SDS385 Fall '16: Statistical Models For Big Data
Exercises 01 - Generalized linear models

Matteo Vestrucci

September 1st 2016

A)

The negative log-likelihood can be rewritten like this:

$$
\begin{aligned}
l(\beta) &= -\log\left\{\prod_{i=1}^{N} p(y_i|\beta)\right\} \\
&= -\log\left[\prod_{i=1}^{N}\binom{m_i}{y_i} w_i^{y_i}(1-w_i)^{m_i-y_i}\right] \\
&= c - \sum_{i=1}^{N} y_i \log(w_i) - \sum_{i=1}^{N} m_i \log(1-w_i) + \sum_{i=1}^{N} y_i \log(1-w_i) \\
&= c - \sum_{i=1}^{N} y_i \log\left(\frac{w_i}{1-w_i}\right) - \sum_{i=1}^{N} m_i \log(1-w_i) \\
&= c - \sum_{i=1}^{N} y_i \log\left(\frac{1}{e^{-x_i^T\beta}}\right) - \sum_{i=1}^{N} m_i \log\left(\frac{e^{-x_i^T\beta}}{1+e^{-x_i^T\beta}}\right) \\
&= c - \sum_{i=1}^{N} y_i x_i^T\beta + \sum_{i=1}^{N} m_i x_i^T\beta + \sum_{i=1}^{N} m_i \log(1+e^{-x_i^T\beta}) \\
&\approx \sum_{i=1}^{N} (m_i - y_i) x_i^T\beta + \sum_{i=1}^{N} m_i \log(1+e^{-x_i^T\beta})
\end{aligned}
$$

Its gradient is:

$$\begin{aligned}
\nabla l(\beta) &= \nabla \left(\sum_{i=1}^{N} (m_i - y_i) x_i^T \beta + \sum_{i=1}^{N} m_i \log(1 + e^{-x_i^T \beta}) \right) \\
&= \sum_{i=1}^{N} (m_i - y_i) x_i + \sum_{i=1}^{N} m_i w_i e^{-x_i^T \beta} (-x_i) \\
&= \sum_{i=1}^{N} m_i x_i - \sum_{i=1}^{N} y_i x_i - \sum_{i=1}^{N} m_i (1 - w_i) x_i \\
&= - \sum_{i=1}^{N} y_i x_i + \sum_{i=1}^{N} m_i w_i x_i \\
&= \sum_{i=1}^{N} (m_i w_i - y_i) x_i = \sum_{i=1}^{N} (\hat{y} - y_i) x_i = X^T S
\end{aligned}$$

where S is a vector with i-th element $m_i w_i - y_i$.

B)

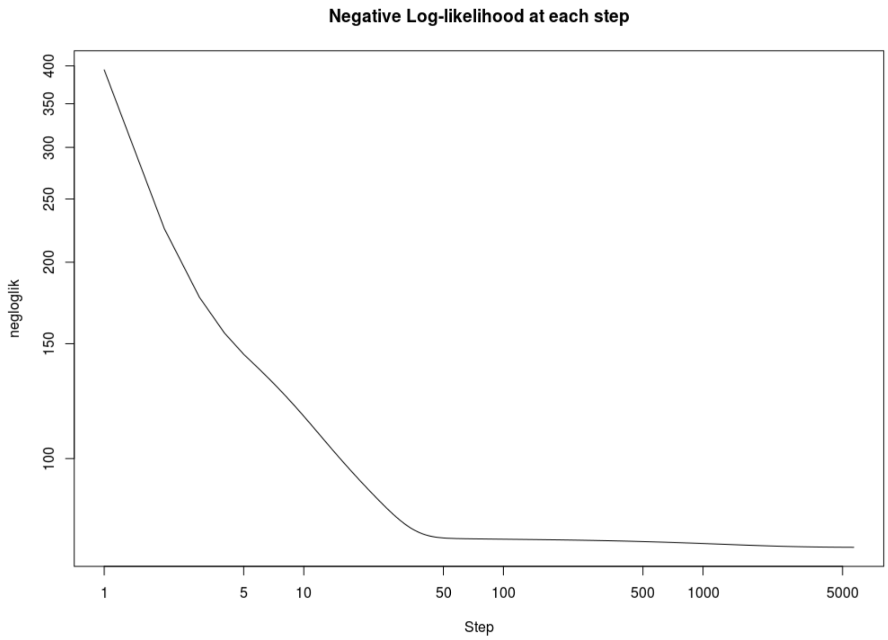
The gradient descent updates at each step the values of our estimated β using this formula:

$$\beta_{t+1} = \beta_t - a \nabla l(\beta_t)$$

with a being the step size. Running the code in appendix, we can obtain the following $\hat{\beta}$:

c	0.352797332
V3	-4.036418435
V4	1.650240487
V5	-3.813750020
V6	12.587370606
V7	1.051996544
V8	-0.005401012
V9	0.729982621
V10	2.582549729
V11	0.444998302
V12	-0.463949952

The following graph of the objective function shows us that indeed it's being decreased at each step.



C)

The Hessian is:

$$\begin{aligned}
\nabla^2 l(\beta) &= \nabla \left(-\sum_{i=1}^{N} y_i x_i + \sum_{i=1}^{N} m_i w_i x_i \right) \\
&= \nabla \left(c + \sum_{i=1}^{N} m_i x_{i1} \frac{1}{1+e^{-x_i^T\beta}} \ , \ \dots \ , \ c + \sum_{i=1}^{N} m_i x_{iP} \frac{1}{1+e^{-x_i^T\beta}} \right) \\
&= \begin{bmatrix} \sum_{i=1}^{N} m_i x_{i1} \frac{-e^{-x_i^T\beta}}{\left(1+e^{-x_i^T\beta}\right)^2}(-x_{i1}) & \cdots & \sum_{i=1}^{N} m_i x_{i1} \frac{-e^{-x_i^T\beta}}{\left(1+e^{-x_i^T\beta}\right)^2}(-x_{iP}) \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{N} m_i x_{iP} \frac{-e^{-x_i^T\beta}}{\left(1+e^{-x_i^T\beta}\right)^2}(-x_{i1}) & \cdots & \sum_{i=1}^{N} m_i x_{iP} \frac{-e^{-x_i^T\beta}}{\left(1+e^{-x_i^T\beta}\right)^2}(-x_{iP}) \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^{N} m_i x_{i1} x_{i1} w_i(1-w_i) & \cdots & \sum_{i=1}^{N} m_i x_{i1} x_{iP} w_i(1-w_i) \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{N} m_i x_{iP} x_{i1} w_i(1-w_i) & \cdots & \sum_{i=1}^{N} m_i x_{iP} x_{iP} w_i(1-w_i) \end{bmatrix} = X^T D X
\end{aligned}$$

where D is a diagonal matrix with ii-th element $m_i w_i(1-w_i)$. Notice that all the values on the diagonal are positive: this means that the Hessian is defined positive and that our objective function is convex. Then, considering that we can ignore the constant values adding and subtracting them as we need, we can obtain the requested result like this:

$$
\begin{aligned}
q_l(\beta, \beta_0) &= l(\beta_0) + \nabla(\beta_0)^T(\beta - \beta_0) + \frac{1}{2}(\beta - \beta_0)^t \nabla^2(\beta_0)(\beta - \beta_0) \\
&= c + S^T X(\beta - \beta_0) + \frac{1}{2}(\beta - \beta_0)^T X^T D X(\beta - \beta_0) \\
&= c + S^T X\beta + \frac{1}{2}\beta^T X^T D X\beta - \frac{1}{2}\beta^T X^T D X\beta_0 - \frac{1}{2}\beta_0^T X^T D X\beta_0 \\
&= c - (\beta_0^T X^T D - S^T)X\beta + \frac{1}{2}\beta^T X^T D X\beta \\
&= c - (\beta_0^T X^T D - S^T)D^{-1} D X\beta + \frac{1}{2}\beta^T X^T D X\beta \\
&\quad + \frac{1}{2}(\beta_0^T X^T D - S^T)D^{-1} D D^{-1}(\beta_0^T X^T D - S^T)^T \\
&= c + \frac{1}{2}[D^{-1}(\beta_0^T X^T D - S^T)^T - X\beta]^T D[(D^{-1}(\beta_0^T X^T D - S^T)^T - X\beta] \\
&= c + \frac{1}{2}[(X\beta_0 - D^{-1}S) - X\beta]^T D[(X\beta_0 - D^{-1}S) - X\beta] \\
&= c + \frac{1}{2}[Z - X\beta]^T W[Z - X\beta]
\end{aligned}
$$

where Z is the vector $X\beta_0 - D^{-1}S$ and where W is the matrix D.

D)

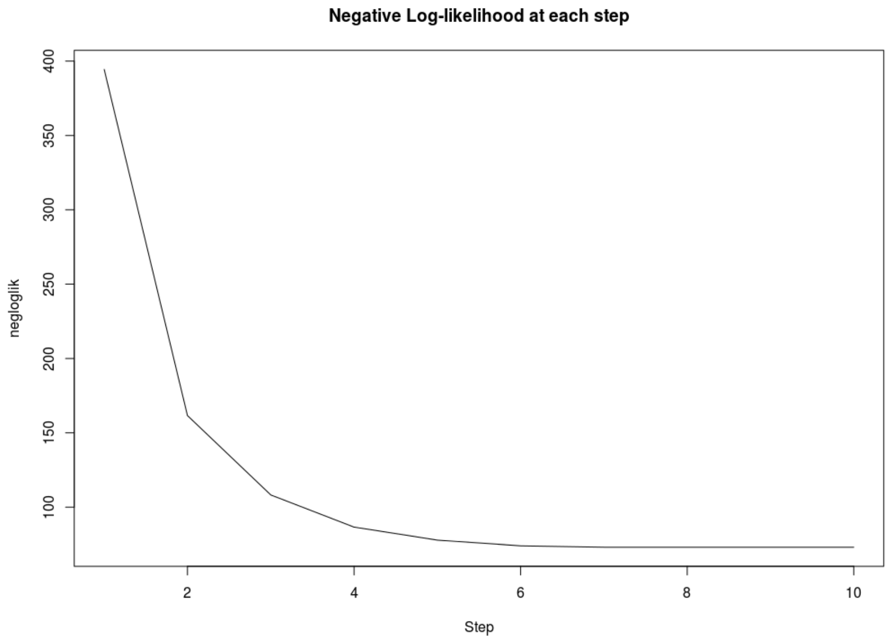
The newton's method updates at each step the values of our estimated β using this formula:

$$\beta_{t+1} = \beta_t - [\nabla^2 l(\beta_t)]^{-1} \nabla l(\beta_t)$$

Running the code in appendix, we can obtain the following $\hat{\beta}$:

c	0.48701675
V3	-7.22185053
V4	1.65475615
V5	-1.73763027
V6	14.00484560
V7	1.07495329
V8	-0.07723455
V9	0.67512313
V10	2.59287426
V11	0.44625631
V12	-0.48248420

The following graph of the objective function shows us that indeed it's being decreased at each step.



E)

Newton's method reached convergence in eleven steps, instead the gradient descent required more than 5000 steps using the same requirements accuracy-wise. This happened because Newton's method is known to fare better if the objective function can be well approximated using a quadratic function, and indeed in point C) we proved that it's true for our case. Specifically Newton's method exploits the additional information that comes from the curvature of the function to take steps in a direction that is not the steepest but that it's more convenient looking at the whole optimization process. Nonetheless in other cases it could be more advantageous to use the gradient descent because each step of Newton's method requires the inversion of the Hessian matrix, and that is a very expensive operation.

CODE)

```
library(Matrix)

negloglikelihood<-function(m,y,X,beta){
  total<-0
  N<-length(y)
  for(i in 1:N) total<-total+(m[i]-y[i])*t(X[i,])%*%beta+m[i]*log(1+exp(-t(X[i,])%*%beta))
  return(total)
}

gradient_negloglik<-function(m,y,X,beta){
  w<-1/(1+exp(-X%*%beta))
  S<-m*w-y
  grad<-t(X)%*%S
  return(grad)
}

gradient_descent<-function(m,y,X,beta0,stepsize,maxstepnumber,
                           accuracy_obj_fun,accuracy_beta_val){
  negloglik<-numeric(maxstepnumber)
  negloglik[1]<-negloglikelihood(m,y,X,beta0)
  gradient<-gradient_negloglik(m,y,X,beta0)
  diff_beta_val<-accuracy_beta_val+1
  diff_obj_fun<-accuracy_obj_fun+1
  i<-1
  while(!(i==maxstepnumber)&&
        ((accuracy_beta_val<diff_beta_val)||
        (accuracy_obj_fun<diff_obj_fun))){
    i<-i+1
    beta1<-beta0-stepsize*gradient
    diff_beta_val<-sum(abs(beta0-beta1))
    beta0<-beta1
    negloglik[i]<-negloglikelihood(m,y,X,beta0)
    diff_obj_fun<-negloglik[i-1]-negloglik[i]
    gradient<-gradient_negloglik(m,y,X,beta0)
  }
  return(list(betahat=beta0,negloglik=negloglik,step=i))
}

data_wdbc<-read.csv("./wdbc.csv", header=FALSE)
X<-as.matrix(cbind(rep(1,569),data_wdbc[,3:12]))
y<-data_wdbc[,2]
y<-as.numeric(y=="M")
m<-rep(1,569)

beta0<-rep(0,11)
stepsize<-2/10^8
maxstepnumber<-10000
```

```
accuracy_obj_fun<-0.001
accuracy_beta_val<-0.00001
result_grad_desc<-gradient_descent(m,y,X,beta0,stepsize,maxstepnumber,
                                   accuracy_obj_fun,accuracy_beta_val)

result_grad_desc$betahat
plot(result_grad_desc$negloglik[1:result_grad_desc$step],
     main = "Negative Log-likelihood at each step",
     xlab="Step",ylab="negloglik",type="l")

hessian_negloglik<-function(m,y,X,beta){
  w<-as.vector(1/(1+exp(-X%*%beta)))
  D<-diag(m*w*(1-w))
  hes<-t(X)%*%D%*%X
  return(hes)
}

newton_descent<-function(m,y,X,beta0,maxstepnumber,
                         accuracy_obj_fun,accuracy_beta_val){
  negloglik<-numeric(maxstepnumber)
  negloglik[1]<-negloglikelihood(m,y,X,beta0)
  gradient<-gradient_negloglik(m,y,X,beta0)
  hessian<-hessian_negloglik(m,y,X,beta0)
  diff_beta_val<-accuracy_beta_val+1
  diff_obj_fun<-accuracy_obj_fun+1
  i<-1
  while(!(i==maxstepnumber)&&
        ((accuracy_beta_val<diff_beta_val)||
        (accuracy_obj_fun<diff_obj_fun))){
    i<-i+1
    beta1<-beta0-solve(hessian,gradient)
    diff_beta_val<-sum(abs(beta0-beta1))
    beta0<-beta1
    negloglik[i]<-negloglikelihood(m,y,X,beta0)
    diff_obj_fun<-negloglik[i-1]-negloglik[i]
    gradient<-gradient_negloglik(m,y,X,beta0)
    hessian<-hessian_negloglik(m,y,X,beta0)
  }
  return(list(betahat=beta0,negloglik=negloglik,step=i))
}

result_newton_desc<-newton_descent(m,y,X,beta0,maxstepnumber,
                                   accuracy_obj_fun,accuracy_beta_val)

result_newton_desc$betahat
plot(result_newton_desc$negloglik[1:result_newton_desc$step],
     main = "Negative Log-likelihood at each step",
     xlab="Step",ylab="negloglik",type="l")
```