Numerical Maximum Likelihood[1]

STA 312: Fall 2012

[1] See last slide for copyright information.

Overview

Maximum Likelihood

General framework

$$Y_1, \ldots, Y_n \stackrel{i.i.d.}{\sim} F_\beta,\ \beta \in \mathcal{B}$$
$$\ell(\beta) = \prod_{i=1}^n f(y_i;\beta)$$
$$L(\beta) = \log \ell(\beta) = \sum_{i=1}^n \log f(y_i;\beta)$$

- The maximum likelihood estimate is the parameter value that makes the likelihood as great as possible.
- That is, it maximizes the probability of observing the data we did observe.

Close your eyes and differentiate?

- Often, can differentiate the log likelihood with respect to the parameter, set the derivative to zero, and solve.
- But it does not always work.
- Consider a gamma distribution with parameter $\alpha > 0$ unknown, and $\beta = 1$.

$$f(y|\alpha) = \frac{1}{\Gamma(\alpha)} e^{-y} y^{\alpha-1}$$

Gamma log likelihood

$f(y|\alpha) = \frac{1}{\Gamma(\alpha)} e^{-y} y^{\alpha-1}$

$$
\begin{aligned}
\log \ell(\alpha) &= \log \prod_{i=1}^{n} \frac{1}{\Gamma(\alpha)} e^{-y_i} y_i^{\alpha-1} \\
&= \log \left(\Gamma(\alpha)^{-n} \exp\{-\sum_{i=1}^{n} y_i\} \left(\prod_{i=1}^{n} y_i \right)^{\alpha-1} \right) \\
&= -n \log \Gamma(\alpha) - \sum_{i=1}^{n} y_i + (\alpha - 1) \sum_{i=1}^{n} \log y_i
\end{aligned}
$$

Differentiate the log likelihood?

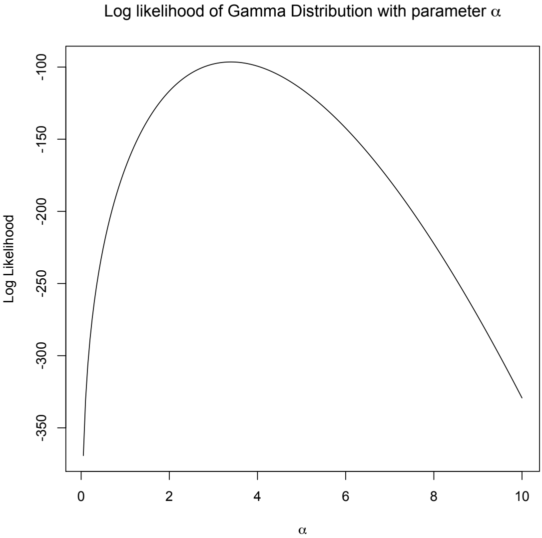
Differentiate

$$\begin{aligned}
\frac{\partial \log \ell(\alpha)}{\partial \alpha} &= \frac{\partial}{\partial \alpha}\left(-n \log \Gamma(\alpha) - \sum_{i=1}^{n} y_i + (\alpha - 1)\sum_{i=1}^{n} \log y_i\right) \\
&= -n \frac{\Gamma'(\alpha)}{\Gamma(\alpha)} + \sum_{i=1}^{n} \log y_i \\
&\stackrel{\text{set}}{=} 0 \\
&\Leftrightarrow \frac{\Gamma'(\alpha)}{\Gamma(\alpha)} = \frac{1}{n}\sum_{i=1}^{n} \log y_i
\end{aligned}$$

- Good luck trying to solve this for α.
- But given some data, we can still find the MLE.

```
> y
 [1] 4.51 4.01 3.34 1.43 3.62 0.37 2.21 2.01 4.56 2.15 9.24 3.08 2.27 3.29 1.75 3.38 2.72
[18] 2.73 5.09 3.81 1.50 4.15 1.97 2.90 2.32 6.65 2.30 2.29 1.01 5.52 3.03 1.85 2.51 6.92
[35] 3.67 2.10 2.50 9.27 2.40 2.96 0.96 3.15 1.30 4.04 2.40 5.49 2.42 6.75 5.42 4.75
```

We can plot the log likelihood



Grid search

```
> # Max seems to be between 2 and 4. Where is it?
> a[LL==max(LL)]
[1] 3.4
> # That's rough: Grid search
> alpha = seq(from=3.35,to=3.45,by=0.01)
> GLL = function(alpha,data) # Gamma Log Likelihood
+       {
+       n = length(data)
+       GLL = -n*lgamma(alpha) - sum(data) + (alpha-1)*sum(log(data))
+       GLL
+       } # End function GLL
> loglike = GLL(alpha,data=y)
> cbind(alpha,loglike)
      alpha   loglike
 [1,]  3.35 -96.47796
 [2,]  3.36 -96.47072
 [3,]  3.37 -96.46521
 [4,]  3.38 -96.46143
 [5,]  3.39 -96.45936
 [6,]  3.40 -96.45901
 [7,]  3.41 -96.46037
 [8,]  3.42 -96.46343
 [9,]  3.43 -96.46818
[10,]  3.44 -96.47462
```

Numerical minimization

Most numerical optimization functions like to minimize things.

```
> # Numerical minimization (define a new function)
> mGLL = function(alpha,data) # Minus gamma Log Likelihood
+   {mGLL = -1*GLL(alpha,data); mGLL}
> start = mean(y); start # Could start at 3.4
[1] 3.4014
> gamsearch = nlm(mGLL,p=start,data=y) # p is parameter starting value
> gamsearch
$minimum
[1] 96.45894

$estimate
[1] 3.397055

$gradient
[1] 5.019944e-09

$code
[1] 1

$iterations
[1] 3
```

Is the derivative zero at the MLE?

$$\frac{\partial \log \ell(\alpha)}{\partial \alpha} = -n \frac{\Gamma'(\alpha)}{\Gamma(\alpha)} + \sum_{i=1}^{n} \log y_i$$

```
> # Does the derivative equal zero at the MLE?
> n = length(y)
> alphahat = gamsearch$estimate
> -n * digamma(alphahat) + sum(log(y))
[1] -1.163801e-07
```

Logistic regression

$$\log\left(\frac{\pi}{1-\pi}\right) = \mathbf{x}'\boldsymbol{\beta} \;\Leftrightarrow\; \pi = \frac{e^{\mathbf{x}'\boldsymbol{\beta}}}{1+e^{\mathbf{x}'\boldsymbol{\beta}}}$$

Log likelihood

$$
\begin{aligned}
\log \ell(\boldsymbol{\beta}) &= \log \prod_{i=1}^{n} \left(\frac{e^{\mathbf{x}_i'\boldsymbol{\beta}}}{1+e^{\mathbf{x}_i'\boldsymbol{\beta}}} \right)^{y_i} \left(1 - \frac{e^{\mathbf{x}_i'\boldsymbol{\beta}}}{1+e^{\mathbf{x}_i'\boldsymbol{\beta}}} \right)^{1-y_i} \\
&= \log \prod_{i=1}^{n} \left(\frac{e^{\mathbf{x}_i'\boldsymbol{\beta}}}{1+e^{\mathbf{x}_i'\boldsymbol{\beta}}} \right)^{y_i} \left(\frac{1}{1+e^{\mathbf{x}_i'\boldsymbol{\beta}}} \right)^{1-y_i} \\
&= \log \prod_{i=1}^{n} \frac{e^{y_i\mathbf{x}_i'\boldsymbol{\beta}}}{1+e^{\mathbf{x}_i'\boldsymbol{\beta}}} \\
&= \log \frac{\exp\{\sum_{i=1}^{n} y_i\mathbf{x}_i'\boldsymbol{\beta}\}}{\prod_{i=1}^{n}(1+e^{\mathbf{x}_i'\boldsymbol{\beta}})} \\
&= \sum_{i=1}^{n} y_i\mathbf{x}_i'\boldsymbol{\beta} - \sum_{i=1}^{n} \log\left(1+e^{\mathbf{x}_i'\boldsymbol{\beta}}\right)
\end{aligned}
$$

Specialize to simple logistic regression

One explanatory variable

$$
\begin{aligned}
\log \ell(\boldsymbol{\beta}) &= \sum_{i=1}^{n} y_i \mathbf{x}_i' \boldsymbol{\beta} - \sum_{i=1}^{n} \log\left(1 + e^{\mathbf{x}_i' \boldsymbol{\beta}}\right) \\
&= \sum_{i=1}^{n} y_i(\beta_0 + \beta_1 x_i) - \sum_{i=1}^{n} \log\left(1 + e^{\beta_0 + \beta_1 x_i}\right) \\
&= \beta_0 \sum_{i=1}^{n} y_i + \beta_1 \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} \log\left(1 + e^{\beta_0 + \beta_1 x_i}\right)
\end{aligned}
$$

Differentiate, obtaining two equations in two unknowns

$$\log \ell(\boldsymbol{\beta}) = \beta_0 \sum_{i=1}^{n} y_i + \beta_1 \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} \log\left(1 + e^{\beta_0 + \beta_1 x_i}\right)$$

$$\begin{aligned} \frac{\partial L}{\partial \beta_0} &= \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} \\ \frac{\partial L}{\partial \beta_1} &= \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} \frac{x_i e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} \end{aligned}$$

Set to zero and solve.

Given some data, can compute the minus log likelihood

$\log \ell(\boldsymbol{\beta}) = \beta_0 \sum_{i=1}^n y_i + \beta_1 \sum_{i=1}^n x_i y_i - \sum_{i=1}^n \log\left(1 + e^{\beta_0+\beta_1 x_i}\right)$

```
LRLL = function(beta,data)
# Log likelihood for simple logistic regression
    {
    beta0 = beta[1]; beta1 = beta[2]
    X = data[,1]; Y = data[,2]; xb = beta0 + beta1*X
    LRLL = beta0*sum(Y) + beta1 * sum(X*Y) - sum(log(1+exp(xb)))
    LRLL
    } # End function LRLL

LRmLL = function(beta,data) # Minus LL
    {LRmLL = -LRLL(beta,data);  LRmLL}


> # Get halfway reasonable starting values
> start = lm(y~x)$coefficients; start
(Intercept)           x
 -1.0480186   0.1533418
```

Minimize numerically

```
> xy = cbind(x,y) # Put data in a matrix
> nlm(LRmLL,p=start,data=xy)
$minimum
[1] 41.43899

$estimate
[1] -11.500810   1.143899

$gradient
[1] -4.089719e-07 -5.713416e-06

$code
[1] 1

$iterations
[1] 19

> # For comparison
> betahat = glm(y~x,family=binomial)$coefficients
> betahat
(Intercept)           x
 -11.500808    1.143899
```

Log-linear models

It's numerical maximum likelihood, but ...

- Does not just walk downhill in the parameter space.
- Goes straight to estimated expected values $\widehat{\mu}_{ijk}$
- Called "Iterative proportional model fitting."

Idea behind Iterative proportional model fitting

- The model specifies certain marginals, meaning marginal tables.
- For example, $(XY)(Z)$ specifies a two-dimensional $X \times Y$ table and a one-dimensional Z table.
- The marginal estimated expected frequencies must match the observed marginal totals for the marginals specified by the model (proved).
- That is, $\widehat{\mu}_{ij+} = n_{ij+}$ and $\widehat{\mu}_{++k} = n_{++k}$.
- Start with complete independence.
- Adjust the cell expected frequencies $\widehat{\mu}_{ijk}$ up or down so as to match the totals of the marginals in the model.
- But don't introduce any *additional* relationships in the process.

Outline of the algorithm

For iterative proportional model fitting

For each marginal in the model

  For each cell in the table (say is the marginal is $ij+$)

    If $\widehat{\mu}_{ij+} < n_{ij+}$, adjust $\widehat{\mu}_{ijk}$ up.

    If $\widehat{\mu}_{ij+} > n_{ij+}$, adjust $\widehat{\mu}_{ijk}$ down.

  Re-calculate all expected marginal totals in the model.

Keep cycling until the expected marginal totals are very close to the observed.

Example: $(XY)(XZ)(YZ)$

Know analytically that

$$\begin{aligned} \widehat{\mu}_{ij+} &= n_{ij+} \\ \widehat{\mu}_{i+k} &= n_{i+k} \\ \widehat{\mu}_{+jk} &= n_{+jk} \end{aligned}$$

Try to make marginals match up

Re-calculate *all* expected marginal totals after each step.

$$
\begin{aligned}
\widehat{\mu}_{ijk}^{(0)} &= 1 \\
\widehat{\mu}_{ijk}^{(1)} &= \frac{n_{ij+}}{\widehat{\mu}_{ij+}^{(0)}} \widehat{\mu}_{ijk}^{(0)} \\
\widehat{\mu}_{ijk}^{(2)} &= \frac{n_{i+k}}{\widehat{\mu}_{i+k}^{(1)}} \widehat{\mu}_{ijk}^{(1)} \\
\widehat{\mu}_{ijk}^{(3)} &= \frac{n_{+jk}}{\widehat{\mu}_{+jk}^{(2)}} \widehat{\mu}_{ijk}^{(2)}
\end{aligned}
$$

See how it's *proportional*?

Now repeat the cycle

Re-calculating all expected marginal totals after each step.

$$\begin{aligned}
\widehat{\mu}_{ijk}^{(4)} &= \frac{n_{ij+}}{\widehat{\mu}_{ij+}^{(3)}}\widehat{\mu}_{ijk}^{(3)} \\
\widehat{\mu}_{ijk}^{(5)} &= \frac{n_{i+k}}{\widehat{\mu}_{i+k}^{(4)}}\widehat{\mu}_{ijk}^{(4)} \\
\widehat{\mu}_{ijk}^{(6)} &= \frac{n_{+jk}}{\widehat{\mu}_{+jk}^{(5)}}\widehat{\mu}_{ijk}^{(5)}
\end{aligned}$$

- Keep repeating until the $\widehat{\mu}_{ijk}$ stop moving.
- They will stop moving; the algorithm converges.
- It converges to the right answer (Proved).

A small numerical example: $(X)(Y)$

```
> obs = rbind(c(10,20),
+             c( 5, 5))
> # See what the expected frequencies should be
> goal = chisq.test(obs)$expected; goal

      [,1]  [,2]
[1,] 11.25 18.75
[2,]  3.75  6.25
>
> namz = c("1","2")
> rownames(obs) = namz; colnames(obs) = namz
>
>
> addmargins(obs)
     1  2 Sum
1   10 20  30
2    5  5  10
Sum 15 25  40
```

Match these marginals: Get started

```
> rowmarg = margin.table(obs,1); rowmarg
 1  2
30 10
> colmarg = margin.table(obs,2); colmarg
 1  2
15 25
>
> # Start with no relationship: Total n does not matter
> muhat = rbind(c(1,1),
+               c(1,1))
> rowhat = margin.table(muhat,1); colhat = margin.table(muhat,2)
```

Step 1

$$\widehat{\mu}_{ijk}^{(1)} = \frac{n_{ij+}}{\widehat{\mu}_{ij+}^{(0)}} \widehat{\mu}_{ijk}^{(0)}$$

```
> # Step 1: Row marginals
> muhat[1,1] = rowmarg[1]/rowhat[1] * muhat[1,1]
> muhat[1,2] = rowmarg[1]/rowhat[1] * muhat[1,2]
> muhat[2,1] = rowmarg[2]/rowhat[2] * muhat[2,1]
> muhat[2,2] = rowmarg[2]/rowhat[2] * muhat[2,2]
> muhat
     [,1] [,2]
[1,]   15   15
[2,]    5    5
> # Re-calculate the marginal totals
> rowhat = margin.table(muhat,1); colhat = margin.table(muhat,2)
```

Step 2

```
> # Step 2: Col marginals
> muhat[1,1] = colmarg[1]/colhat[1] * muhat[1,1]
> muhat[1,2] = colmarg[2]/colhat[2] * muhat[1,2]
> muhat[2,1] = colmarg[1]/colhat[1] * muhat[2,1]
> muhat[2,2] = colmarg[2]/colhat[2] * muhat[2,2]
> muhat
      [,1]  [,2]
[1,] 11.25 18.75
[2,]  3.75  6.25
> # Re-calculate the marginal totals
> rowhat = margin.table(muhat,1); colhat = margin.table(muhat,2)
>
> goal
      [,1]  [,2]
[1,] 11.25 18.75
[2,]  3.75  6.25
>
> # We're done, but the algorithm does not know it yet.
```

Step 3

```
> # Step 3: Row marginals again
> muhat[1,1] = rowmarg[1]/rowhat[1] * muhat[1,1]
> muhat[1,2] = rowmarg[1]/rowhat[1] * muhat[1,2]
> muhat[2,1] = rowmarg[2]/rowhat[2] * muhat[2,1]
> muhat[2,2] = rowmarg[2]/rowhat[2] * muhat[2,2]
> muhat
      [,1]  [,2]
[1,] 11.25 18.75
[2,]  3.75  6.25
> # Now it will stop, because mu-hats did not change.
```

Do it with loglin

```
> muhat
      [,1]  [,2]
[1,] 11.25 18.75
[2,]  3.75  6.25
> loglin(obs,margin=c(1,2),fit=T)$fit
2 iterations: deviation 0
      1     2
1 11.25 18.75
2  3.75  6.25
```

This is remarkable

- Very general.
- No starting values required.
- Can handle very large problems without crashing.
- Parameters may be redundant, but it doesn't matter.
- The expected frequencies are what you want anyway.
- This is typical of *mature* maximum likelihood.
- You get the right answer, but not the way you would think.

Copyright Information