

Computational Statistics with Application to Bioinformatics

Prof. William H. Press
Spring Term, 2008
The University of Texas at Austin

Unit 12: Maximum Likelihood Estimation (MLE) on a Statistical Model

Unit 12: Maximum Likelihood Estimation (MLE) on a Statistical Model (Summary)

- Review examples of MLE that we have already seen
- Write a likelihood function for fitting the exon length distribution to two Gaussians
 - find parameters that maximize it
 - note that we get quite a different answer from when we binned the data – why?
- Introduce the Fisher Information Matrix for calculating MLE parameter uncertainties
 - we write our own Hessian routine for Matlab
 - see that our different answer (above) has small uncertainties
- Discuss the sensitivity of multiple Gaussian models to outliers
 - when we binned the data, we in effect chopped off the outliers
 - it's the outliers that are changing the answer
- A better model for capturing the outliers uses Student-t's
 - new parameter is ν ; should we add it as a parameter?
 - AIC and BIC both say yes
 - now we get an answer much closer to when we binned the data

Direct Maximum Likelihood Estimation (MLE) of the parameters in a statistical model

- We've already done MLE twice
 - weighted nonlinear least squares (NLS)
 - it's "exactly" MLE if experimental errors are normal
 - or in the CLT limit
 - GMMs
 - 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
- But neither example was a general case
 - In our NLS example, we binned the data
 - the data was a sampled distribution (exon lengths)
 - binning converts sampled distribution data to (x_i, y_i, σ_i) data points
 - which NLS easily digests
 - but binning, in general, loses information
 - In our GMM example, we had to use a Gaussian model
 - by definition
- Let's do an example where we apply MLE directly to the raw data
 - for simple models it's a reasonable approach
 - also, for complicated models, it prepares us for MCMC

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).

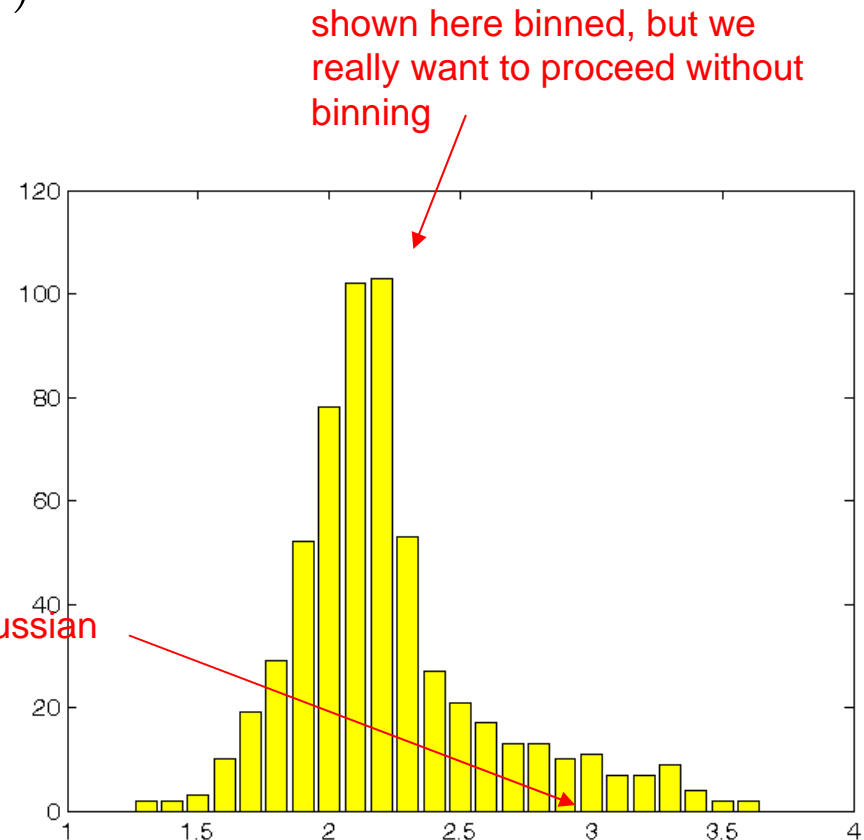
MLE is just: $\boldsymbol{\theta} = \operatorname{argmax}_{\boldsymbol{\theta}} P(\mathbf{D} \mid \boldsymbol{\theta})$

Let's go back to the exon length example:

Suppose we had only 600 exon lengths:

```
exl ogl en = log10(cel l 2mat(g. exonl en));  
exsamp = rand sample(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 2nd Gaussian



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);
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! Is it possibly right? How do we get an error estimate? Can something else go wrong?

Theorem: The expected value of the second derivatives of the log-likelihood is the inverse matrix to the covariance of the estimated parameters. (A mouthful!)

Wasserman
All of Statistics

9.10 Multiparameter Models

These ideas can directly be extended to models with several parameters. Let $\theta = (\theta_1, \dots, \theta_k)$ and let $\hat{\theta} = (\hat{\theta}_1, \dots, \hat{\theta}_k)$ be the MLE. Let $\ell_n = \sum_{i=1}^n \log f(X_i; \theta)$,

$$H_{jj} = \frac{\partial^2 \ell_n}{\partial \theta_j^2} \quad \text{and} \quad H_{jk} = \frac{\partial^2 \ell_n}{\partial \theta_j \partial \theta_k}.$$

Define the **Fisher Information Matrix** by

$$I_n(\theta) = - \begin{bmatrix} \mathbb{E}_\theta(H_{11}) & \mathbb{E}_\theta(H_{12}) & \cdots & \mathbb{E}_\theta(H_{1k}) \\ \mathbb{E}_\theta(H_{21}) & \mathbb{E}_\theta(H_{22}) & \cdots & \mathbb{E}_\theta(H_{2k}) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbb{E}_\theta(H_{k1}) & \mathbb{E}_\theta(H_{k2}) & \cdots & \mathbb{E}_\theta(H_{kk}) \end{bmatrix}. \quad (9.18)$$

Let $J_n(\theta) = I_n^{-1}(\theta)$ be the inverse of I_n .

9.27 Theorem. Under appropriate regularity conditions,

$$(\hat{\theta} - \theta) \approx N(0, J_n).$$

Also, if $\hat{\theta}_j$ is the j^{th} component of $\hat{\theta}$, then

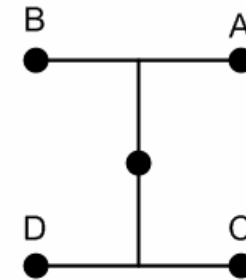
$$\frac{(\hat{\theta}_j - \theta_j)}{\widehat{\text{se}}_j} \rightsquigarrow N(0, 1) \quad (9.19)$$

where $\widehat{\text{se}}_j^2 = J_n(j, j)$ is the j^{th} diagonal element of J_n . The approximate covariance of $\hat{\theta}_j$ and $\hat{\theta}_k$ is $\text{Cov}(\hat{\theta}_j, \hat{\theta}_k) \approx J_n(j, k)$.

We need a numerical Hessian (2nd derivative) function.
Centered 2nd difference is good enough:

```
cml e =
    2. 0959      0. 17069      0. 18231      2. 3994      0. 57102
```

```
function h = hessian(fun, c, del)
h = zeros(numel(c));
for i=1: numel(c)
    for j=1: i
        ca = c; ca(i) = ca(i)+del; ca(j) = ca(j)+del;
        cb = c; cb(i) = cb(i)-del; cb(j) = cb(j)+del;
        cc = c; cc(i) = cc(i)+del; cc(j) = cc(j)-del;
        cd = c; cd(i) = cd(i)-del; cd(j) = cd(j)-del;
        h(i, j) = (fun(ca)+fun(cd)-fun(cb)-fun(cc))/(4*del ^2);
        h(j, i) = h(i, j);
    end
end
```



not immediately obvious that this coding is correct for the case $i=j$, but it is (check it!)

```
covar = inv(hessian(fun, cml e, .001))
stdev = sqrt(diag(covar))
covar =
    0. 00013837  -3. 8584e-006  -2. 8393e-005  -9. 2711e-005  5. 7889e-005
   -3. 8584e-006   0. 00016515  -0. 00013811   0. 00029641   0. 00010633
   -2. 8393e-005  -0. 00013811   0. 0010754  -0. 00076202  -0. 00064362
   -9. 2711e-005   0. 00029641  -0. 00076202   0. 0025821   0. 00036249
    5. 7889e-005   0. 00010633  -0. 00064362   0. 00036249   0. 00098611
stdev =
    0. 011763
    0. 012851
    0. 032793
    0. 050814
    0. 031402
```

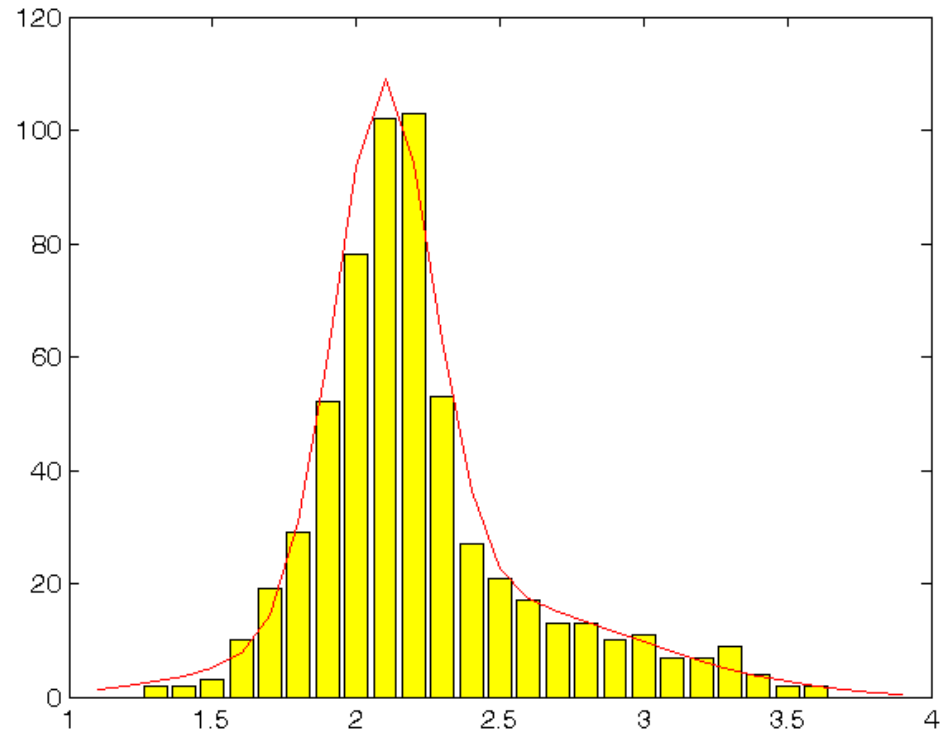
these are the standard errors of the individual fitted parameters around *cml e*

Linear propagation of errors, as we did before:

```
syms c1 c2 c3 c4 c5 real
func = c2/(c3*c5);
c = [c1 c2 c3 c4 c5];
symgrad = jacobian(func, c)';
grad = double(subs(symgrad, c, cml e));
mu = double(subs(func, c, cml e))
sigma = sqrt(grad' * covar * grad)
mu =
    1.6396
sigma =
    0.30839
bml e = [1 cml e];
factor = 94;
plot(cbin, factor*model twog(bml e, cbin), 'r')
hold off
```

Although the sample is much smaller than that for the binned fit, that's not the problem. The MLE really does give a much smaller ratio than the binned fit for this data set, even if you use the full set.

How can this happen?



- Fitting Gaussians is very sensitive to data on the tails!
- We reduced this sensitivity when we binned the data
 - by truncating first and last bin
 - by adding a pseudocount to the error estimate in each bin
- Here, MLE expands the width of the 2nd component to capture the tail, thus severely biasing *ratio*
- 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!)
 - a fix is to use a heavy-tailed distribution, like Student-t
- Despite the pitfalls in this example, MLE is generally superior binning
 - just don't do it uncritically
 - and think hard about whether your model has enough parameters to capture the data features (here, tails)

So, let's try a model with 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));

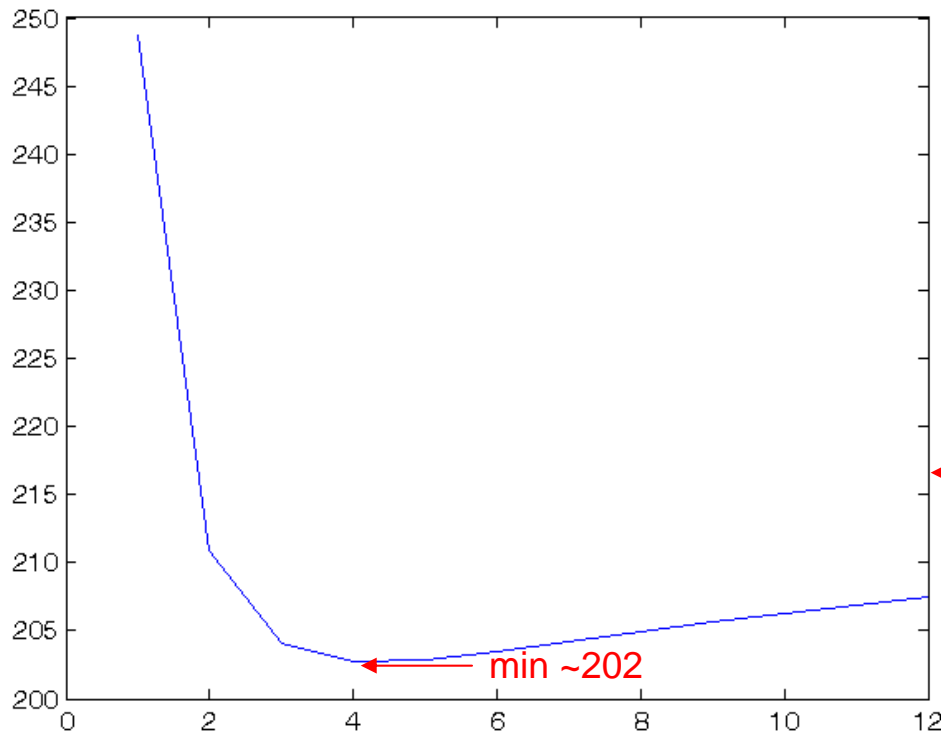
rat = @(c) c(2)/(c(3)*c(5));
fun4 = @(cc) twostudloglike(cc, exsamp, 4);
ct4 = fminsearch(fun4, cml e)
ratio = rat(ct4)
ct4 =
    2.1146    0.19669    0.089662    3.116    0.2579
ratio =
    8.506

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

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
```

How does the log-likelihood change with v ?

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



Model Selection:

Should we add v as a model parameter?
Adding a parameter to the model always makes it better, so it depends on how much the $-\log$ -likelihood decreases:

Bayes information criterion:

$$\text{BIC: } \Delta L > \frac{1}{2} \ln N \approx 3.2$$

Akaike information criterion:

$$\text{AIC: } \Delta L > 1$$

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

← Gaussian ~216

You might have thought that this was a settled issue in statistics, but it isn't at all! Also, it's subjective, since it depends on what value of v you started with as the “natural” value.

(MCMC is the alternative, as we will soon see.)

Here, both AIC and BIC tell us to add v as an MLE parameter:

```
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
```

```
covar = inv(hessian(funu, ctnu, .001))
stdev = sqrt(diag(covar))
```

```
covar =
    0.00011771    6.2065e-006   -4.82e-006     0.000177   -0.00012137   -0.0017633
    6.2065e-006    0.00014127    4.6935e-005    0.00013538  -2.6128e-005    0.0073335
   -4.82e-006    4.6935e-005    0.00029726    4.0581e-005  -0.00036269    0.0031577
    0.000177     0.00013538    4.0581e-005    0.0036802   -0.0014988   -0.0098572
  -0.00012137  -2.6128e-005  -0.00036269   -0.0014988    0.0021267     0.013769
  -0.0017633    0.0073335     0.0031577   -0.0098572    0.013769     1.0745
```

```
stdev =
    0.010849
    0.011886
    0.017241
    0.060665
    0.046116
    1.0366
```

Editorial: You should take away a certain humbleness about the formal errors of model parameters. The real errors includes subjective “choice of model” errors. You should quote ranges of parameters over plausible models, not just formal errors for your favorite model. Bayes, we will see, can average over models. This improves things, but only somewhat.