CS395T
Computational Statistics with Application to Bioinformatics

Prof. William H. Press
Spring Term, 2011
The University of Texas at Austin

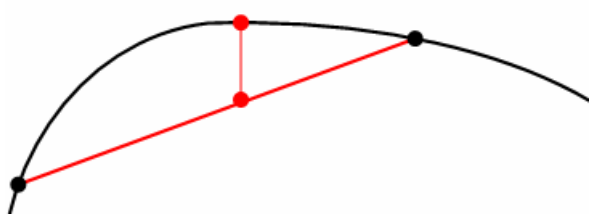
Lecture 17

Let's look at the theory behind EM methods more generally:

Preliminary: Jensen's inequality

If a function is concave (downward), then

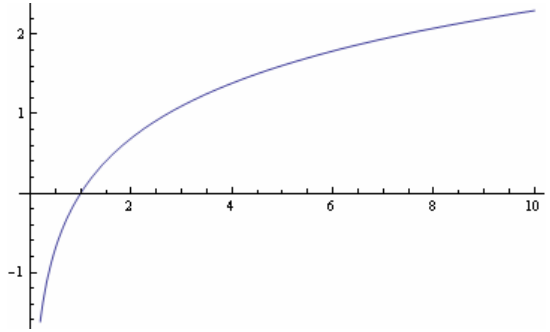
function(interpolation) ≥ interpolation(function)



Log is concave (downward). Jensen's inequality is thus:

$$\text{If} \qquad \sum_i \lambda_i = 1$$

$$\text{Then} \qquad \ln \sum_i \lambda_i Q_i \geq \sum_i \lambda_i \ln Q_i$$



This gets used a lot when playing with log-likelihoods. Proof of the EM method that we now give is just one example.

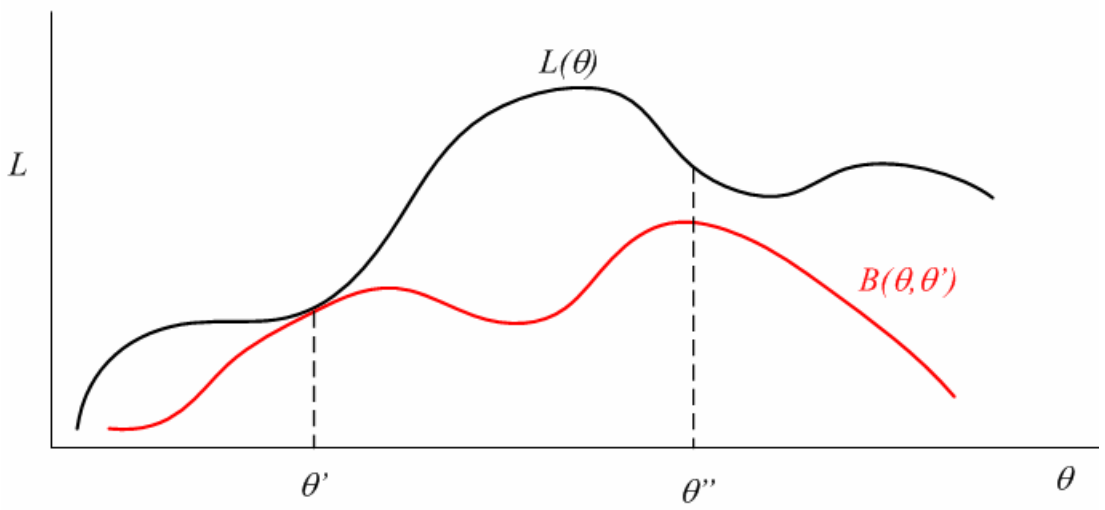
The basic EM theorem:

$\mathbf{x}$ are the data
$\mathbf{z}$ are missing data or nuisance variables
$\boldsymbol{\theta}$ are parameters to be determined

Find $\boldsymbol{\theta}$ that maximizes the log-likelihood of the data:

$$
\begin{aligned}
L(\boldsymbol{\theta}) &\equiv \ln P(\mathbf{x}|\boldsymbol{\theta}) \\
&= \ln \left[\sum_{\mathbf{z}} P(\mathbf{x}|\mathbf{z}\boldsymbol{\theta}) P(\mathbf{z}|\boldsymbol{\theta}) \right] \qquad \text{marginalize over z} \\
&= \ln \left[\sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}\boldsymbol{\theta}') \frac{P(\mathbf{x}|\mathbf{z}\boldsymbol{\theta}) P(\mathbf{z}|\boldsymbol{\theta})}{P(\mathbf{z}|\mathbf{x}\boldsymbol{\theta}')} \right] - \ln P(\mathbf{x}|\boldsymbol{\theta}') + L(\boldsymbol{\theta}') \\
&\geq \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}\boldsymbol{\theta}') \ln \left[\frac{P(\mathbf{x}|\mathbf{z}\boldsymbol{\theta}) P(\mathbf{z}|\boldsymbol{\theta})}{P(\mathbf{z}|\mathbf{x}\boldsymbol{\theta}') P(\mathbf{x}|\boldsymbol{\theta}')} \right] + L(\boldsymbol{\theta}') \\
&= \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}\boldsymbol{\theta}') \ln \left[\frac{P(\mathbf{x}\mathbf{z}|\boldsymbol{\theta})}{P(\mathbf{z}\mathbf{x}|\boldsymbol{\theta}')} \right] + L(\boldsymbol{\theta}') \\
&\equiv B(\boldsymbol{\theta}, \boldsymbol{\theta}') \quad \text{for any } \boldsymbol{\theta}', \text{ a bound on } L(\boldsymbol{\theta})
\end{aligned}
$$

Notice that at $\boldsymbol{\theta} = \boldsymbol{\theta}'$ we have $L(\boldsymbol{\theta}) = L(\boldsymbol{\theta}')$,
so the bound touches the actual likelihood:



So, if we maximize $B(\boldsymbol{\theta}, \boldsymbol{\theta}')$ over $\boldsymbol{\theta}$, we are guaranteed that the new max $\boldsymbol{\theta}''$ will increase $L(\boldsymbol{\theta})$. This can terminate only by converging to (at least a local) max of $L(\boldsymbol{\theta})$ (Can you see why?)

And it works whether the maximization step is local or global.

So the general EM algorithm repeats the maximization:

$$\boldsymbol{\theta}'' = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}\boldsymbol{\theta}') \ln \left[\frac{P(\mathbf{xz}|\boldsymbol{\theta})}{P(\mathbf{zx}|\boldsymbol{\theta}')} \right]$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}\boldsymbol{\theta}') \ln \left[P(\mathbf{xz}|\boldsymbol{\theta}) \right]$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}\boldsymbol{\theta}') \ln \left[P(\mathbf{x}|\mathbf{z}\boldsymbol{\theta}) P(\mathbf{z}|\boldsymbol{\theta}) \right]$$

each form is sometimes useful

sometimes (missing data) no dependence on z

sometimes (nuisance parameters) a uniform prior

computing this is the E-step

This is an expectation that can often be computed in some better way than literally integrating over all possible values of z.

maximizing this is the M-step

This is a general way of handling missing data or nuisance parameters if you can estimate the probability of what is missing, given what you see (and a parameters guess).

Might not be instantly obvious how GMM fits this paradigm!

$\mathbf{z}$ (missing) is the assignment of data points to components
$\boldsymbol{\theta}$ consists of the $\boldsymbol{\mu}$s and $\boldsymbol{\Sigma}$s

$$P(\mathbf{z}|\mathbf{x}\boldsymbol{\theta}') \to p_{nk}$$

$$\sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}\boldsymbol{\theta}') \ln\left[P(\mathbf{x}|\boldsymbol{\theta})\right] \to -\sum_{n,k} p_{nk}\left[(\mathbf{x}_n - \boldsymbol{\mu}_k)\cdot\boldsymbol{\Sigma}_k^{-1}\cdot(\mathbf{x}_n - \boldsymbol{\mu}_k) - \ln\det\boldsymbol{\Sigma}_k\right]$$

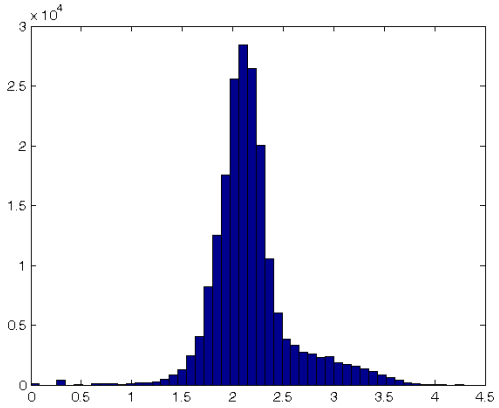
Showing that this is maximized by the previous re-estimation formulas for μ and Σ is a multidimensional (fancy) form of the theorem that the mean is the measure of central tendency that minimizes the mean square deviation.

See Wikipedia: Expectation-Maximization Algorithm for detailed derivation.

The next time we see an EM method will be when we discuss Hidden Markov Models. The "Baum-Welch re-estimation algorithm" for HMMs is an EM method.

Let's come back to the exon-length distribution yet again! Why? Haven't we done everything possible already?

- We binned the data and did χ^2 fitting to a model (2 Gaussians)
  - a.k.a. weighted nonlinear least squares (NLS)
  - it's MLE if experimental errors are normal
  - or in the CLT limit
- We did a 2-component GMM without binning the data
  - GMM is also an MLE
    - but it's specific to Gaussian components
  - good to avoid arbitrary binning when we can
  - EM methods are often computationally efficient for doing MLE that would otherwise be intractable
  - but not all MLE problems can be done by EM methods
- So, what we haven't done yet is MLE on unbinned data for a model more general than GMM
  - nothing magic about it, but it requires general nonlinear optimization
  - for simple models it's a reasonable approach
  - also, for complicated models, it prepares us for MCMC, as we will see



As always, we focus on

$$P(\text{data} \,|\, \text{model parameters}) = P(\mathbf{D}|\boldsymbol{\theta})$$

For a frequentist this is the likelihood of the parameters.
For a Bayesian it is the probability of the parameters (with uniform prior).

MLE is just: $\boldsymbol{\theta} = \text{argmax}_{\boldsymbol{\theta}}\ P(\mathbf{D}|\boldsymbol{\theta})$

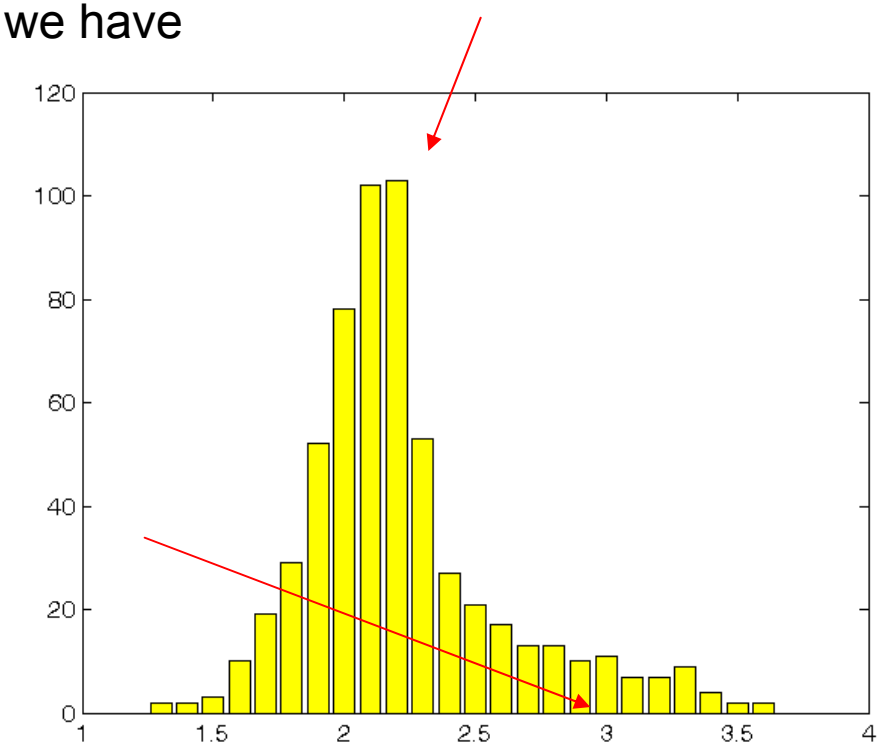
To make binning less tempting, suppose we have much less data than before:

Data is 600 exon lengths:

```
exloglen = log10(cell2mat(g.exonlen));
exsamp = randsample(exloglen,600);
[count cbin] = hist(exsamp,(1:.1:4));
count = count(2:end-1);
cbin = cbin(2:end-1);
bar(cbin,count,'y')
```

and we want to know the fraction of exons in the 2nd component

shown here binned, but the goal now is NOT to bin



Since the data points are independent, the likelihood is the product of the model probabilities over the data points.

The likelihood would often underflow, so it is common to use the log-likelihood. (Also, we'll see that log-likelihood is useful for estimating errors.)

```
function el = twogloglike(c,x)
p1 =      exp(-0.5.*((x-c(1))./c(2)).^2);
p2 = c(3)*exp(-0.5.*((x-c(4))./c(5)).^2);
p = (p1 + p2) ./ (c(2)+c(3)*c(5));
el = sum(-log(p));
```

the log likelihood function

must normalize, but need not keep π's, etc.

need a starting guess

```
c0 = [2.0 .2 .16 2.9 .4];
fun = @(cc) twogloglike(cc,exsamp);
cmle = fminsearch(fun,c0)
```

The maximization is not always as easy as this looks! Can get hung up on local extrema, choose wrong method, etc. Choose starting guess to be closest successful previous parameter set.

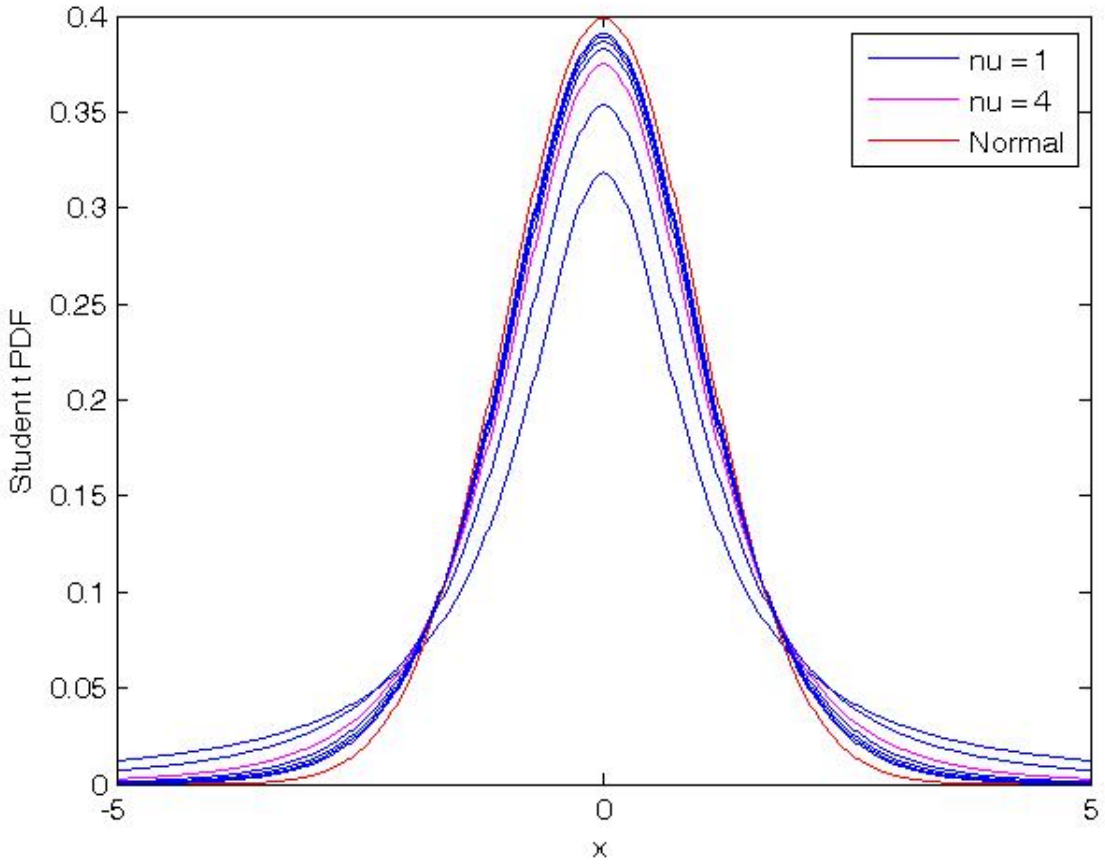
```
cmle =
       2.0959      0.17069      0.18231       2.3994      0.57102

ratio = cmle(2)/(cmle(3)*cmle(5))
ratio =
       1.6396
```

Wow, this estimate of the ratio of the areas is really different from our previous estimate (binned values) of 6.3 +- 0.1 (see Lecture 12)! Is it possibly right?

We could estimate the Hessian (2nd derivative) matrix at the maximum to get an estimate of the error of the ratio. It would come out only ~0.3. Doesn't help.

- The problem is not with MLE, it's with our model!
- Time to face up to the sad truth: this particular data really isn't a mixture of Gaussians
  - it's "heavy tailed"
  - previously, with lots of data, we tried to mitigate this by tail trimming, or maybe a mixture with a broad third component, but now we have only sparse data, so the problem is magnified
- When the data is too sparse to bin, you have to be sure that your model isn't dominated by the tails
  - CLT convergence slow, or maybe not even at all
  - resampling would have shown this
    - e.g., *ratio* would have varied widely over the resamples (try it!)
- Let's try MLE on a heavy-tailed model: use Student-t's instead of Gaussians
  - this adds a parameter ν to the model
  - is it justified? (the problem of model selection)
  - note: Gaussian is limit of $\nu \to \infty$

Student(ν) is a convenient empirical parameterization of heavy-tailed models



Two Student-t's instead of two Gaussians:

```
function el = twostudloglike(c,x,nu)
p1 =      tpdf((x-c(1))./c(2),nu);
p2 = c(3)*tpdf((x-c(4))./c(5),nu);
p = (p1 + p2) ./ (c(2)+c(3)*c(5));
el = sum(-log(p));

rat = @(c) c(2)/(c(3)*c(5));

fun10 = @(cc) twostudloglike(cc,exsamp,10);
ct10 = fminsearch(fun10,cmle)
ratio = rat(ct10)
ct10 =
       2.0984      0.18739      0.11091        2.608      0.55652
ratio =
       3.0359

fun4 = @(cc) twostudloglike(cc,exsamp,4);
ct4 = fminsearch(fun4,cmle)
ratio = rat(ct4)
ct4 =
       2.1146      0.19669     0.089662        3.116       0.2579
ratio =
        8.506

fun2 = @(cc) twostudloglike(cc,exsamp,2);
ct2 = fminsearch(fun2,ct4)
ratio = rat(ct2)
ct2 =
       2.1182      0.17296     0.081015        3.151      0.19823
ratio =
        10.77
```

Evidently, the answer is sensitive to the assumed value of ν.
Therefore, we should let ν be a parameter in the model, and fit for it, too:

```
function el = twostudloglikenu(c,x)
p1 =      tpdf((x-c(1))./c(2),c(6));
p2 = c(3)*tpdf((x-c(4))./c(5),c(6));
p = (p1 + p2) ./ (c(2)+c(3)*c(5));
el = sum(-log(p));

ctry = [ct4 4];
funu= @(cc) twostudloglikenu(cc,exsamp);
ctnu = fminsearch(funu,ctry)
rat(ctnu)
ctnu =
       2.1141      0.19872     0.090507       3.1132      0.26188       4.2826
ans =
       8.3844
```

Model Selection:
Should we <u>really</u> add ν as a model parameter? Adding a parameter to the model always makes it better, so model selection depends on <u>how much</u> the (minus) log-likelihood decreases:

You might have thought that model selection was a settled issue in statistics, but it isn't at all!

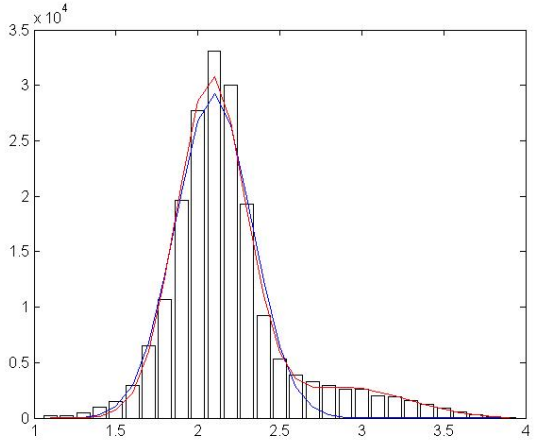
Example (too idealized): Suppose a model with q parameters is correct. So its χ^2 has N-q degress of freedom and has expectation (N-q) ± sqrt[2(N-q)]

If we now add r unnecessary parameters, it is still a "correct" model, but now χ^2 has the smaller expectation N-q-r. "Decreases by 1 for each added parameter".

But this is overfitting. For fixed MLE parameter values, we would not expect a new data set to fit as well. We fit r parameters of randomness, not of signal.

This suggests that it's ok to add parameters as long as $\Delta\chi^2 >> 1$. But…

Example (real life): Suppose a model with q parameters fits badly and is thus incorrect. As we start adding parameters (and decreasing χ^2), we are rightly looking at the data residuals to pick good ones. But it would not be surprising if we can get $\Delta\chi^2 > 1$ even when overfitting and exploiting particular randomness.



The real problem here is that we don't have a theory of the accessible "space of models", or of why model simplicity is itself an indicator of model correctness (a deep philosophical question!)

Various model selection schemes make different trade-offs between goodness of fit and model simplicity

Akaiki information criterion (AIC): $\mathrm{AIC}: \Delta L > 1$ (Recall that $\Delta L = \frac{1}{2}\,\Delta\chi^2$.)

Bayes information criterion (BIC): $\mathrm{BIC}: \Delta L > \frac{1}{2}\ln N \approx 3.2$

Google for "AIC BIC" and you will find all manner of unconvincing explanations!

Note how this is all a bit subjective, since it often depends on what value of ν you started with as the "natural" value.

One alternative approach (e.g., in machine learning) is to use purely empirical approaches to test for overfitting as you add parameters.

Bootstrap resampling, leave-one-out, k-fold cross-validation, etc.

Some empirical check on a model is always a good idea!

A more radical alternative approach to model selection is "large Bayesian models" with lots of parameters, but calculating the full posterior distribution of all the parameters.

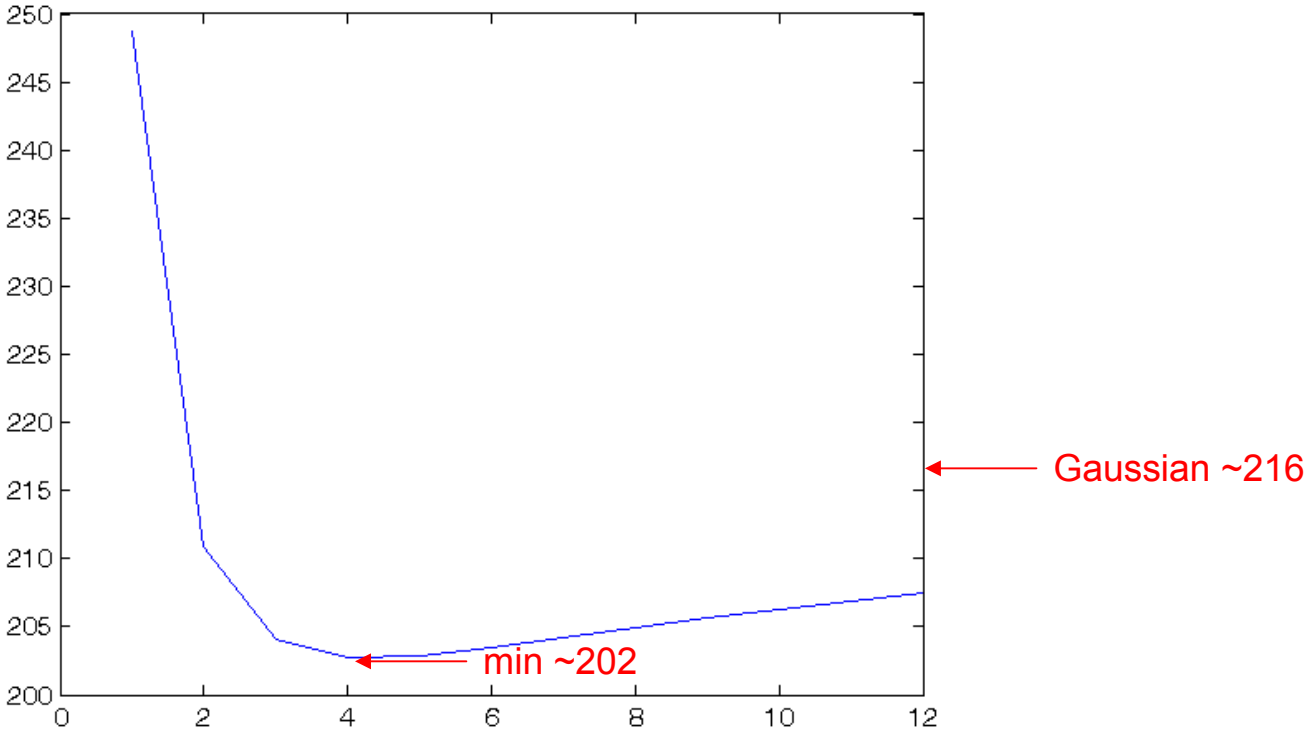
"Marginalization" as a substitute for "model simplicity". Less emphasis on goodness-of-fit (we have seen this before in Bayesian viewpoints).

Can impose additional simplicity constraints "by hand" through (e.g., massed) priors.

MCMC is the computational technique that makes such models feasible.

Back in our Student t example, how does the log-likelihood change with ν?

```
ff = @(nu) twostudloglike( ...
    fminsearch(@(cc) twostudloglike(cc,exsamp,nu),ct4) ...
    ,exsamp,nu);
plot(1:12,arrayfun(ff,1:12))
```



So here, both AIC and BIC tell us to add ν as an MLE parameter: