Likelihood and Model Evaluation

Zack Treisman

Spring 2021

Philosophy

The time has come for us to discuss in some detail how our models are being constructed and evaluated.

- The key idea is **likelihood**, the probability that a model would produce the data we observe.

When do we say *likelihood* instead of *probability*?

- The probability $P(\text{data}|\text{model})$ considers the model as fixed, and the data as variable.
- The likelihood $\mathcal{L}(\text{model}|\text{data})$ considers the data to be fixed (they are what has been observed) and the model as variable.

This kind of model-space/ data-space parameter duality is all over the place in math, so it must be kind of important. But this is not a class for that sort of *abstract nonsense*[1].

[1] A technical term. No seriously.

Maximum Likelihood Estimation

To find a parametric model to fit data we need to choose values for the parameters according to some criteria.

- Minimizing mean squared error is ideal if the true error distribution is normal with constant variance.
- Non-normal errors are modeled with a GLM.
- Heterogenous variance can be accounted for by a mixed/ multilevel model. (Coming soon)
- Finding parameters by maximizing the likelihood agrees with least squares when errors are normal and homoscedastic gives good results when they are not.

Computing likelihood

Since likelihoods are just probabilities with a shift in what we consider to be a variable, we have the tools to compute them.

```r
x <- subset(ReedfrogPred,pred=="pred" & density==10 & size=="small")
x
```

```
##    density pred  size surv propsurv
## 13      10 pred small    7      0.7
## 14      10 pred small    5      0.5
## 15      10 pred small    9      0.9
## 16      10 pred small    9      0.9
```

```r
likhd_indiv <- dbinom(x$surv, x$density, p=0.7)
likhd_indiv
```

```
## [1] 0.2668279 0.1029193 0.1210608 0.1210608
```

```r
likhd <- prod(likhd_indiv)
likhd
```

```
## [1] 0.0004024719
```

Optimizing

Given a parameter and a function of that parameter, the `optim` command searches for the value of the parameter that gives the optimal (minimum or maximum) value of the function.

```
likhd_fn <- function(p){prod(dbinom(x$surv, x$density, p=p))}
opt0 <- optim(fn = likhd_fn, par = list(p=0.7),
              control=list(fnscale=-1), #maximize instead of minimize
              method="BFGS") #better for one variable
opt0$par
```

```
##         p
## 0.7499974
```

```
sum(x$surv)/sum(x$density)
```

```
## [1] 0.75
```

Negative log-likelihoods

For computational reasons, it is easier to deal with the logs of likelihood values.

- Products of more than a few numbers between 0 and 1 get really small.
- Logs turn products into sums.

For emotional reasons, it's easier to minimize than to maximize.

- Because gravity?

Replace `prod` with `-sum` and add in `log=TRUE`. Same result.

```r
nll_fn <- function(p){-sum(dbinom(x$surv, x$density, p=p, log = TRUE))}
opt1 <- optim(fn = nll_fn, par = list(p=0.7),
              method="BFGS") #better for one variable
opt1$par
```

```
##         p
## 0.7500001
```

The mle2 command

The bbmle package provides mle2 for maximum likelihood estimates.

```
#library(bbmle)
mle2(nll_fn, start = list(p=0.7), data=x)
```

```
##
## Call:
## mle2(minuslogl = nll_fn, start = list(p = 0.7), data = x)
##
## Coefficients:
##         p
## 0.7500001
##
## Log-likelihood: -7.57
```

MLE agrees with ordinary least squares

```
x<-rnorm(50,5,1)
y<-rnorm(50, 3+2*x,1.5)
lm(y~x)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##       2.937        2.001
```

```
nll_fn <- function(a,b,s) -sum(dnorm(y, a+b*x, s, log = TRUE))
mle2(nll_fn, start = list(a=2,b=3,s=1))
```

```
##
## Call:
## mle2(minuslogl = nll_fn, start = list(a = 2, b = 3, s = 1))
##
## Coefficients:
##        a        b        s
## 2.937374 2.000535 1.387094
##
## Log-likelihood: -87.31
```

MLE is used to fit a GLM

```r
n <- 50; x <- runif(n, min=0, max=5); y <- rpois(n, 2+x)
glm1 <- glm(y~x, family = poisson)
coef(glm1)
```

```
## (Intercept)           x
##   0.6901044   0.3066626
```

- In `mle2` both the negative log-likelihood function and the parameters can be expressed as formulae.

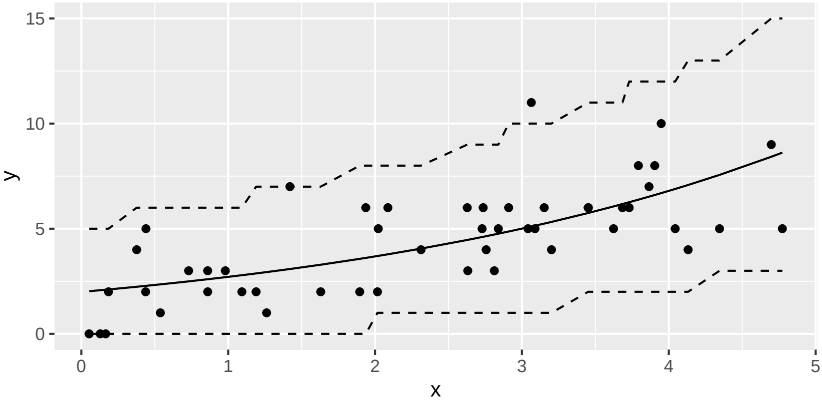
```r
mle2(y~dpois(exp(loglambda)),       ## use log link/exp inverse-link
     data=data.frame(y,x),          ## need to specify as data frame
     parameters=list(loglambda~x),  ## linear model for loglambda
     start=list(loglambda=0))       ## start values for *intercept*
```

```
##
## Call:
## mle2(minuslogl = y ~ dpois(exp(loglambda)), start = list(loglambda = 0),
##     data = data.frame(y, x), parameters = list(loglambda ~ x))
##
## Coefficients:
## loglambda.(Intercept)           loglambda.x
##             0.6901146             0.3066593
##
## Log-likelihood: -99.66
```

A plot of glm1

Always plot your data and models.

```
ggplot(data.frame(x=x, y=y), aes(x,y))+
  geom_point()+
  geom_line(aes(y=exp(predict(glm1))))+
  geom_line(aes(y=qpois(0.025,lambda = exp(predict(glm1)))), linetype=2)+
  geom_line(aes(y=qpois(0.975,lambda = exp(predict(glm1)))), linetype=2)
```



Analysis of a GLM

```r
summary(glm1)
```

```
##
## Call:
## glm(formula = y ~ x, family = poisson)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0484  -0.7530  -0.0464   0.4526   2.2594
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.69010    0.16782   4.112 3.92e-05 ***
## x            0.30666    0.05337   5.746 9.14e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 79.092  on 49  degrees of freedom
## Residual deviance: 43.788  on 48  degrees of freedom
## AIC: 203.32
##
## Number of Fisher Scoring iterations: 5
```

Deviance

In general, the deviance compares any two models. Here, the fitted glm is compared to a model perfectly overfit to the sample.

- This perfect model is known as the saturated model, or the Bayes optimal model.
- The responses $f_{sat}(x_i)$ equal the observed responses y_i, or $f_{sat}(x_i)$ is the mean of the relevant y_i if there are multiple observations with the same x_i.

In code:

$$f_{sat}(x_i) = \texttt{mean}(\texttt{y}[\texttt{x} == \texttt{xi}])$$

Deviance is the difference between the log-likelihoods of the fitted and saturated models. Times -2.

$$D = -2(L(\text{model}) - L_{sat})$$

Deviance acts kind of like variance. Since likelihood is computed as a sum of contributions from each data point, it's easy to split total deviance into *residual deviances*.

Dispersion

For the models of the lily data we computed the ratio of residual deviance to degrees of freedom and compared this ratio to one. If this ratio was greater than one we called the model *overdispersed*.

In the Poisson distribution, the mean equals the variance, which ends up being really restrictive.

Null and residual deviance and how they connect to degrees of freedom

The null deviance is the deviance of the null model

`glm(y~1, family=poisson)`

and the residual deviance is the deviance of the fitted model

`glm(y~x, family=poisson)`.

Dividing by degrees of freedom puts these on the same scale in a "number of ways to alter the model" sense of same.

Likelihood and AIC

Akaike's Information Criterion is the negative log-likelihood plus a penalty term for each coefficient that is estimated in the model. Times 2.

```
L <- logLik(glm1)[1]
L
```

```
## [1] -99.65924
```

```
k <- length(coef(glm1))
k
```

```
## [1] 2
```

```
-2*L+2*k
```

```
## [1] 203.3185
```

```
AIC(glm1)
```

```
## [1] 203.3185
```

Interpreting AIC

Given a set of M different models $\text{mod}_1, \ldots, \text{mod}_M$ write AIC_i for the AIC of the i^{th} model and AIC_{min} for the smallest of these values.

- Define $\Delta_i = AIC_i - AIC_{min}$. These differences in AIC values are more relevant than the actual AICs.
- An estimate of the relative likelihood of model i is $\exp\left(-\frac{\Delta_i}{2}\right)$. These are sometimes called *evidence ratios*.
- Scaling the relative likelihoods by the sum of the relative likelihoods for all models in consideration give the AIC weight.

$$w_i = \frac{\exp\left(-\frac{\Delta_i}{2}\right)}{\sum_{j=i}^{M} \exp\left(-\frac{\Delta_j}{2}\right)}$$

This is roughly interpretable as the probability that the i^{th} model is the best model.

Customary meanings for differences in AIC

There are no definite meanings for differences in AIC, but the following are somewhat customary interpretations.

- Models with $\Delta < 2$ are basically indistinguishable from the model with AIC_{min}.
- Models with $\Delta > 10$ are much less predictive or explanatory than the model with AIC_{min}.
- For $2 < \Delta < 10$ there are a variety of adjectives used to describe model quality.

Other information critera

Other options exist, they are also all modifications of the likelihood, and behave and are interpreted similarly to AIC.

```r
AIC(glm1)+2*k*(k+1)/(n-k-1)
```

```
## [1] 203.5738
```

```r
AICc(glm1) #from MuMIn. bbmle's doesn't work on glm
```

```
## [1] 203.5738
```

```r
-2*L+log(n)*k
```

```
## [1] 207.1425
```

```r
BIC(glm1)
```

```
## [1] 207.1425
```

Fisher scoring iterations

Fisher's method of iteratively adjusting the parameters to get an increase in maximum likelihood is how the job is done. The number of iterations is how long it took to converge.

Other applications of likelihood

- Likelihood ratio tests

These can be implemented using the `anova` command on a `glm` and specifying `test="LRT"`. See `?anova.glm`. There is also the command `drop1` if you want to do the equivalent of marginal instead of sequential ANOVA.

- Bayesian analysis

The likelihood is a key element in Bayesian analysis, which takes the idea that model parameters are the real variables, and the data are the values that we really know and runs with it.

Likelihood ratio tests

```
x1 <- runif(n, min=0, max=5); x2 <- runif(n, min=0, max=10)
y <- rpois(n, 2+2*x1+x2)
glm2interact <- glm(y~x1*x2, family = poisson)
glm2both <- glm(y~x1+x2, family = poisson)
glm2x1 <- glm(y~x1, family = poisson)
glm2x2 <- glm(y~x2, family=poisson)
glm2null <- glm(y~1, family = poisson)
anova(glm2null,glm2x1,glm2both, glm2interact, test="LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: y ~ 1
## Model 2: y ~ x1
## Model 3: y ~ x1 + x2
## Model 4: y ~ x1 * x2
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1        49    106.014
## 2        48     60.389  1   45.625 1.432e-11 ***
## 3        47     40.998  1   19.391 1.065e-05 ***
## 4        46     37.316  1    3.682   0.05501 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Other applications of mle2

This tool can do maximum likelihood estimation to find parameters for many sorts of models that `glm` can't. It's tricky to use, and can rely in unexpected ways on the `start` values, so often it's easier to find a prepackaged tool like `glm`.

But if you know exactly the model you want to build and have good guesses for the parameters that you hope to estimate, this could be the tool for you.

Things in `summary.lm` but not `summary.glm`

There are two things that we are missing.

- R^2

There's no consensus about what we can or should use to replace R^2. There's a good discussion of this by Bolker here: [https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html#how-do-i-compute-a-coefficient-of-determination-r2-or-an-analogue-for-glmms]

- An overall p-value.

There's no obvious replacement for F, or clear reason why we can't keep using F. But since there are likelihood ratios (and other metrics of goodness-of-fit) it would be presumptuous to give an overall p-value based on an F test.