MATH 559: Bayesian Theory and Methods

Bayesian inference for GLMs

In a generalized linear model (GLM) the conditional expectation of a response variable Y_i is modelled as a function of covariates $\mathbf{x}_i = (1, x_{i1}, \ldots, x_{id})$, specifically

$$\mathbb{E}[Y_i|\mathbf{x}_i;\beta] = g(\mathbf{x}_i\beta) = g(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_{id} x_{id}) = \mu_i$$

say for some inverse link function $g(.)$, and where the conditional distribution of Y_i is an Exponential-Dispersion family model. In such a model

$$\text{Var}[Y_i|\mathbf{x}_i;\beta] = \phi V(\mu_i)$$

for dispersion parameter ϕ, and variance function $V(.)$.

Poisson Regression: In a Poisson regression model,

$$Y_i|\mathbf{x}_i;\beta \sim Poisson(\mu_i)$$

and the usual approach is to choose $g(t) = e^t$, so that

$$\mu_i = \exp(\mathbf{x}_i\beta) \qquad V(\mu_i) = \mu_i \qquad \phi = 1.$$

Then the likelihood function is

$$\mathcal{L}_n(\beta) = \prod_{i=1}^n f_{Y|X}(y_i|\mathbf{x}_i;\beta) = \prod_{i=1}^n \frac{e^{-\mu_i}}{y_i!}\mu_i^{y_i}$$

so that the log likelihood is

$$\ell_n(\beta) = \sum_{i=1}^n \left(-\mu_i + y_i \log \mu_i - \log y_i!\right) = \sum_{i=1}^n \left(-\exp\{\mathbf{x}_i\beta\} + y_i\mathbf{x}_i\beta - \log y_i!\right).$$

To find the maximum likelihood estimate (mle), we differentiate with respect to β and equate the result to zero: we have

$$\dot{\ell}_n(\beta) = \sum_{i=1}^n \left(-\exp\{\mathbf{x}_i\beta\}\mathbf{x}_i^\top + y_i\mathbf{x}_i^\top\right) = \sum_{i=1}^n \left(y_i - \exp\{\mathbf{x}_i\beta\}\right)\mathbf{x}_i^\top = \sum_{i=1}^n \left(y_i - \mu_i\right)\mathbf{x}_i^\top$$

and equating to the $(d+1)$-dimensional zero vector yields the mle $\widehat{\beta}_n$. The second derivative $\ddot{\ell}_n(\beta)$ evaluated at $\widehat{\beta}_n$ yields the curvature of the log-likelihood at the mle:

$$\ddot{\ell}_n(\beta) = -\sum_{i=1}^n \exp\{\mathbf{x}_i\beta\}\mathbf{x}_i^\top\mathbf{x}_i = -\sum_{i=1}^n \mu_i\mathbf{x}_i^\top\mathbf{x}_i = -\mathbf{X}^\top\mathbf{D}_n\mathbf{X}$$

say, where

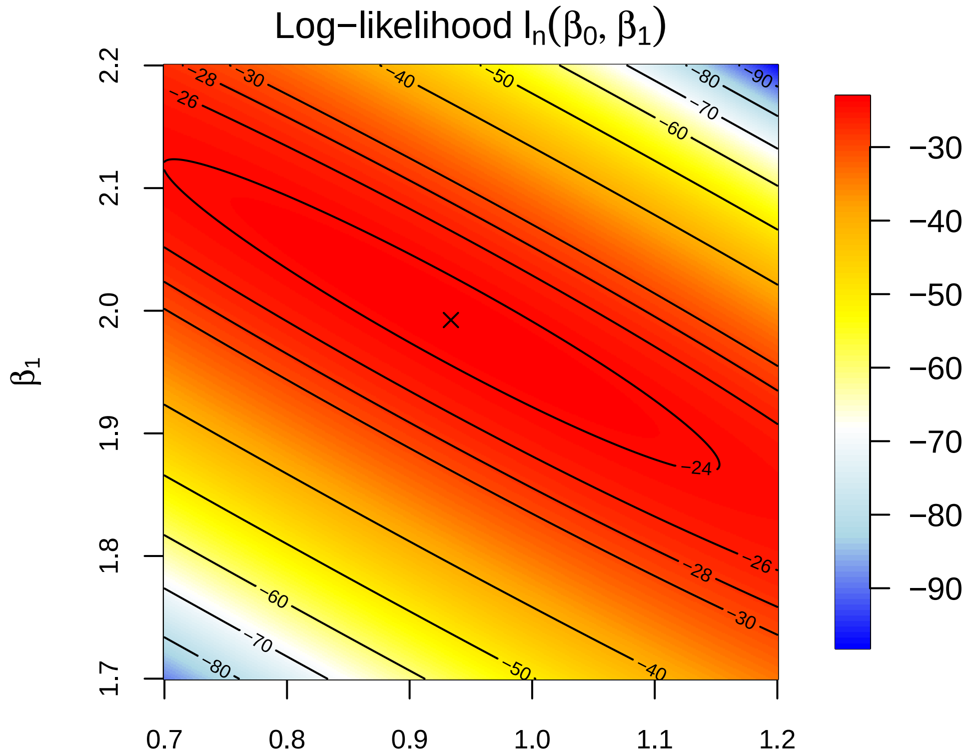
$$\mathbf{X} = \begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \vdots\\ \mathbf{x}_n\end{pmatrix} \qquad \mathbf{D}_n = \mathbf{D}_n(\beta) = \text{diag}(\mu_1, \ldots, \mu_n).$$

Using the quadratic approximation to the log-likelihood, we see that for large n the posterior distribution is approximately Normal

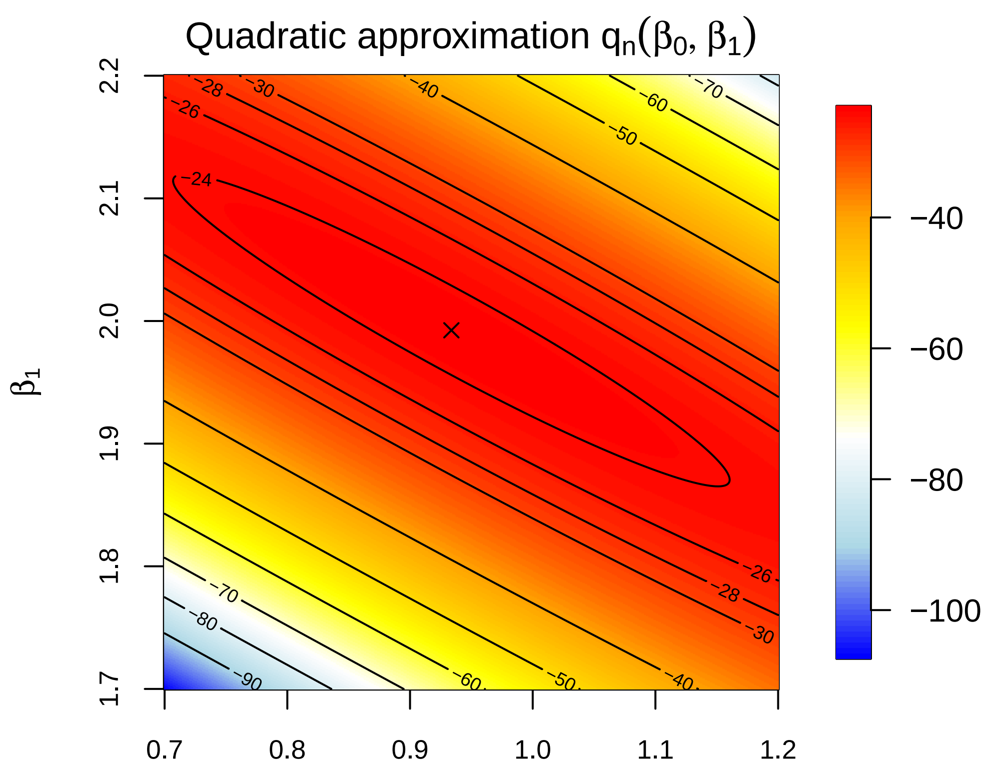
$$\pi_n(\beta) \approx Normal\left(\widehat{\beta}_n, (\mathbf{X}^\top\widehat{\mathbf{D}}_n\mathbf{X})^{-1}\right)$$

and $\widehat{\mathbf{D}}_n = \mathbf{D}_n(\widehat{\beta}_n)$. The mle and variance matrices can be computed using `glm` in `R`.

```r
set.seed(234)
n<-20
x<-rnorm(n,-1,2)
X<-cbind(1,x)
be<-c(1,2)
mu<-exp(X %*% be)
y<-rpois(n,mu)
fit1<-glm(y~x,family=poisson)
th.n<-coef(fit1)
post.var.n<-summary(fit1)$cov.unscaled
th.n

+ (Intercept)           x
+   0.9337213   1.9924322

post.var.n

+             (Intercept)           x
+ (Intercept)  0.04070200 -0.02188226
+ x           -0.02188226  0.01281759

#Exact log likelhood
log.like<-function(th0,th1,xv,yv){
    muv<-exp(th0+th1*xv)
    return(sum(dpois(yv,muv,log=T)))
}
f <- Vectorize(log.like,vectorize.args=c("th0","th1"))
th0v<-seq(0.7,1.2,by=0.001)
th1v<-seq(1.7,2.2,by=0.001)
lmat<-outer(th0v,th1v,f,xv=x,yv=y)
lmax<-max(lmat)

#Quadratic approximation
quad.func<-function(th0,th1,m,M){

    tv<-c(th0,th1)-m
    q<--0.5*t(tv) %*% solve(M) %*% tv
    return(q[1])
}
f <- Vectorize(quad.func,vectorize.args=c("th0","th1"))
th0v<-seq(0.7,1.2,by=0.001)
th1v<-seq(1.7,2.2,by=0.001)
qmat<-outer(th0v,th1v,f,m=th.n,M=post.var.n)+lmax


#Plot
library(fields,quietly=TRUE)
par(pty='s',mar=c(2,3,2,2))
colfunc <- colorRampPalette(c("blue","lightblue","white","yellow","orange","red"))
image.plot(th0v,th1v,lmat,col=colfunc(100),
           xlab=expression(beta[0]),ylab=expression(beta[1]),cex.axis=0.8)
contour(th0v,th1v,lmat,add=T,levels=c(seq(-90,-40,by=10),seq(-30,-20,by=2)))
title(expression(paste('Log-likelihood ',l[n](beta[0],beta[1]))))
points(th.n[1],th.n[2],pch=4)
```




```
image.plot(th0v,th1v,qmat,col=colfunc(100),
           xlab=expression(beta[0]),ylab=expression(beta[1]),cex.axis=0.8)
contour(th0v,th1v,qmat,add=T,levels=c(seq(-90,-40,by=10),seq(-30,-20,by=2)))
title(expression(paste('Quadratic approximation ',q[n](beta[0],beta[1]))))
points(th.n[1],th.n[2],pch=4)
```



Logistic Regression: In a logistic regression model,

$$Y_i|\mathbf{x}_i;\beta \sim Bernoulli(\mu_i)$$

and the usual approach is to choose $g(t) = e^t/(1+e^t) = \text{expit}\{t\}$, so that

$$\mu_i = \text{expit}(\mathbf{x}_i\beta) \qquad V(\mu_i) = \mu_i(1-\mu_i) \qquad \phi = 1.$$

Then the likelihood function is

$$\mathcal{L}_n(\beta) = \prod_{i=1}^{n} f_{Y|X}(y_i|\mathbf{x}_i;\beta) = \prod_{i=1}^{n} \mu_i^{y_i}(1-\mu_i)^{1-\mu_i}$$

so that the log likelihood is

$$\ell_n(\beta) = \sum_{i=1}^{n} \left(y_i \log \mu_i + (1-y_i)\log(1-\mu_i)\right) = \sum_{i=1}^{n} \left(y_i\mathbf{x}_i\beta - \log(1+\exp(\mathbf{x}_i\beta))\right).$$

To find the maximum likelihood estimate (mle), we differentiate with respect to β and equate the result to zero: we have

$$\dot{\ell}_n(\beta) = \sum_{i=1}^{n} \left(y_i\mathbf{x}_i^\top - \text{expit}(\mathbf{x}_i\beta)\mathbf{x}_i^\top\right) = \sum_{i=1}^{n} (y_i - \mu_i)\,\mathbf{x}_i^\top$$

and equating to the $(d+1)$-dimensional zero vector yields the mle $\widehat{\beta}_n$. The second derivative $\ddot{\ell}_n(\beta)$ evaluated at $\widehat{\beta}_n$ yields the curvature of the log-likelihood at the mle:

$$\ddot{\ell}_n(\beta) = -\sum_{i=1}^{n} \mu_i(1-\mu_i)\mathbf{x}_i^\top\mathbf{x}_i = -\mathbf{X}^\top\mathbf{D}_n\mathbf{X}$$

say, where

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} \qquad \mathbf{D}_n = \text{diag}(\mu_1(1-\mu_1),\ldots,\mu_n(1-\mu_n)).$$

Using the quadratic approximation to the log-likelihood, we see that for large n the posterior distribution is approximately Normal

$$\pi_n(\beta) \approx Normal\left(\widehat{\beta}_n, (\mathbf{X}^\top\widehat{\mathbf{D}}_n\mathbf{X})^{-1}\right)$$

The mle and variance matrices can be computed using `glm` in `R`.

In the following plots, the exact log-likelihood, and the corresponding quadratic approximation, are plotted for

(i) $n = 20$

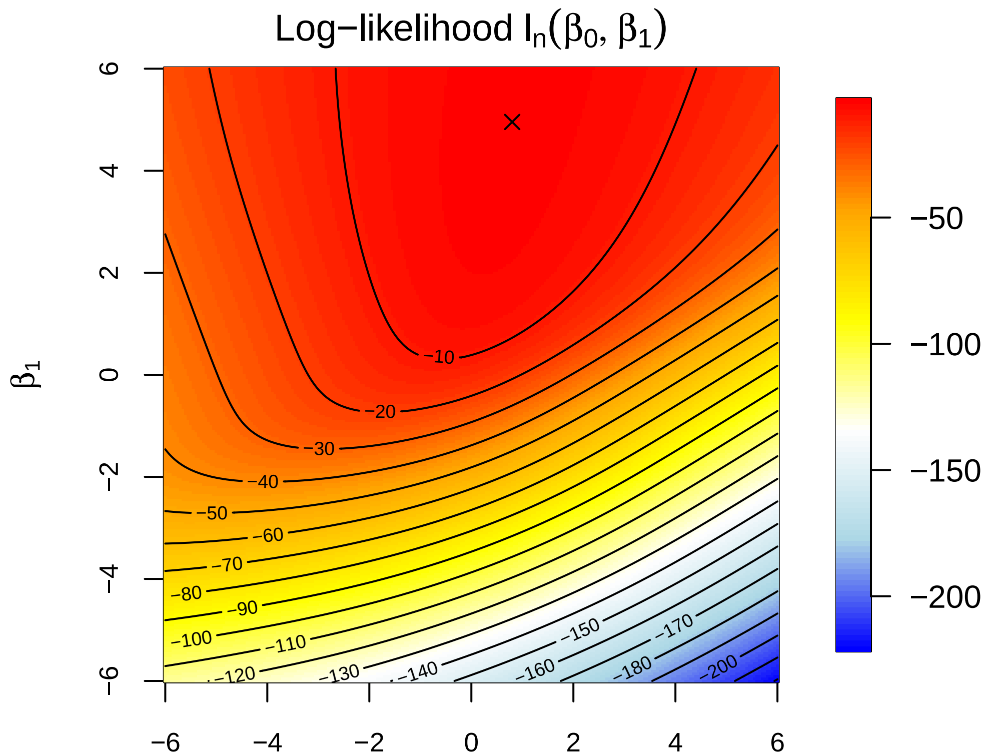
(ii) $n = 200$

(iii) $n = 2000$

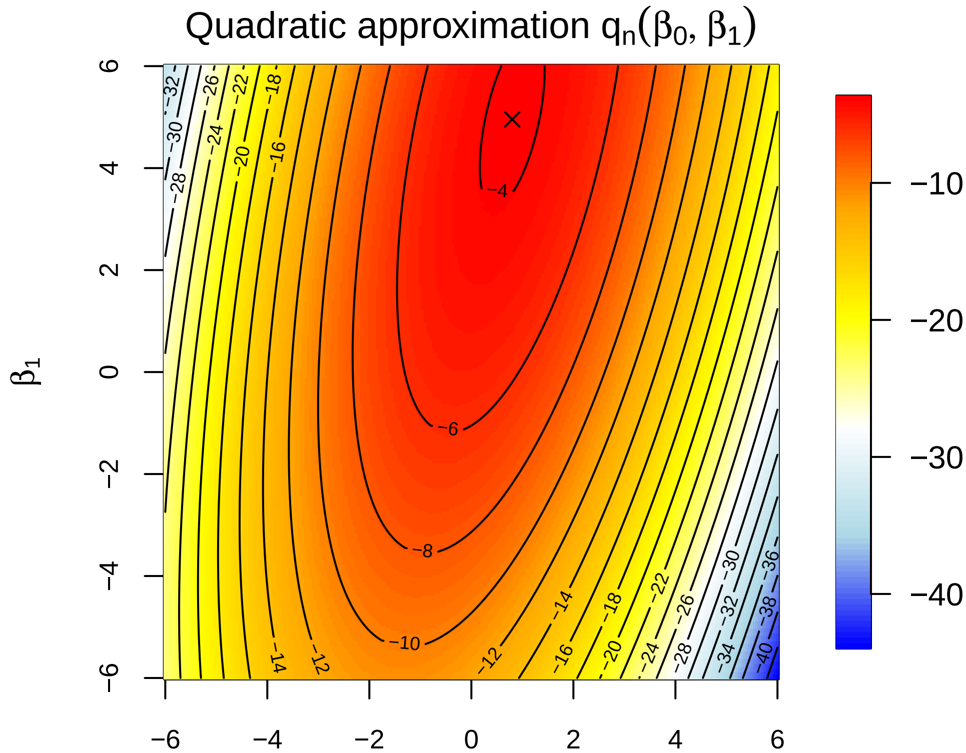
Even at the largest sample size, the quadratic approximation is good only quite close to the maximum value.

```r
set.seed(2634)
n<-20
x<-rnorm(n,-1,1)
X<-cbind(1,x)
be<-c(1,2)
expit<-function(x){return(1/(1+exp(-x)))}
mu<-expit(X %*% be)
y<-rbinom(n,1,mu)
table(y)

+ y
+  0  1
+ 14  6

fit1<-glm(y~x,family=binomial)
th.n<-coef(fit1)
post.var.n<-summary(fit1)$cov.unscaled
th.n

+ (Intercept)           x
+   0.8008863   4.9555134

post.var.n

+             (Intercept)        x
+ (Intercept)    1.173084 1.736133
+ x              1.736133 8.685884

#Exact log likelhood
log.like<-function(th0,th1,xv,yv){
    muv<-expit(th0+th1*xv)
    return(sum(dbinom(yv,1,muv,log=T)))
}
f <- Vectorize(log.like,vectorize.args=c("th0","th1"))
th0v<-seq(-6,6,by=0.05)
th1v<-seq(-6,6,by=0.05)
lmat<-outer(th0v,th1v,f,xv=x,yv=y)
lmax<-max(lmat)

#Quadratic approximation
quad.func<-function(th0,th1,m,M){

    tv<-c(th0,th1)-m
    q<--0.5*t(tv) %*% solve(M) %*% tv
    return(q[1])
}
f <- Vectorize(quad.func,vectorize.args=c("th0","th1"))
qmat<-outer(th0v,th1v,f,m=th.n,M=post.var.n)+lmax


#Plot
library(fields,quietly=TRUE)
par(pty='s',mar=c(2,3,2,2))
colfunc <- colorRampPalette(c("blue","lightblue","white","yellow","orange","red"))
image.plot(th0v,th1v,lmat,col=colfunc(100),
```

```
         xlab=expression(beta[0]),ylab=expression(beta[1]),cex.axis=0.8)
contour(th0v,th1v,lmat,add=T,nlevels=20)
title(expression(paste('Log-likelihood ',l[n](beta[0],beta[1]))))
points(th.n[1],th.n[2],pch=4)
```

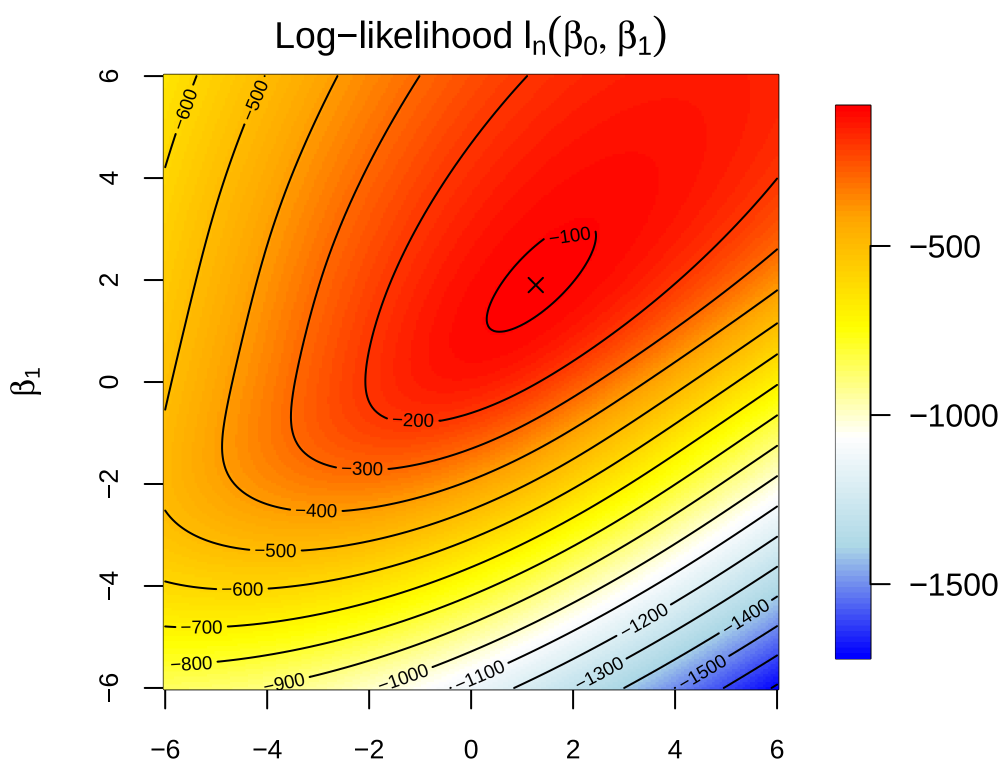



```
image.plot(th0v,th1v,qmat,col=colfunc(100),
         xlab=expression(beta[0]),ylab=expression(beta[1]),cex.axis=0.8)
contour(th0v,th1v,qmat,add=T,nlevels=20)
title(expression(paste('Quadratic approximation ',q[n](beta[0],beta[1]))))
points(th.n[1],th.n[2],pch=4)
```

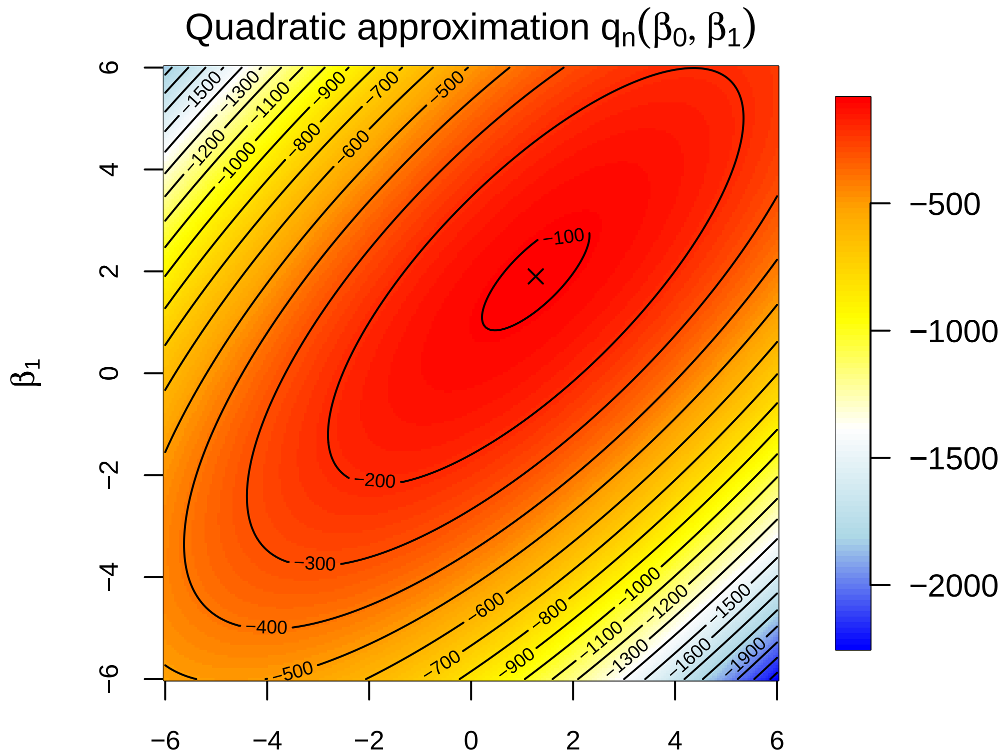



```
set.seed(2634)
n<-200
x<-rnorm(n,-1,1)
X<-cbind(1,x)
be<-c(1,2)
expit<-function(x){return(1/(1+exp(-x)))}
mu<-expit(X %*% be)
y<-rbinom(n,1,mu)
table(y)

+ y
+   0   1
+ 115  85

fit1<-glm(y~x,family=binomial)
th.n<-coef(fit1)
post.var.n<-summary(fit1)$cov.unscaled
th.n

+ (Intercept)           x
+    1.267769    1.901749

post.var.n

+             (Intercept)          x
+ (Intercept)  0.07745220 0.05903546
+ x            0.05903546 0.07801711

#Exact log likelihood
log.like<-function(th0,th1,xv,yv){
```

```
    muv<-expit(th0+th1*xv)
    return(sum(dbinom(yv,1,muv,log=T)))
}
f <- Vectorize(log.like,vectorize.args=c("th0","th1"))
th0v<-seq(-6,6,by=0.05)
th1v<-seq(-6,6,by=0.05)
lmat<-outer(th0v,th1v,f,xv=x,yv=y)
lmax<-max(lmat)

#Quadratic approximation
quad.func<-function(th0,th1,m,M){

    tv<-c(th0,th1)-m
    q<--0.5*t(tv) %*% solve(M) %*% tv
    return(q[1])
}
f <- Vectorize(quad.func,vectorize.args=c("th0","th1"))
qmat<-outer(th0v,th1v,f,m=th.n,M=post.var.n)+lmax


#Plot
library(fields,quietly=TRUE)
par(pty='s',mar=c(2,3,2,2))
colfunc <- colorRampPalette(c("blue","lightblue","white","yellow","orange","red"))
image.plot(th0v,th1v,lmat,col=colfunc(100),
           xlab=expression(beta[0]),ylab=expression(beta[1]),cex.axis=0.8)
contour(th0v,th1v,lmat,add=T,nlevels=20)
title(expression(paste('Log-likelihood ',l[n](beta[0],beta[1]))))
points(th.n[1],th.n[2],pch=4)
```

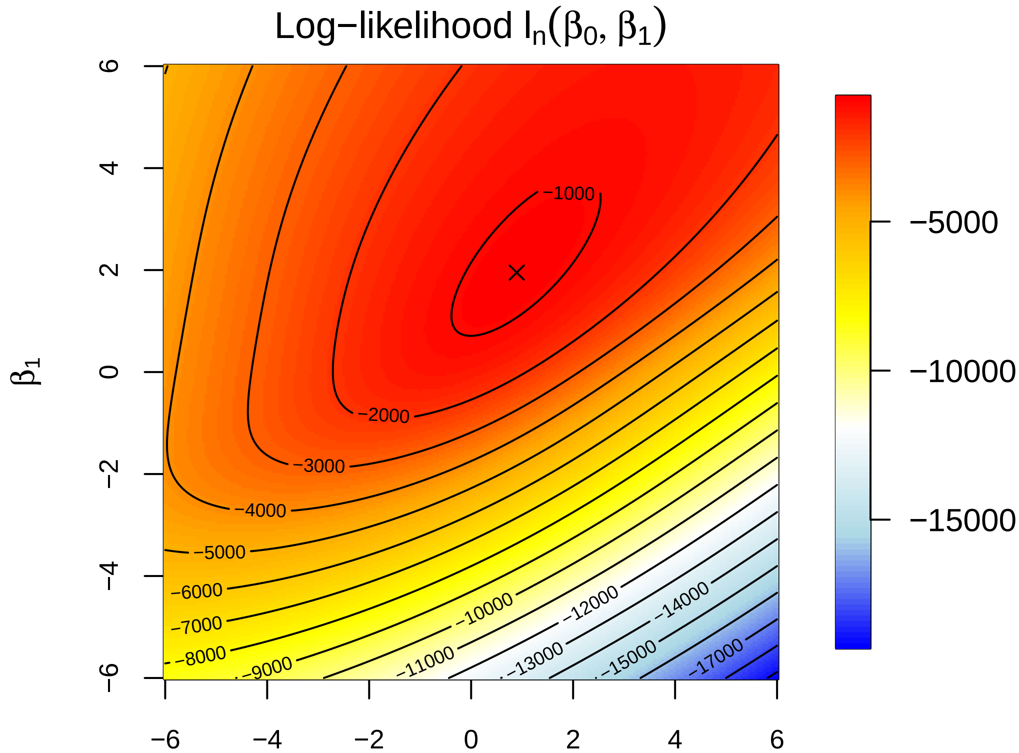


```
image.plot(th0v,th1v,qmat,col=colfunc(100),
           xlab=expression(beta[0]),ylab=expression(beta[1]),cex.axis=0.8)
contour(th0v,th1v,qmat,add=T,nlevels=20)
title(expression(paste('Quadratic approximation ',q[n](beta[0],beta[1]))))
points(th.n[1],th.n[2],pch=4)
```




```
set.seed(2634)
n<-2000
x<-rnorm(n,-1,1)
X<-cbind(1,x)
be<-c(1,2)
expit<-function(x){return(1/(1+exp(-x)))}
mu<-expit(X %*% be)
y<-rbinom(n,1,mu)
table(y)

+ y
+    0    1
+ 1310  690

fit1<-glm(y~x,family=binomial)
th.n<-coef(fit1)
post.var.n<-summary(fit1)$cov.unscaled
th.n

+ (Intercept)           x
+   0.8939345   1.9505547

post.var.n
```

```
+              (Intercept)           x
+ (Intercept) 0.007516227 0.005855666
+ x           0.005855666 0.008640838

#Exact log likelhood
log.like<-function(th0,th1,xv,yv){
    muv<-expit(th0+th1*xv)
    return(sum(dbinom(yv,1,muv,log=T)))
}
f <- Vectorize(log.like,vectorize.args=c("th0","th1"))
th0v<-seq(-6,6,by=0.05)
th1v<-seq(-6,6,by=0.05)
lmat<-outer(th0v,th1v,f,xv=x,yv=y)
lmax<-max(lmat)

#Quadratic approximation
quad.func<-function(th0,th1,m,M){

    tv<-c(th0,th1)-m
    q<--0.5*t(tv) %*% solve(M) %*% tv
    return(q[1])
}
f <- Vectorize(quad.func,vectorize.args=c("th0","th1"))
qmat<-outer(th0v,th1v,f,m=th.n,M=post.var.n)+lmax


#Plot
library(fields,quietly=TRUE)
par(pty='s',mar=c(2,3,2,2))
colfunc <- colorRampPalette(c("blue","lightblue","white","yellow","orange","red"))
image.plot(th0v,th1v,lmat,col=colfunc(100),
           xlab=expression(beta[0]),ylab=expression(beta[1]),cex.axis=0.8)
contour(th0v,th1v,lmat,add=T,nlevels=20)
title(expression(paste('Log-likelihood ',l[n](beta[0],beta[1]))))
points(th.n[1],th.n[2],pch=4)
```




```
image.plot(th0v,th1v,qmat,col=colfunc(100),
           xlab=expression(beta[0]),ylab=expression(beta[1]),cex.axis=0.8)
contour(th0v,th1v,qmat,add=T,nlevels=20)
title(expression(paste('Quadratic approximation ',q[n](beta[0],beta[1]))))
points(th.n[1],th.n[2],pch=4)
```