma}_k| = \frac{N_k}{2|\boldsymbol{\Sigma}_k|} |\boldsymbol{\Sigma}_k| \left(\boldsymbol{\Sigma}_k\right)^{-1} = \frac{N_k}{2} \boldsymbol{\Sigma}_k^{-1}$$

Now back to the Cookbook, Section 2.2 expression third from bottom, shows that

$$\frac{\partial}{\partial \mathbf{X}} \mathbf{a}^\mathsf{T} \mathbf{X}^{-1} \mathbf{b} = -\mathbf{X}^{-1} \mathbf{a} \mathbf{b}^\mathsf{T} \mathbf{X}^{-1}$$

again we have dropped the matrix transpose as we are working with a symmetric matrix, using this expression then

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \sum_{n=1}^{N_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) = -\sum_{n=1}^{N_k} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T} \boldsymbol{\Sigma}_k^{-1}$$

plugging everything together then we obtain

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \log \mathcal{L}_k = -\frac{N_k}{2}\boldsymbol{\Sigma}_k^{-1} + \frac{1}{2}\sum_{n=1}^{N_k} \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathsf{T}}\boldsymbol{\Sigma}_k^{-1}$$

So setting this gradient to zero, replacing the mean vectors with their ML estimate and after a little manipulation we see that our estimate for the class-conditioned covariance is, as we would expect

$$\widehat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k}\sum_{n=1}^{N_k}(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)^{\mathsf{T}}$$

We can apply the ML estimation method to any parametric form of probability density or distribution function in a straightforward manner. Of course we can also adopt a Bayesian approach to estimating the parameters of a density function by setting appropriate priors for the mean and covariance terms. Resisting the temptation as much as possible we will not explore the Bayesian approach further in this section.

2.2 Illustrative Examples

The ML estimates of the mean vector and covariance matrix of a multivariate Gaussian turn out to be quite intuitive in that sample estimates emerge. As an example the Matlab script `gauss_density_est.m`, which can be downloaded form the module website, provides and illustration of the ML estimation of a multivariate Gaussian from an available sample of data. The script allows you to generate a random sample of data drawn from a 2D Gaussian with parameters

$$\boldsymbol{\mu} = \begin{bmatrix} 1.0 \\ 3.0 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1.5 & 0.6 \\ 0.6 & 0.4 \end{bmatrix}$$

We can now use the sample to obtain estimates for the required parameters, see Figure (1), clearly as the sample size $N \to \infty$ then the estimates will converge to the true values[1].

For a typical sample of size $N = 30$ the following estimates are obtained

$$\widehat{\boldsymbol{\mu}} = \begin{bmatrix} 0.91 \\ 3.05 \end{bmatrix} \quad \widehat{\mathbf{C}} = \begin{bmatrix} 2.20 & 0.90 \\ 0.90 & 0.53 \end{bmatrix}$$

[1]In actual fact the estimate of the covariance is biased and a normalisation of $\frac{1}{N-1}$ is required to remove this bias.

Given that the estimate of the covariance depends on our estimate of the mean as well then we can expect to see some degree of higher variability in the estimates of the covariance terms as indeed we do.

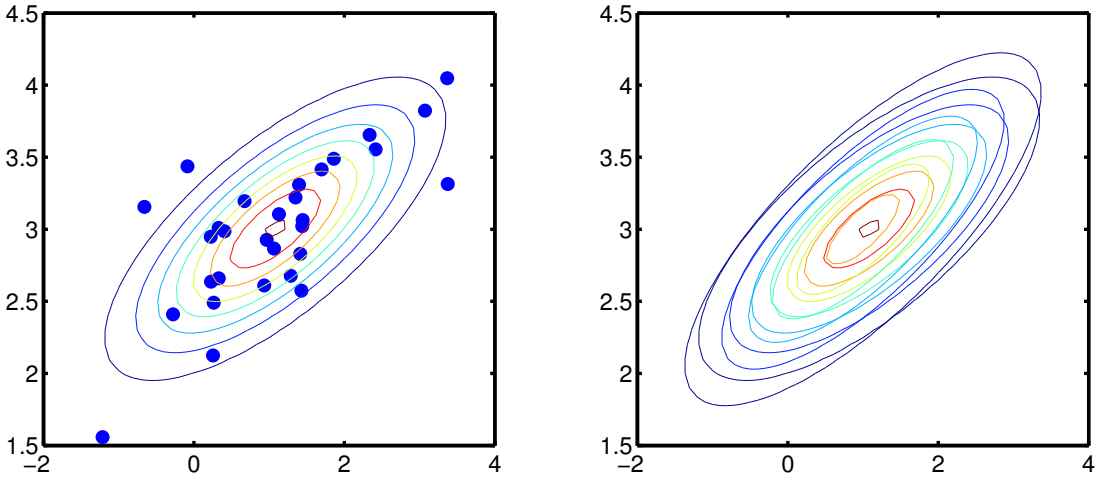

Figure 1: The left plot shows a random sample of 30 points drawn from a 2D Gaussian, the iso-contours of estimated probability density are superimposed on the plot. The iso-contours of probability density for the Gaussian with the actual parameter values are given on the left hand plot superimposed upon the iso-contours of estimated density.

Now this is excellent as if we are interested in devising a classifier we can now plug these estimates of the mean values $\widehat{\boldsymbol{\mu}}$ and covariance $\widehat{\mathbf{C}}$ into our class conditional densities to obtain $p(\mathbf{x}|\widehat{\boldsymbol{\mu}}_k, \widehat{\mathbf{C}}_k)$ which then allows one to produce the posterior probabilities required to produce a discriminant.

2.3 Non-Gaussian Example

Let us now look at another example of data for which we, wrongly, will assume that the density is also Gaussian. The data in fact will be drawn from a mixture of two Gaussians. In Figure (2) we see a random sample of 30 points drawn, with equal probability, from two separate Gaussians each which have mean and covariances of

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 0.5 \\ 2.0 \end{bmatrix} \quad \mathbf{C}_1 = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$$

$$\boldsymbol{\mu}_2 = \begin{bmatrix} 3.0 \\ 4.0 \end{bmatrix} \quad \mathbf{C}_2 = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$$

So our data can be considered as coming from two sub-populations, or there are two distinct generating processes each responsible for producing the data we observe. A classic example is the discrimination of two species of salt water crab based on a number of physical measurements from the shell. Of course the examples from one species will include both male and female and so the class conditional densities for each species would be well described by a mixture of densities one describing the variability of the measurements in the female sub-population of the species and the other for the males of the species.

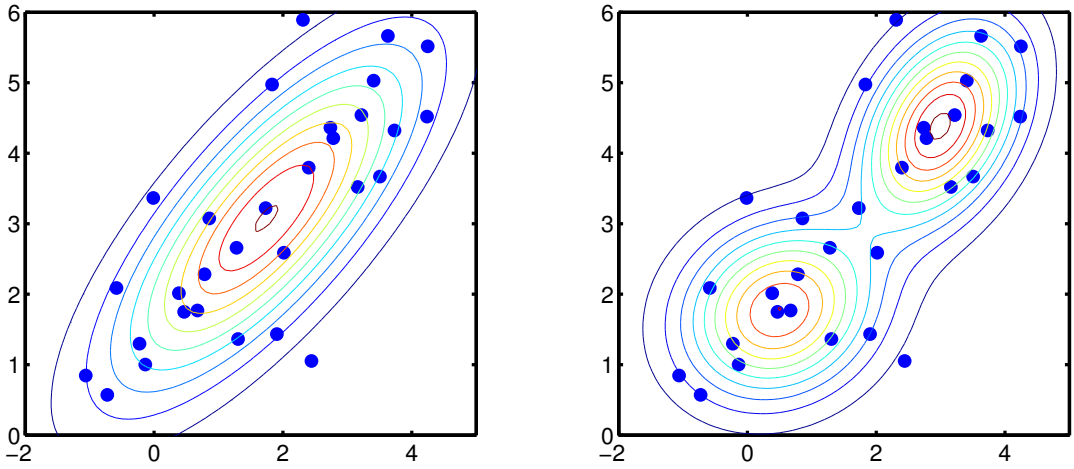

Figure 2: The left plot shows a random sample of 30 points drawn from 2 distinct 2D Gaussians, the iso-contours of estimated probability density under the assumption that the density is a single 2D Gaussian, are superimposed on the plot. The right plot shows the same random sample with the iso-contours of estimated probability density where the true functional form has been employed i.e. two 2D Gaussians.

Now for the data we are considering in this example we know that there

are two Gaussians responsible for the data we see in Figure (2). We have stated that each of the two Gaussians is equally probable for the generation of data points which means that, on average, we would anticipate equal proportions of data points coming from each Gaussian component.

If we measure the average likelihood of points spread uniformly across the regions shown in the figures under the assumption of a probability density function which is a single Gaussian then the average TEST likelihood is -3.261 whereas the averge TEST likelihood for the same points when a mixture of two Gaussians has been assumed, and the associated parameters estimated, turn out to be -3.123. This is higher than that achieved when assuming a single Gaussian and so provides a superior predictive generative model of the data. We will find that such forms of mixture of components is an important class of probability density functions and the following section will now consider these in some more detail. A Matlab script `mix_gauss_density.m` is available on the course website to allow you to replicate these results.

3 Mixture Models

The probability density function for the case of two Gaussians can be represented as

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\theta}) &= \pi p(\mathbf{x}|\boldsymbol{\theta}_1) + (1-\pi)p(\mathbf{x}|\boldsymbol{\theta}_2) \\ &= \pi \mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}_1, \mathbf{C}_1) + (1-\pi)\mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}_2, \mathbf{C}_2) \end{aligned}$$

where $\boldsymbol{\theta} = \{\pi, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ and each set of parameters is defined by $\boldsymbol{\theta}_1 = \{\boldsymbol{\mu}_1, \mathbf{C}_1\}$ and $\boldsymbol{\theta}_2 = \{\boldsymbol{\mu}_2, \mathbf{C}_2\}$.

The parameter π is the probability that a point $\mathbf{x}$ will be generated from $p(\mathbf{x}|\boldsymbol{\theta}_1)$ and so the probability that the point will be generated from $p(\mathbf{x}|\boldsymbol{\theta}_2)$ is $1-\pi$.

In the more general case where there are M components, of arbitrary parametric form[2] in our mixture model then the probability density will be expressed as

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{m=1}^{M} \pi_m p(\mathbf{x}|\boldsymbol{\theta}_m)$$

[2]The mixture components can take on any parametric form appropriate for the modeling task at hand.

where now of course the whole parameter set is defined as $\boldsymbol{\theta} = \{\pi_1 \cdots \pi_M, \boldsymbol{\theta}_1 \cdots \boldsymbol{\theta}_M\}$ and

$$\sum_{m=1}^{M} \pi_m = 1$$

as each π_m is the probability that the mth component of the mixture will produce a data point and so it must sum to one to be a valid probability over the M selection events.

This form of mixture of parametric models is often referred to as a Semi-Parametric Mixture Model and one of the main tasks ahead of us is the estimation of the parameters of a mixture model $\boldsymbol{\theta} = \{\pi_1 \cdots \pi_M, \boldsymbol{\theta}_1 \cdots \boldsymbol{\theta}_M\}$ which is not as straightforward a ML estimation of a parametric model and we now consider this in some detail.

4 Parameter Estimation in Mixture Models and the EM Algorithm

Suppose that we have a sample of data $\mathcal{D} = \{\mathbf{x}_1 \cdots \mathbf{x}_N\}$ and we have, with some appropriate justification, assumed that the probability density is of a semi-parametric form based on a mixture model of the form given above i.e $p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{m=1}^{M} \pi_m p(\mathbf{x}|\boldsymbol{\theta}_m)$. Now then we need to estimate the parameters of the model, well this looks to be a straightforward task.

To obtain estimates of each π_m, which is the relative probability of data being generated by each of the m components, then all we need to do is count how many points from $\mathcal{D}$ were produced by each of the M components and then simply normalise by the number of samples N. So if we count that there are N_m points in $\mathcal{D}$ drawn from component m then

$$\widehat{\pi}_m = \frac{N_m}{N}$$

where each N_m can be obtained from

$$N_m = \sum_{n=1}^{N} z_{mn}$$

where each $z_{mn} = 1$ if the nth point was drawn from component m and $z_{mn} = 0$ otherwise.

We are now making some progress. What of the specific parameters of each of the components $\boldsymbol{\theta}_m$?

Well this is also easy as all we need to do is obtain the estimates $\widehat{\boldsymbol{\theta}}_m$ which maximise the likelihood of the data points which were drawn from component m under the parametric form $p(\mathbf{x}|\boldsymbol{\theta}_m)$.

For example if the mixture components were Gaussians then the Maximum-Likelihood estimate for the component mean vectors would simply be

$$\widehat{\boldsymbol{\mu}}_m = \frac{\sum_{n=1}^{N} z_{nm}\mathbf{x}_n}{\sum_{n=1}^{N} z_{nm}} = \frac{1}{N_m}\sum_{n\in m}\mathbf{x}_n$$

The expression for the covariance matrices for each component would follow simply as

$$\widehat{\boldsymbol{\Sigma}}_m = \frac{1}{N_m}\sum_{n=1}^{N} z_{mn}(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_m)(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_m)^{\mathsf{T}}$$

and we are then finished.

There is one small difficulty which we have overlooked, we do not have values for the indictor variables z_{mn} on which we have relied.

This is a major difficulty as the fact that the variables z_{mn} are hidden or latent then our ML estimates cannot follow in the straightforward manner we had anticipated.

In addition to the estimation problem we will have to impute the values of the indictor values z_{mn} which assign each data point to a mixture component and this is a non-trivial problem. Bet you are glad you took the Machine Learning module?

4.1 The EM Algorithm

The problem with our reasoning in the previous section was that we assumed that we knew the value of the allocation or indicator variables z_{mn}. So what is needed is the joint likelihood of our data $\mathbf{X} = \{\mathbf{x}_1 \cdots \mathbf{x}_N\}$ and the indicator variables $\mathbf{Z} = \{\mathbf{z}_1 \cdots \mathbf{z}_N\}$ where each $\mathbf{z}_n = \{z_{1n} \cdots z_{Mn}\}$. Now given the parameter values $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1 \cdots \boldsymbol{\theta}_M\}$ we can marginalise over all possible component allocations of the data such that

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$$

where the summation is over all possible values which $\mathbf{Z}$ may take on.

Now we use the following

$$\begin{aligned}\log p(\mathbf{X}|\boldsymbol{\theta}) &= \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \\ &= \log \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X}) \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{P(\mathbf{Z}|\mathbf{X})}\end{aligned}$$

There is an inequality for convex functions such as the logarithm which states that $\log E\{f(X)\} \geq E\{\log f(X)\}$ now the above is an expectation with respect to $P(\mathbf{Z}|\mathbf{X})$ and so we can write

$$\begin{aligned}\log \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X}) \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{P(\mathbf{Z}|\mathbf{X})} &\geq \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X}) \log \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{P(\mathbf{Z}|\mathbf{X})} \\ &= \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X}) \log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \\ &- \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X}) \log P(\mathbf{Z}|\mathbf{X})\end{aligned}$$

As the assumption that each $\mathbf{x}_n$ is generated iid from each mth component exclusively then the summation over all $\mathbf{Z}$ will be equal to a summation over all n and m i.e.

$$\begin{aligned}\mathcal{L}_B = \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X}) \log \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{P(\mathbf{Z}|\mathbf{X})} &= \sum_{m=1}^{M} \sum_{n=1}^{N} P(m|\mathbf{x}_n) \log \frac{p(\mathbf{x}_n|\boldsymbol{\theta}_m) P(m)}{P(m|\mathbf{x}_n)} \\ &= \sum_{m=1}^{M} \sum_{n=1}^{N} P(m|\mathbf{x}_n) \log p(\mathbf{x}_n|\boldsymbol{\theta}_m) P(m) \\ &- \sum_{m=1}^{M} \sum_{n=1}^{N} P(m|\mathbf{x}_n) \log P(m|\mathbf{x}_n)\end{aligned}$$

where now $P(m|\mathbf{x}_n)$ is the probability that $z_{mn} = 1$ and $P(m)$ is the probability that $z_{mn} = 1$ for any n.

Lets just step back for a second. The data log-likelihood $\log p(\mathbf{X}|\boldsymbol{\theta})$ requires marginalisation of the allocation variable for all data points n and all values m. We have shown that a lower-bound on the marginal likelihood

can be obtained by introducing the $P(m|\mathbf{x}_n)$ and using the inequality for convex functions introduced above. The reason for this is that the allocation variables $\mathbf{Z}$ are hidden from us, in other words we do not know which of the mixture components generated a particular data point say $\mathbf{x}_n$. If we did know $\mathbf{Z}$ then our job would be straightforward, as it is we do not know $\mathbf{Z}$ and so we must infer them and then make our estimates of the parameters $\boldsymbol{\theta}_m$.

The Expectation Maximisation (EM) algorithm is a general purpose method for this particular problem where we would like to *Maximise* the likelihood of the complete data ($\mathbf{X}$ & $\mathbf{Z}$) so as to obtain estimates of the component parameters $\boldsymbol{\theta}_m$. However before performing the *Maximisiation* step we require to obtain the *Expected* values of a set of hidden (unobserved or latent) binary allocation variables z_{mn}. Once we have obtained the *Expected* values of the latent variables we can then perform the *Maximisation* step to obtain our current parameter estimates. This EM interleaving is continued until some convergence criterion is achieved.

4.1.1 Expectation Step

Taking functional derivatives of the lower-bound with respect to $P(m|\mathbf{x}_n)$ then

$$\frac{\partial \mathcal{L}_B}{\partial P(m|\mathbf{x}_n)} = \log P(m|\mathbf{x}_n) - \log p(\mathbf{x}_n|\boldsymbol{\theta}_m)P(m) - 1$$

setting this to zero then we see that $P(m|\mathbf{x}_n) \propto p(\mathbf{x}_n|\boldsymbol{\theta}_m)P(m)$ and normalising appropriately yields the distribution of the form

$$P(m|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|\boldsymbol{\theta}_m)P(m)}{\sum_{m'=1}^{M} p(\mathbf{x}_n|\boldsymbol{\theta}_{m'})P(m')}$$

You should now be able to see that this is the posterior distribution over the mixture components m which generated $\mathbf{x}_n$, or the expected value of the binary variable z_{mn}.

Now that we have maximised the bound with respect to the *Expected* value of the indicator variable we need to *Maximise* the bound with respect to the parameter values.

4.1.2 Maximisation Step

The only terms in the bound $\mathcal{L}_B$ which are dependent on the component parameters are

$$\sum_{m=1}^{M}\sum_{n=1}^{N} P(m|\mathbf{x}_n)\log p(\mathbf{x}_n|\boldsymbol{\theta}_m)P(m)$$

in which case we maximise the above with respect to each $\boldsymbol{\theta}_m$.

As an example assume that each $p(\mathbf{x}_n|\boldsymbol{\theta}_m)$ is a multivariate Gaussian, then expanding and retaining the elements dependent on the parameters we obtain

$$\begin{aligned}
- & \frac{1}{2}\sum_{m=1}^{M}\sum_{n=1}^{N} P(m|\mathbf{x}_n)\log|\boldsymbol{\Sigma}_k| \\
- & \frac{1}{2}\sum_{m=1}^{M}\sum_{n=1}^{N} P(m|\mathbf{x}_n)(\mathbf{x}_n-\boldsymbol{\mu}_m)^{\mathsf{T}}\boldsymbol{\Sigma}_m^{-1}(\mathbf{x}_n-\boldsymbol{\mu}_m) \\
+ & \sum_{m=1}^{M}\sum_{n=1}^{N} P(m|\mathbf{x}_n)\log P(m)
\end{aligned}$$

Taking derivatives with respect to each $\boldsymbol{\mu}_m$ of the above and solving yields

$$\widehat{\boldsymbol{\mu}}_m = \frac{\sum_{n=1}^{N} P(m|\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} P(m|\mathbf{x}_n)}$$

this is a nice result as if we compare the estimator which we obtain in the case where we have perfect knowledge of the allocation variables z_{mn} i.e.

$$\widehat{\boldsymbol{\mu}}_m = \frac{\sum_{n=1}^{N} z_{mn}\mathbf{x}_n}{\sum_{n=1}^{N} z_{mn}}$$

so in the absence of the values z_{mn} we employ the expected values, or the posterior probabilities $P(m|\mathbf{x}_n)$ which are obtained in the *Expectation* step.

Leaving you to have some fun with the derivation of the estimator for the covariance matrices we obtain

$$\widehat{\boldsymbol{\Sigma}}_m = \frac{\sum_{n=1}^{N} P(m|\mathbf{x}_n)(\mathbf{x}_n-\widehat{\boldsymbol{\mu}}_m)(\mathbf{x}_n-\widehat{\boldsymbol{\mu}}_m)^{\mathsf{T}}}{\sum_{n=1}^{N} P(m|\mathbf{x}_n)}$$

again we can see that we have replaced perfect knowledge of the allocation variables with our current estimates of the posteriors $P(m|\mathbf{x}_n)$, is'nt this cool?

Finally we need an estimate for $P(m)$ taking derivatives then we observe that

$$P(m) \propto \sum_{n=1}^{N} P(m|\mathbf{x}_n)$$

This needs to be properly normalised and so

$$P(m) = \frac{1}{N} \sum_{n=1}^{N} P(m|\mathbf{x}_n)$$

and so we are complete. Let us now summarise our EM algorithm for a mixture of Gaussians.

E Step

$$P(m|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|\boldsymbol{\theta}_m)P(m)}{\sum_{m'=1}^{M} p(\mathbf{x}_n|\boldsymbol{\theta}_{m'})P(m')}$$

M Step

$$\begin{aligned}
\widehat{\boldsymbol{\mu}}_m &= \frac{\sum_{n=1}^{N} P(m|\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} P(m|\mathbf{x}_n)} \\
\widehat{\boldsymbol{\Sigma}}_m &= \frac{\sum_{n=1}^{N} P(m|\mathbf{x}_n)(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_m)(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_m)^{\mathsf{T}}}{\sum_{n=1}^{N} P(m|\mathbf{x}_n)} \\
P(m) &= \frac{1}{N} \sum_{n=1}^{N} P(m|\mathbf{x}_n)
\end{aligned}$$

Repeating the EM steps will maximise the lower-bound on the marginal likelihood and thus allow us to estimate the parameters of a mixture density despite having incomplete knowledge of the mixture allocations of each data point.

5 Experiments

There is a Matlab file which you should download from the website `Gauss_Mix_Data.mat` there is a matrix $\mathbf{X}$ which is of dimension 150×2. The data has been

generated by drawing points with equal probability from three possible 2D Gaussians which have a common unit variance isotropic covariance i.e. $\mathbf{I}$ and have means of $[0, 0], [3, 3], [-3, 3]$. There is also a 1500×2 dimensional data set drawn from the same distribution which can be used to obtain values of likelihood on independent test data.

So now we wish to estimate the probability density from which this sample of data has been drawn. Armed with our newly acquired EM algorithm we can set to work in estimating the mixture density. There is, as always, one slight snag, our EM algorithm requires that we provide it with a value for the number of components in the mixture. For now lets assume that we have a good idea what this value is.

Running the Matlab script `gauss_mix_em_demo.m` will produce a dynamic view of how the estimate of the mixture density is evolving at each EM-step. The final converged solution is shown in Figure (3).

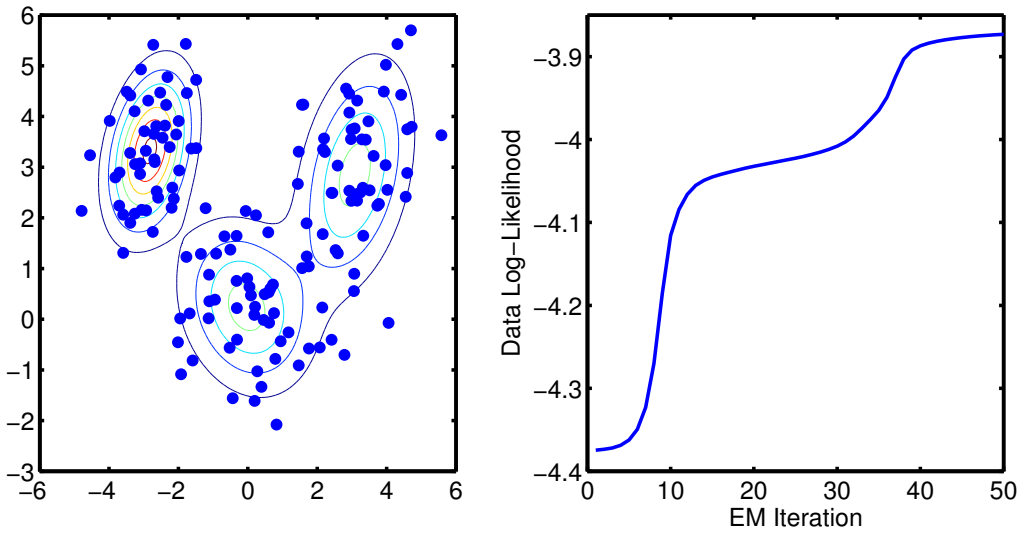

Figure 3: The left plot shows a random sample of 150 points drawn from 3 distinct 2D Gaussians, the iso-contours of estimated probability density under the assumption that the density is a mixture of three 2D Gaussians, are superimposed on the plot. The right plot shows the data likelihood under the mixture model at each EM step, it is clear that the likelihood does not decrease at each step.

We mentioned the requirement to know the number of mixture compo-

nents in the model. One way to assess the required number of components is to use cross-validation and score held-out data based on the likelihood. In Figure (4) we see the likelihood of both test and training data under a mixture model with a varying number of components. As we have now come to expect the likelihood of the training data increases as the model complexity increases whilst the testing likelihood falls after 3 to 4 components have been added so it is clear that around 3 or 4 components are optimal for this data.

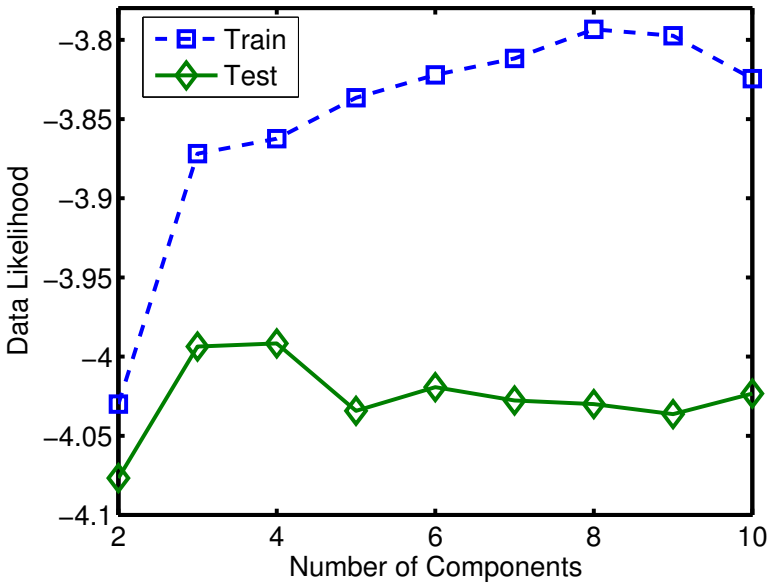

Figure 4: The likelihood obtained for training and testing data for various numbers of mixture components. As expected the out of sample likelihood decreases as too many components are used.

6 Conclusion

The EM algorithm is generic and any mixture of densities or distributions can be provided with an EM algorithm for parameter estimation.

You should attempt to develop an EM algorithm for a mixture of Bernoulli distributions for binary data where each component takes the parametric form of

$$p(\mathbf{x}|m) = \prod_{d=1}^{D} p_{md}^{x_{nd}} (1 - p_{md})^{(1-x_{nd})}$$

and the overall mixture is

$$p(\mathbf{x}) = \sum_{m=1}^{M} \pi_m \prod_{d=1}^{D} p_{md}^{x_{nd}} (1 - p_{md})^{(1-x_{nd})}$$