Tutorial:
Estimation of Finite Normal Mixtures with the EM Algorithm

Prof. Robert J. Frey, Research Professor
Director, Program in Quantitative Finance
Applied Mathematics and Statistics
Stony Brook University

27 March 2015 (01)

There has been increasing interest in modeling certain types of heterogeneous processes by finite mixtures of Normal distributions. Formerly, the approach was to fit the finite mixture either by solving for mixture weights assuming the component distributions are known or the reverse. This tutorial describes how to use the EM-algorithm to achieve a MLE for finite Normal mixtures for which the only assumptions made are (1) the order of the mixture (i.e., the number of component distributions) and (2) the fact that the components are Normal distributions.

As we work our way through the various concepts, we also demonstrate them via an example, embedding the working code in this Mathematica notebook. The approach is somewhat informal with its primary objective to develop the student's intuition. The expectation is that he or she will thereafter be better able to understand and apply more advanced material. A list of guided readings is also provided.

Format Note

In original form this tutorial is intended to read as a active *Mathematica* notebook with executable statements throughout used to illustrate the topics covered. You may, however, find yourself with a PDF of that notebook which is also being made available so that it may be read by a wider audience.

Finite Normal Mixtures

Finite mixture distributions arise in problems for which a single distributional form appears to inadequately fit the data. Especially in situations in which an analyst or researcher believes the observed results may come multiple distinct regimes, the notion that the distribution can be best described as a weighted mixture of simpler distributions is naturally appealing. Often, even if multiple regimes *per se* are not envisioned, a finite mixture can provide a powerful technique for building it up a complex distribution from simpler ones. In this tutorial we will consider the problem of fitting a finite mixture of Normal distributions to a sample.

This is not only a useful tool to have available but it is also an excellent example to help individuals understand the EM algorithm. The EM algorithm has emerged as a powerful and general technique for handling a number of otherwise intractable statistical problems. Understanding it also represents a solid foundation for proceeding on to even more powerful Markov chain Monte Carlo (MCMC) techniques as many examples of MCMC, *e.g.*, Gibbs sampling, can be viewed as replacing a portion of the EM algorithm with a Monte Carlo sample rather than an analytic estimate.

A Generative Model

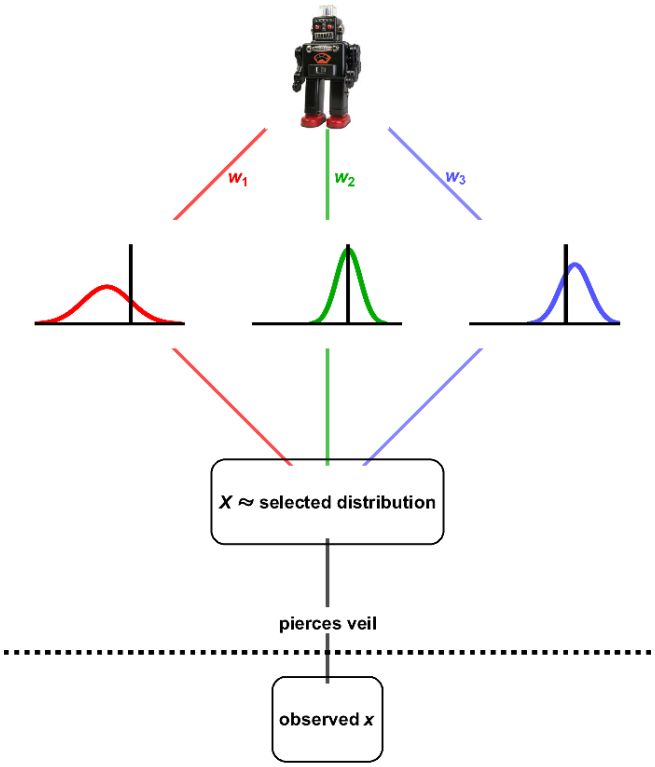
A generative model for the finite Normal mixture starts with a robot who randomly selects one of a finite number of m distinct colors. The color selection follows a multinomial distribution where $0 < w_i$ is the probability of selecting color $i \in \{1, \ldots, m\}$. Clearly, the sum of the propor-

tions w_i over all i must equal 1.

Associated with each color i is a distinct Normal distribution with finite mean μ_i and standard deviation σ_i. Based on the color i chosen, the robot next generates a single random number using the Normal distribution associated with i.

The robot does its work behind a veil and cannot be directly observed by us. Once it has generated the random number from the chosen Normal distribution, it passes out the result from behind the viel to where it can now finally be observed.

The robot repeats this sequence of color selection and color-conditional random number generation N times, generating an i.i.d. sample $\mathbf{x} = \{x_1, \ldots, x_j, \ldots, x_N\}$. *The sample* $\mathbf{x}$ *is said to have been drawn from a finite mixture of Normal distributions:*



The problem we face is: Given the n observations $\mathbf{x}$ and an assumed number of components m, perform a maximum likelihood estimate (MLE) of the parameters of the finite mixture of Normals.

The Likelihood Function

Assume that we have m component Normal distributions, each with its associated mixture weight, represented by the parameter set Ψ:

$$\Psi = \{\{w_1, \mu_1, \sigma_1\}, \ldots, \{w_i, \mu_i, \sigma_i\}, \ldots, \{w_m, \mu_m, \sigma_m\}\} \tag{1}$$

$$\sum_{i=1}^{m} w_i = 1 \ \wedge \ w_i, \sigma_i \geq 0 \tag{2}$$

If $\phi(x \mid \mu, \sigma)$ is the PDF of a Normal distribution with mean μ and standard deviation σ, then the PDF of the mixture distribution $f(x \mid \Psi)$ is:

$$f(x \mid \Psi) = \sum_{i=1}^{m} w_i \, \phi(x \mid \mu_i, \sigma_i) \tag{3}$$

The function $f(x \mid \Psi)$, as a convex combination of probability densities, is hence itself a valid density.

The goal of a maximum likelihood estimate (MLE) for Ψ given a data vector $\mathbf{x}$ is to maximize the likelihood function:

$$\log \mathcal{L}(\Psi \mid x) = \sum_{j=1}^{N} \log\left[\sum_{i=1}^{m} w_i \, \phi(x_j \mid \mu_i, \sigma_i)\right] \tag{4}$$

Solving the MLE for (4) is not computationally tractable. However, if we knew at each sample point from which component distribution the

sample was drawn, then our task would be trivial. Looked at from that perspective, a complete description of a random sample of size n drawn from (3) would consist of an observed data vector $\mathbf{x}$ and occult data in the form of an $m \times n$ membership matrix $\mathbf{Z}$, where $z_{i,j} = 1$ if observation j was drawn from distribution i and $z_{i,j} = 0$ otherwise.

If we possessed both $\mathbf{x}$ and $\mathbf{Z}$, then the estimation of Ψ would be straightforward. Under such happy circumstances the log likelihood function of Ψ given the *complete* sample information $\{\mathbf{x}, \mathbf{Z}\}$ is:

$$\log L_C(\Psi \mid \mathbf{x}, \mathbf{Z}) \propto \sum_{i=1}^{m}\sum_{j=1}^{N} z_{i,j}\left[\log[w_i] - \frac{1}{2}\left(\log[\sigma_i^2] + \frac{(x_j - \mu_i)^2}{\sigma_i^2}\right)\right] \tag{5}$$

Note that each draw is from a specific component; hence,

$$\sum_{i=i}^{m} z_{i,j} = 1 \;\; \forall \;\; j \in \{1, \ldots, n\} \tag{6}$$

Thus, we have recast the finite mixture problem embodied in (4) into a missing value problem embodied in (5) where the membership of each outcome $z_{i,j}$ is cast in the role of missing data.

The EM Algorithm

The EM algorithm [Dempster *et al*., 1977] can be used for MLE when there are occult (*i.e*., hidden, latent, censored, or missing) data or when a problem can be reformulated in those terms. Informally, the EM algorithm takes the observed data and a current estimate of the parameters and uses them to estimate the missing data; it then takes the observed data and the missing data estimates and uses them to produce a new estimate of the parameters. The two steps are repeated until the estimates stabilize. Remarkably, under suitable conditions, this iterative process converges to a local maximum of the likelihood function [Dempster *et al*. 1977].

More formally, at the η^{th} iteration given a complete data set X, occult data Z, and a working estimate of parameters $\vartheta^{(\eta)}$.

The expectation (E) step forms the expected value function of the log likelihood function of the conditional distribution of Z given X and $\vartheta^{(\eta)}$.

$$Q\left[\vartheta, \vartheta^{(\eta)}\right] = \mathrm{E}\left[\log \mathcal{L}[Z \mid X, \vartheta] \mid X, \vartheta^{(\eta)}\right] \tag{7}$$

For problems in which the data's distribution is one of the exponential family, which includes the Normal distribution, the the E-step involves computing sufficient statistics for the missing data.

The maximization (M) step then finds a successor parameter set $\vartheta^{(\eta+1)}$ which maximizes that expectation.

$$\vartheta^{(\eta+1)} = \underset{\vartheta}{\operatorname{argmax}}\, Q\left[\vartheta, \vartheta^{(\eta)}\right] \tag{8}$$

The parameter set $\vartheta^{(\eta+1}$ then becomes the new working parameter set. EM iterations continue until $\log \mathcal{L}[\vartheta \mid X]$ converges.

Issues with the EM Algorithm

Each EM iteration results in a non-decreasing likelihood, but the EM algorithm converges sub-linearly and may find only a local maximum. There may be material numerical instabilities affecting computations in practice. Convergence can be difficult to establish. The log likelihood can seem to level off indicating convergence but then suddenly take off. This behavior can trick us into terminating the algorithm prematurely.

In the case of finite mixtures the algorithm may set certain elements of ϑ to explain a single point, and this causes the log likelihood of the sample to be indeterminate. Such single-point fits mean either that a less complex model is appropriate or that the offending data point is such an extreme outlier that it must be removed from the sample. Choosing the number of components m can be difficult. Often several values of m are run and some penalized likelihood method, *i.e*., the log likelihood penalized for the number of parameters estimated, is used to select the best value. Unfortunately, different penalization approaches lead to different choices with no obvious way to choose between them.

Estimating Finite Mixtures

The EM algorithm can be a bit abstract and daunting in its general form but is really based upon a quite simple idea. A well chosen example can make its application clear, and we have chosen estimating finite Normal mixtures as our prototypical example. The E- and M-steps are straightforward, and the intuition behind them is easy to grasp. The seminal work by [Dempster *et al*., 1977] showed that, remarkably, the likelihood increases with each iteration. It placed many iterative techniques that statisticians had used on an *ad hoc* basis onto a solid theoretical foundation,

The E-Step

The k–th E-step for our problem estimates $\mathbf{Z}$. The *expected values* of the $z_{i,j}$ given some Ψ. This is the i–th component distribution's proportion of the total likelihood for the j–th observation.

$$z_{i,j}^{(k)} = w_i^{(k)}\, \varphi\left(x_j;\; \mu_i^{(k)},\; \sigma_i^{(k)}\right) \Big/ f\left(x_j;\; \Psi^{(k)}\right) \tag{9}$$

Note that (9) by construction ensures that $0 \le z_{i,j}^{(k)} \le 1$ and that equation (6) above is satisfied. The value $z_{i,j}^{(k)}$ has been recast from membership

indicator to a likelihood conditioned on $\Psi^{(k)}$ as defined by equation (5). Thus, the j–th column of $\mathbf{Z}^{(k)}$ represents the PMF (*i.e.*, discrete distribution) that observation x_j belongs to a given component.

The M-Step

If we knew $\mathbf{Z}$, then we could use it to partition the data and estimate the parameters for each component distribution directly. Thus, the k–th M-step, which produces the iterate $\Psi^{(k+1)}$, computes the parameter estimates for the i–th component distribution by using the current estimate $\mathbf{Z}_i^{(k)} = \{z_{i,1}, \ldots, z_{i,j}, \ldots, z_{i,N}\}^{(k)}$:

To estimate the weight $w_i^{(k+1)}$ we simply add up the row of $\mathbf{Z}_i^{(k)}$ and divide by the sample size n. Equation (10) is guaranteed to satisfy equation (2):

$$w_i^{(k+1)} = \sum_{j=1}^{N} z_{i,j}^{(k)} \Big/ n \tag{10}$$

The weightings guide us in our estimates of the means and standard deviations. Note that each estimate is based on a proper weighted average of the observations:

$$\mu_i^{(k+1)} = \sum_{j=1}^{N} z_{i,j}^{(k)}\, x_j \Big/ \sum_{j=1}^{N} z_{i,j}^{(k)} \tag{11}$$

$$\sigma_i^{(k+1)} = \sqrt{\sum_{j=1}^{N} z_{i,j}^{(k)} \left(x_j - \mu_i^{(k+1)}\right)^2 \Big/ \sum_{j=1}^{N} z_{i,j}^{(k)}} \tag{12}$$

Combining the results (10), (11), and (12) gives us $\Psi^{(k+1)}$.

The Likelihood

We use equation (4) to compute the log likelihood and compare it to its previous value. If the result is "close enough" or we have hit some predetermined maximum number of iterations, then we terminate and report our results; otherwise, we loop back and perform another EM iteration.

Mathematica Example

Mathematica has a number of tools for estimating and manipulating a wide array of distributions, including finite mixtures. We will use them here to illustrate fitting a finite Normal mixture to a data sample.

MixtureDistribution[ ]

Mathematica's MixtureDistribution[ ] can be used to define finite mixtures:

```
? MixtureDistribution
```

MixtureDistribution[{w_1, …, w_n}, {$dist_1$, …, $dist_n$}] represents a mixture distribution whose CDF is given as a sum of the CDFs of the component distributions $dist_i$, each with weight w_i. ≫

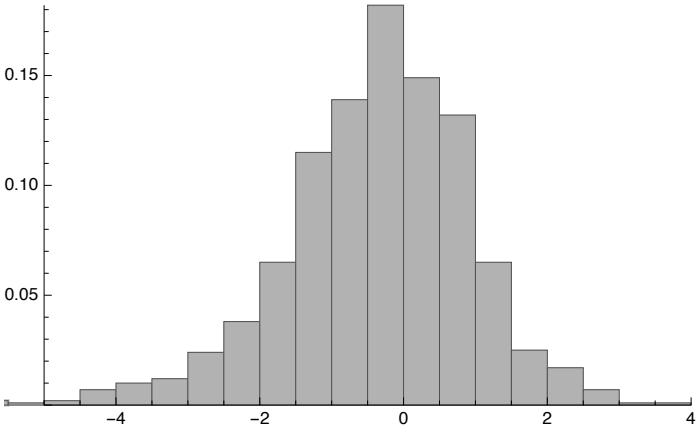
We define a finite mixture of Normals with $\Psi_{\mathcal{M}}$ = {{0.7, 0.0, 1.0}, {0.2, −1.0, 1.0}, {0.1, −2.0, 2.5}}

```
ℳ = MixtureDistribution[{0.7, 0.2, 0.1},
    {NormalDistribution[0., 1.], NormalDistribution[-1., 1.],
     NormalDistribution[-2., 2.5]}];
```

and generate a random sample for the distribution $\mathcal{M}$ of size N = 1000:

```
sample = RandomVariate[ℳ, 1000];
```

```
Histogram[sample, 20, "Probability"]
```



EstimatedDistribution[ ]

EstimatedDistribution[ ] can be used to fit distributions to data. There are a number criteria used to fit the distribution that can be selected via an option; however, the default is "MaximumLikelihood" and we can use the function in its simplest form:

```
? EstimatedDistribution
```

EstimatedDistribution[*data*, *dist*] estimates the parametric distribution *dist* from *data*.
EstimatedDistribution[*data*, *dist*, {{*p*, p_0}, {*q*, q_0}, …}]
estimates the parameters *p*, *q*, … with starting values p_0, q_0, ….
EstimatedDistribution[*data*, *dist*, *idist*] estimates distribution *dist* with starting values taken from the instantiated distribution *idist*. ≫

Finite Mixture of Normals $\mathcal{E}$

We will discuss later an approach to selecting m, the order of the mixture, but here we will simple assume $m = 3$ and fit using a mixture of univariate Normals, $\mathcal{E}$:

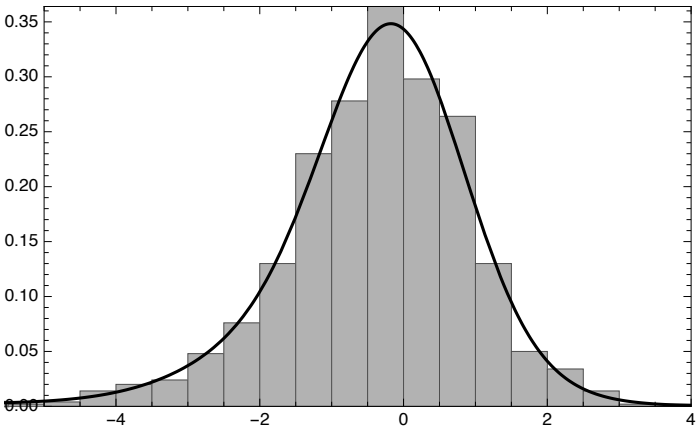
```
ℰ = EstimatedDistribution[sample,
  MixtureDistribution[{w1, w2, w3},
   {NormalDistribution[μ1, σ1], NormalDistribution[μ2, σ2],
    NormalDistribution[μ3, σ3]}]]
```

EstimatedDistribution: Failed to converge to the requested accuracy or precision within 500 iterations.

```
MixtureDistribution[{0.623162, 0.358563, 0.0182758},
 {NormalDistribution[-0.100498, 0.948316],
  NormalDistribution[-0.767334, 1.5385], NormalDistribution[-4.72546, 2.72476]}]
```

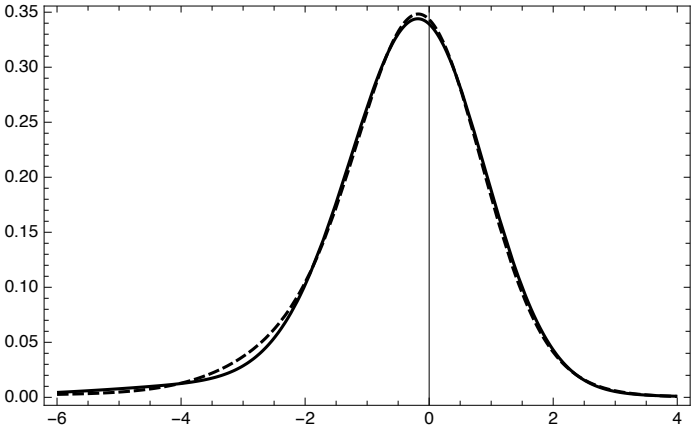
A plot of the distribution compared to the histogram of the data shows close correspondence:

```
Show[
 Histogram[sample, 20, "ProbabilityDensity", Frame → True],
 Plot[PDF[ℰ, x], {x, -6., 4.}]
]
```



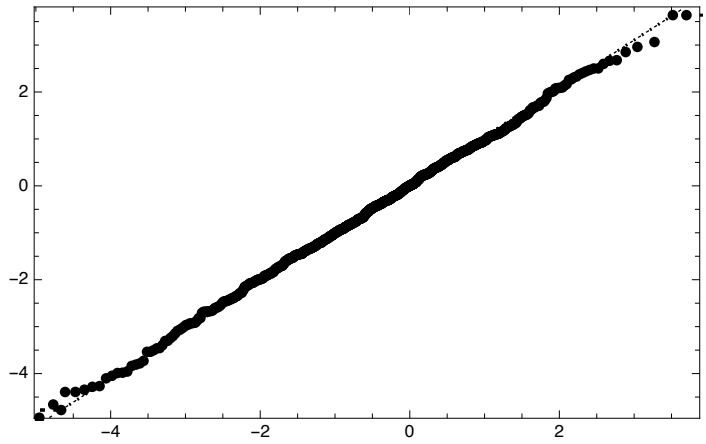
A comparison plot of $\mathcal{E}$ and the "true" distribution $\mathcal{M}$ illustrates how successful we have been:

```
Plot[Evaluate[{PDF[ℳ, x], PDF[ℰ, x]}], {x, -6., 4.}, Frame → True]
```



A quantile plot, sometimes called a qq-plot, shows that we close correspondence between $\mathcal{E}$ and the sample over the whole range:

```
QuantilePlot[sample, ε]
```



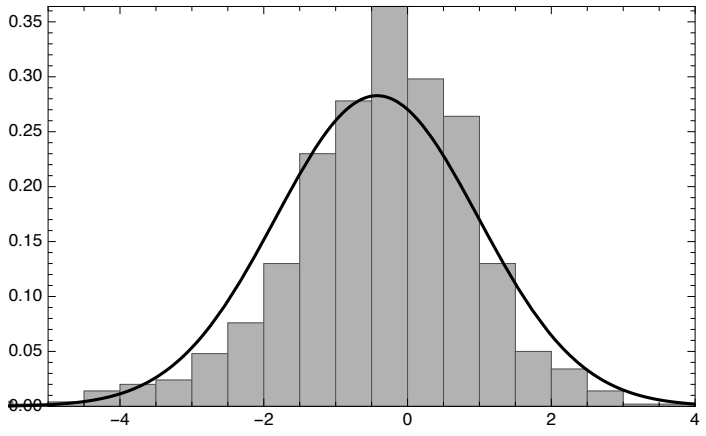
Univariate Normal Distribution $\mathcal{N}$

We perform the same analyses using a univariate Normal distribution, $\mathcal{N}$, realizing clearly less satisfactory results:

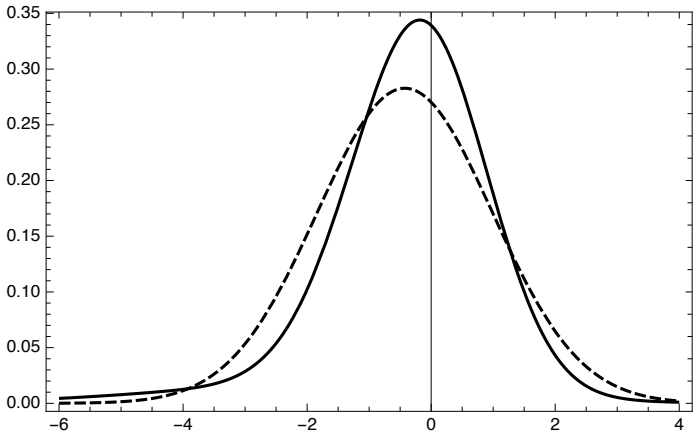
```
𝒩 = EstimatedDistribution[sample, NormalDistribution[μ, σ]]
```

```
NormalDistribution[-0.424125, 1.41083]
```

```
Show[
 Histogram[sample, 20, "ProbabilityDensity", Frame → True],
 Plot[PDF[𝒩, x], {x, -6., 4.}]
]
```

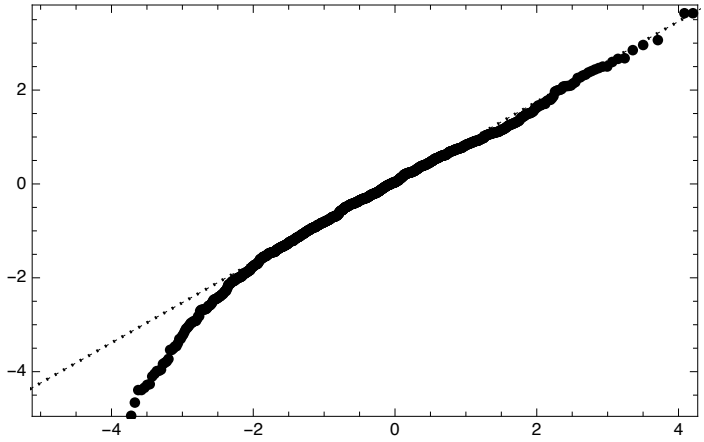


```
Plot[Evaluate[{PDF[ℳ, x], PDF[𝒩, x]}], {x, -6., 4.}, Frame → True]
```



Note how poorly we capture the lower tail with the Normal distribution:

```
QuantilePlot[sample, 𝒩]
```



Summary of Moments

Note that, not surprisingly, we get the first and second central moments right with all the models, but $\mathcal{E}$ matches the skewness and kurtosis of the sample closely. Comparing $\mathcal{M}$ and the sample, note that the kurtosis of a distribution, based on the 4-th power of observations net of the mean, is an extremely unstable statistic.

```
moments = x ↦ Through[{Mean, StandardDeviation, Skewness, Kurtosis}[x]];

TableForm[moments /@ {ℳ, sample, ℰ, 𝒩},
 TableHeadings → {{"ℳ", "sample", "ℰ", "𝒩"}, {"mean", "sdev", "skew", "kurt"}}]
```

	mean	sdev	skew	kurt
$\mathcal{M}$	−0.4	1.40178	−1.06299	6.6875
sample	−0.424125	1.41154	−1.27577	8.19508
$\mathcal{E}$	−0.424125	1.41083	−1.25648	8.28972
$\mathcal{N}$	−0.424125	1.41083	0	3

Penalized Log Likelihood

In deciding the order of the mixture, *i.e.*, the value of m, we can use the log likelihood as a guide. In *Mathematica* the LogLikelihood[ ] function can be used to compute $\log\mathcal{L}$ of a distribution or stochastic process:

```
? LogLikelihood
```

LogLikelihood[*dist*, {x_1, x_2, …}] gives the
log-likelihood function for observations x_1, x_2, … from the distribution *dist*.
LogLikelihood[*proc*, {{t_1, x_1}, {t_2, x_2}, …}] gives the log–likelihood function
for the observations x_i at time t_i from the process *proc*.
LogLikelihood[*proc*, {$path_1$, $path_2$, …}] gives the log–likelihood function for
the observations from $path_1$, $path_2$, … from the process *proc*. ≫

While we wish to have a high likelihood, unfortunately it is always possible to gain increases in the likelihood of a MLE by adding complexity to the model, *e.g.*, more dependent variables to a regression, more components to a finite mixture and so forth. Too much model complexity may achieve high levels of likelihood *ex post* (*i.e.*, in-sample) but that is spurious over-fitting. Such fits tend to have poor *ex ante* (*i.e.*, out-of-sample) performance. What we need is some means of *penalizing* the log likelihood for model complexity so that these effects can be balanced.

You Are Already Familiar with Penalized Likelihood

In many statistical procedures, such as ordinary least squares (OLS), measures of goodness-of-fit are not based solely on likelihood but on one that penalizes for model complexity. In OLS for example, likelihood is maximized by minimizing mean square error; however, measures such as the error variance and *r*–squared are computed by dividing the sum of squared errors not by the sample size N but by $N - k$ where k is the number of parameters, including the intercept. Another example is the usual unbiased estimate of the variance where the squared deviations from the mean are divided by $N - 1$ to account for the fact that we have used the same sample to estimate the mean; *i.e.*, we penalize for one parameter.

We want to minimize the mean square error in an OLS but only if it "pays" to do so at the given level of model complexity k. At some point the sum of squared errors decreases more slowly than does the adjusted sample size $N - k$. The error variance actually increases from adding parameters after that. The theory behind the EM algorithm requires us to use MLE. If we are to be guided by log likelihood, then we need some appropriate mechanism for penalizing it for model complexity.

The Bayesian Information Criterion (BIC)

The Bayesian information criterion (BIC) is a penalized likelihood measure that selects from among a finite set of models. The numerical values of the dependent variable must be identical across all models. The model with the lowest BIC is preferred.

With the sample log likelihood, $\log \hat{\mathcal{L}}$, the number of independent parameters estimated, k, and the sample size, N, the BIC is as follows:

$$\text{BIC} = -2 \log \hat{\mathcal{L}} + k + \log(N) \tag{13}$$

The order of the mixture m above is three:

```
mixtureOrder = 3;
```

and we have k = weights + means + standard deviations:

$$k = m + \ m + m = 3\, m \tag{14}$$

```
bicℰ = -2 LogLikelihood[ℰ, sample] + 3 mixtureOrder + Log[Length[sample]]
```

```
3380.53
```

As we can see the BIC for the univariate Normal distribution $\mathcal{N}$ is much greater than that of the finite mixture of Normal distributions, $\mathcal{E}$; hence, based on the BIC we would prefer $\mathcal{E}$ to $\mathcal{N}$.

```
bic𝒩 = -2 LogLikelihood[𝒩, sample] + 2 + Log[Length[sample]]
```

```
3535.14
```

It is left as an exercise for the student to compute the BIC for values of $m \in \{1, 2, 3, 4\}$ and plot the BIC as a function of m.

As an aside, can you determine how the code below correctly counts the number of parameters, 9 and 2, in each case?

```
Length@Flatten[(List @@ # &) //@ ε]
Length@Flatten[(List @@ # &) //@ N]
```

```
9
```

```
2
```

Armed with above we can write a general function for computing the BIC and demonstrate that we get results identical to our "manual" computations above.

```
calcBIC =
  {dist, data} ↦ -2 LogLikelihood[dist, data] + Length[Flatten[(List @@ # &) //@ dist]] +
    Log[Length[data]];

calcBIC[ε, sample]
calcBIC[N, sample]
```

```
3380.53
```

```
3535.14
```

The Akaike Information Criterion (AIC)

The Akaike information criterion (AIC) is another penalized likelihood measure that can used the same manner as the BIC. *As with the BIC, the model with the lowest AIC is preferred*.

As before the log $\hat{\mathcal{L}}$ is the log likelihood and k is the number of independent parameters estimated. The AIC does not consider sample size. Its computation is:

$$\text{AIC} = 2\,k - 2\log\hat{\mathcal{L}} \tag{15}$$

And performing computations:

```
calcAIC = {dist, data} ↦ 2 Length[Flatten[(List @@ # &) //@ dist]] -
    2 LogLikelihood[dist, data];

calcAIC[ε, sample]
calcAIC[N, sample]
```

```
3382.62
```

```
3530.23
```

Again the 3-component finite Normal mixture is clearly preferred.

General Warnings

Penalized likelihood functions do not provide a goodness-of-fit test in the same way a null hypothesis test does. In the case of the BIC in particular and unlike a statistic such as the r–squared, we cannot look at a number and assess whether it is a "good" (null hypothesis not rejected) or "not good enough" (null hypothesis rejected) fit but rather whether it is a "better" or "worse" than a competing model. Comparisons across different different datasets do not give us a useful comparison and are meaningless. Comparisons across different forms of penalized likelihood, *e.g.*, the BIC and AIC, are also meaningless even when applied to the same model and sample.

Thus, even after we have used the BIC, AIC or similar measure to help us settle on model complexity, we also need to apply other techniques to decide if our model is at all effective.

There are many other penalized likelihood functions. The BIC is considered slightly more conservative than the AIC, although that does not mean it is the best choice in every case. Further references can be found in the Wikipedia links in the References.

References

[Dempster *et al*., 1977] is the seminal paper which formalized the EM algorithm although its development was foreshadowed by the work of many earlier researchers who used similar techniques to handle specific problems involving occult data.

[McLachlan *et al*., 2000] covers finite mixture models in general, including the estimation of more complex mixtures, controlling numerical stability, and specialized algorithms. [McLachlan *et al*., 2008] provides a broad overview of recent developments in the EM algorithm, including coverage a wide array of problems and the development of many algorithmic improvements.

The "Bayesian information criterion" in Wikipedia gives an overview of penalized likelihood in general and the BIC in specific; it is a good place to start to learn more about this important topic. The "Akaike information criterion" in Wikipedia describes this alternative to the BIC.

Finite mixtures are one example from the larger class of mixture distributions. Mixture distributions may also be formed from a countably infinite number of discrete cases or from infinite mixtures where the parameters of the component distributions are themselves characterized by a continuous density.

- "Akaike information criterion", *Wikipedia*, http://en.wikipedia.org/wiki/Akaike_information_criterion, retrieved 2015-Mar-20.
- "Bayesian information criterion", *Wikipedia*, http://en.wikipedia.org/wiki/Bayesian_information_criterion, retrieved 2015-Mar-20.
- Dempster, A.P., N. M. Liard, and D. B. Rubin, "Maximum-Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society*, *Series B*, 39, 1977.
- MacLachlan, Geoffrey, and David A. Peel, *Finite Mixture Models*, Wiley, NY, 2000.
- McLachlan, G. J., and T. Krishnan, *The EM Algorithm and Extensions*, *2nd Ed*., Wiley-Interscience, 2008.
- "Mixture distribution", *Wikipedia*, http://en.wikipedia.org/wiki/Mixture_distribution, retrieved 2015-Mar-20.