

**CS395T**  
**Computational Statistics with**  
**Application to Bioinformatics**

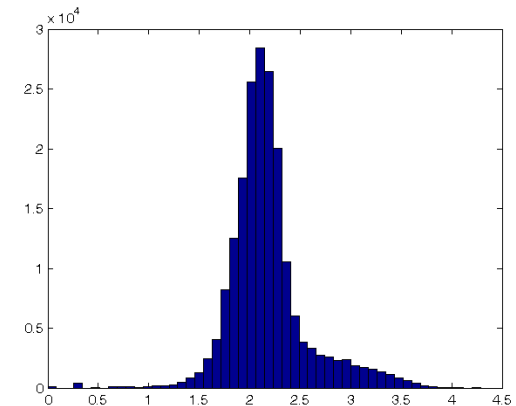
Prof. William H. Press  
Spring Term, 2009

The University of Texas at Austin

**Unit 15: Maximum Likelihood Estimation**

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

- We bin the data and did  $\chi^2$  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

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

For a frequentist this is the likelihood of the parameters.

For a Bayesian it is a factor in the probability of the parameters (needs the prior).

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

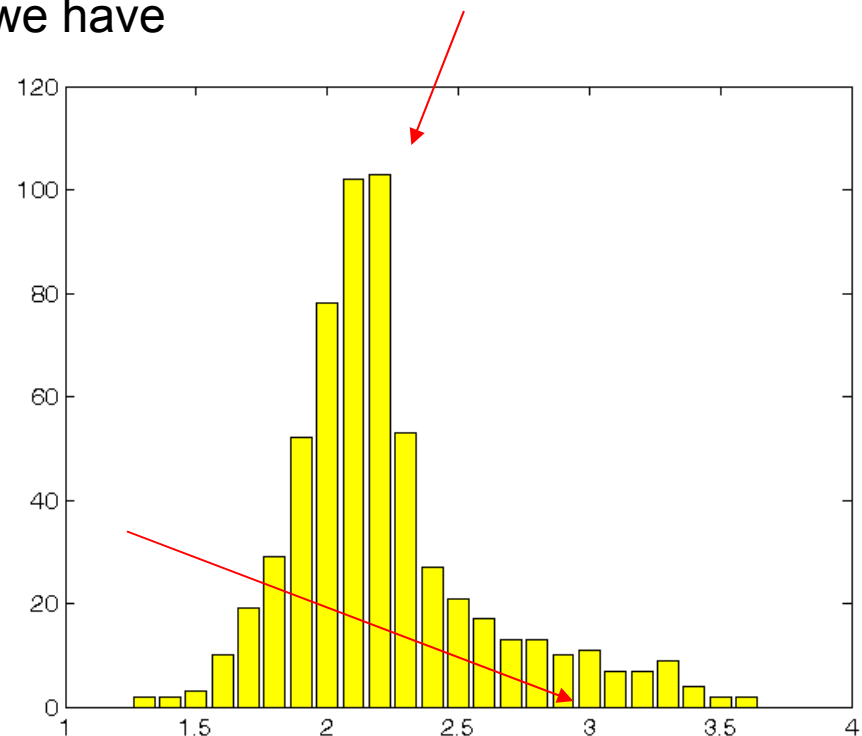
To make binning less tempting, suppose we have much less data than before:

Data is 600 exon lengths:

```
exl ogl en = log10(cel l 2mat(g. exonl en));  
exsamp = randsamp l e(exl ogl en, 600);  
[count cbi n] = hi st(exsamp, (1: . 1: 4));  
count = count(2: end-1);  
cbi n = cbi n(2: end-1);  
bar(cbi n, count, ' y' )
```

and we want to know the fraction of exons in the 2<sup>nd</sup> 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  $\pi$ 's, etc.

need a starting guess

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

```
cml e =
    2.0959    0.17069    0.18231    2.3994    0.57102
```

```
ratio = cml e(2)/(cml e(3)*cml e(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 (2<sup>nd</sup> 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 we tried to mitigate this by tail trimming, but it has now caught up with us
- 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  $\nu$  to the model
  - is it justified? (the problem of model selection)
  - note: Gaussian is limit of  $\nu \rightarrow \infty$

## 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, cml e)
```

```
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, cml e)
```

```
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  $v$ .  
 Therefore, we should let  $v$  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 really add  $v$  as a model parameter? Adding a parameter to the model always makes it better, so model selection depends on how much 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  $\chi^2$  has  $N-q$  degrees of freedom and has expectation  $(N-q) \pm \sqrt{2(N-q)}$

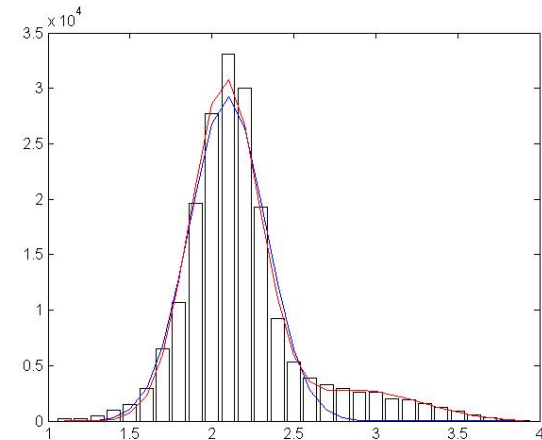
If we now add  $r$  unnecessary parameters, it is still a “correct” model, but now  $\chi^2$  has the smaller expectation  $N-q-r$ . “Decreases by 1 for each added parameter”.

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  $\Delta\chi^2 \gg 1$ . But...

Example (real life): Suppose a model with  $q$  parameters fits badly and is thus incorrect. As we start adding parameters (and decreasing  $\chi^2$ ), 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

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

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

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  $\nu$  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.

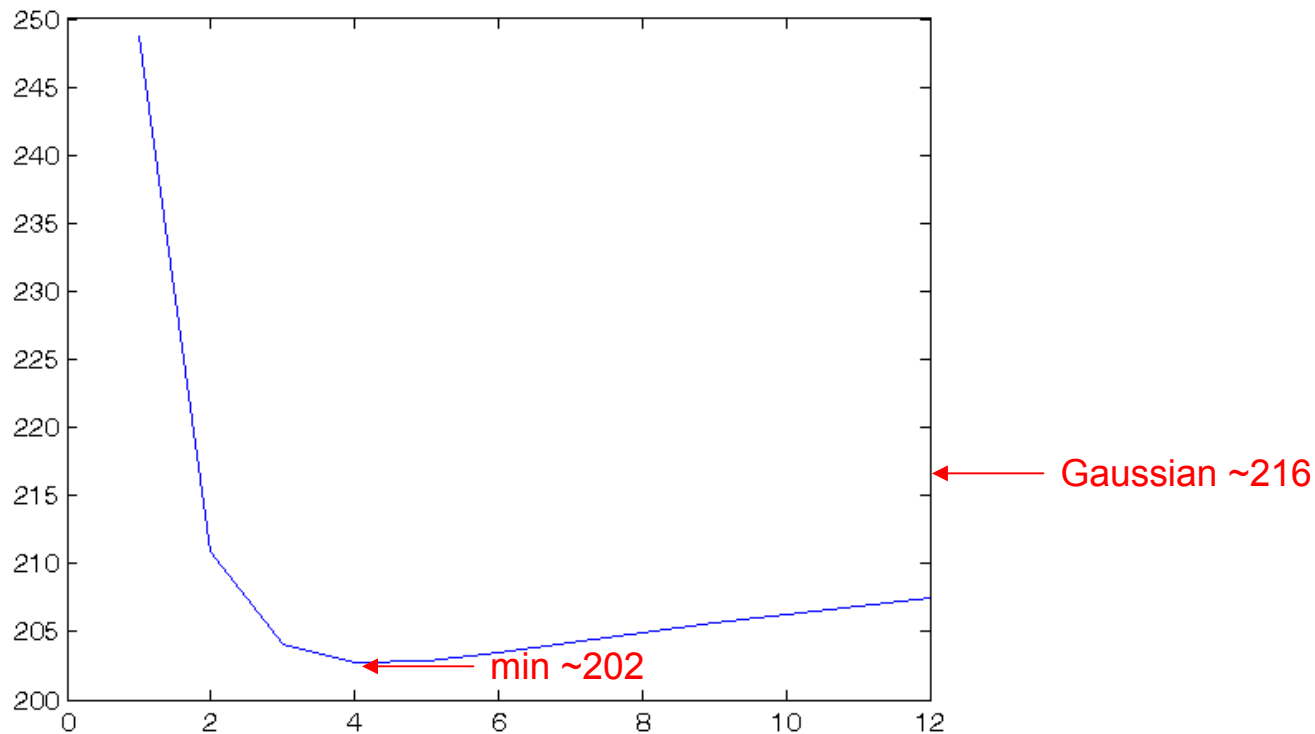
“Marginalization” as a substitute for “model simplicity”. Less emphasis on goodness-of-fit (we have seen this before in Bayesian viewpoints).

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  $\nu$ ?

```
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  $\nu$  as an MLE parameter: