Untitled

Amy Pitts

3/17/2022

Data import

This dataset has 569 rows and 32 columns. The outcome variable of interst is `Diagnosis` which takes on values benign or malignant cases. There are 357 benign cases and 212 malignant cases as seen below in Table 1. One variable is `id` and the rest 30 variables are the mean, sd and largest values of the following criteria.

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

Using this dataset models will be compared and compared on their ability to predict cancer diagnosis.

Characteristic	**Benign**, N = 357	**Malignant**, N = 212	**p-value**
radius_mean	12.147 (1.781)	17.463 (3.204)	<0.001
texture_mean	17.915 (3.995)	21.605 (3.779)	<0.001
perimeter_mean	78.075 (11.807)	115.365 (21.855)	<0.001
area_mean	462.790 (134.287)	978.376 (367.938)	<0.001
smoothness_mean	0.092 (0.013)	0.103 (0.013)	<0.001
compactness_mean	0.080 (0.034)	0.145 (0.054)	<0.001
concavity_mean	0.046 (0.043)	0.161 (0.075)	<0.001
concave.points_mean	0.026 (0.016)	0.088 (0.034)	<0.001
symmetry_mean	0.174 (0.025)	0.193 (0.028)	<0.001
fractal_dimension_mean	0.063 (0.007)	0.063 (0.008)	0.5
radius_se	0.284 (0.113)	0.609 (0.345)	<0.001
texture_se	1.220 (0.589)	1.211 (0.483)	0.6
perimeter_se	2.000 (0.771)	4.324 (2.569)	<0.001
area_se	21.135 (8.843)	72.672 (61.355)	<0.001
smoothness_se	0.007 (0.003)	0.007 (0.003)	0.2
compactness_se	0.021 (0.016)	0.032 (0.018)	<0.001
concavity_se	0.026 (0.033)	0.042 (0.022)	<0.001
concave.points_se	0.010 (0.006)	0.015 (0.006)	<0.001
symmetry_se	0.021 (0.007)	0.020 (0.010)	0.028
fractal_dimension_se	0.004 (0.003)	0.004 (0.002)	<0.001
radius_worst	13.380 (1.981)	21.135 (4.284)	<0.001
texture_worst	23.515 (5.494)	29.318 (5.435)	<0.001
perimeter_worst	87.006 (13.527)	141.370 (29.457)	<0.001

Characteristic	**Benign**, N = 357	**Malignant**, N = 212	**p-value**
area_worst	558.899 (163.601)	1,422.286 (597.968)	<0.001
smoothness_worst	0.125 (0.020)	0.145 (0.022)	<0.001
compactness_worst	0.183 (0.092)	0.375 (0.170)	<0.001
concavity_worst	0.166 (0.140)	0.451 (0.182)	<0.001
concave.points_worst	0.074 (0.036)	0.182 (0.046)	<0.001
symmetry_worst	0.270 (0.042)	0.323 (0.075)	<0.001
fractal_dimension_worst	0.079 (0.014)	0.092 (0.022)	<0.001

Build a logistic model to classify the images into malignant/benign

Before building the model we need to first write down the likelihood function, its gradient and Hessian matrix.

The likelihood function for our data which has a single binary response and 30 numerical explanatory variables is

$$\pi_i = P(Y_i = 1|x_{i,1}, \dots x_{i,30}) = \frac{e^{\beta_0+\beta_1 x_{i,1}+\dots\beta_{30}x_{i,30}}}{1+e^{\beta_0+\beta_1 x_{i,1}+\dots\beta_{30}x_{i,30}}} = \frac{e^{\beta_0+\sum_{j=1}^{30}\beta_i x_{i,j}}}{1+e^{\beta_0+\sum_{j=1}^{30}\beta_i x_{i,j}}}$$

Where $\mathbf{X_i}$ represents the i observation of all 30 of our predictor variables. For the data give we have the likelihood is given by

$$L(\mathbf{X}|\beta) = \prod_{i=1}^{n}\left[\pi_i^{y_i}(1-\pi_i)^{1-y_i}\right]$$

Finding the log-likelihood we have

$$l(\mathbf{X}|\vec{\beta}) = \sum_{i=1}^{n}\left[y_i\left(\beta_0+\sum_{j=1}^{30}\beta_i x_{i,j}\right) - \log\left(1+\exp\left(\beta_0+\sum_{j=1}^{30}\beta_i x_{i,j}\right)\right)\right]$$

The gradient can then can be solved for. Observe

$$\nabla l(\mathbf{X}|\vec{\beta}) = \left[\sum_{i=1}^{n} y_i-\pi_i \quad \sum_{i=1}^{n} x_{i,1}(y_i-\pi) \quad \dots \quad \sum_{i=1}^{n} x_{i,30}(y_i-\pi)\right]^T_{(1\times 31)}$$

Finally, with the gradient we can derive our hessian. Note that due to the 30 predictor variables the hessian will be a 31 by 31 matrix.

$$\begin{aligned}\nabla^2 l(\mathbf{X}|\vec{\beta}) &= -\sum_{i=1}^{n}\begin{pmatrix}1\\X\end{pmatrix}\begin{pmatrix}1 & X\end{pmatrix}\pi_i(1-\pi_i)\\ &= -\begin{pmatrix}1 & X\end{pmatrix} diag(\pi_i(1-\pi_i))\begin{pmatrix}1\\X\end{pmatrix}\end{aligned}$$

Where $X = (x_{i,1}, \dots, x_{i,30})$. Note that this matrix will always be negative definite at all parameters making the this a well behaved problem.

Develop a Newton-Raphson algorithm to estimate your model

Modifications: - I include half stepping in the Newton-Raphson method.

Want to include: - Assent direction

```
dat <- bc %>% select(-id) %>%
  rename(y = diagnosis) %>%
  mutate(y = ifelse(y=="B", 0, 1))
```

```
loglike_func <- function(dat, betavec){
  dat_x = unname(as.matrix(dat[,-1]))
  u = dat_x%*%betavec
  pi <- exp(u) / (1+exp(u))

  # loglikelihood
  loglik <- sum(dat$y*u - log(1 + exp(u)))

  #gradient
  grad <- t(dat_x)%*%(dat$y - pi)

  # Hessian
  save_mat = matrix(0, nrow = dim(dat)[1], ncol = dim(dat)[1])
  diag(save_mat)= pi*(1-pi)
  hess = -t(dat_x)%*% save_mat %*%(dat_x)

  return(list(
    loglik = loglik,
    grad = grad,
    hess = hess
  ))

}

# loglike_func(dat, betavec = c( rep(0.03, 30)))
```

This is not working right now....

```
NewtonRaphson <- function(dat,  start, tol=100, maxiter = 20){
  i = 0
  cur = start
  stuff = loglike_func(dat, cur)
  res <- c(i=0, "loglik" = stuff$loglik,  cur, "step" = 1)
  prevloglik <- -Inf # To make sure it iterates
  while(i < maxiter && abs(stuff$loglik - prevloglik) > tol) {
    step = 1
    i <- i + 1
    prevloglik <- stuff$loglik

    # ascent direct check
    eigen_vals = eigen(stuff$hess)
    if(max(eigen_vals$values) <= 0 ){ # we don't want neg def
      hess = stuff$hess
    } else{ # if it is neg def then need to adjust
      hess = stuff$hess - (max(eigen_vals$values) + 0.1)*diag(nrow(stuff$hess))
    }
```

```
    prev  <- cur
    cur   <- prev - rep(step, length(prev))*(solve(stuff$hess) %*% stuff$grad)
    stuff <- loglike_func(dat, cur) # log-lik, gradient, Hessian

    # doing half stepping
    while(stuff$loglik < prevloglik){
      stuff <- loglike_func(dat, prev)
      step = step / 2 # this is where half steping happens
      cur <- prev - rep(step, length(prev))*(solve(stuff$hess) %*% stuff$grad)
      stuff = loglike_func(dat, cur)
    }
    # Add current values to results matrix
    res <- rbind(res, c(i, stuff$loglik, cur, step))
  }

  colnames(res) <- c(names(dat[,-1]), "i", "loglik", "step")
  return(res)
}
#Running the algorithm with good starting values:
#betavec = rep(1, 31)
betavec = c(rep(0.02, 30))
ans <- NewtonRaphson(dat,betavec)

data.frame(
  i = data.frame(ans)$i,
  step = data.frame(ans)$step,
  loglik = data.frame(ans)$loglik
)
```

```
##            i         step     loglik
## 1    0.02000 1.000000e+00    0.02000
## 2  -15.40564 1.862645e-09   84.51468
## 3  104.03746 1.562500e-02 -903.33928
## 4   12.17873 4.882812e-04   89.93042
## 5 -119.08307 1.250000e-01 -353.94531
```