ΧΑΤΖΟΠΟΥΛΟΣ ΓΕΡΑΣΙΜΟΣ

2. Ας δημιουργήσω έναν κώδικα στην γλωσσά R όπου θα κάνει εκτίμηση παραμέτρων για το μμοντέλο της λογιστικής παλινδρόμησης σε πραγματικά δεδομένα (returns of s&p 500 stock)χρησιμοποιώντας τους αναλυτικούς υπολογιστείς για την βαθμίδα και τους προσεγγιστικούς

Ας δούμε πρώτα με λόγια και με μαθηματικά τι είναι η λογιστική παλινδρόμηση και τι είναι οι παράμετροι της

Στην λογιστική παλινδρόμηση όταν είναι να κάνουμε μια πρόβλεψη για κάποιον π.χ άμα το κεφάλαιο θα ανέβει η θα κατέβει.

Λόγω της φύσης της πιθανότητας, η πρόβλεψη θα πέσει στο [0, 1].

Κατά κανόνα, εάν η προβλεπόμενη πιθανότητα είναι μεγαλύτερη ή ίση με 0,5, τότε μπορούμε να ονομάσουμε να πούμε ότι το found αυτό «προεπιλογή» εάν η προβλεπόμενη πιθανότητα είναι μικρότερη από 0,5, τότε μπορούμε να επισημάνουμε το found "μη προεπιλεγμένο". Ωστόσο, το εύρος της γραμμικής παλινδρόμησης είναι από αρνητικό άπειρο έως θετικό άπειρο, όχι σε [0, 1]. Στη συνέχεια εισάγεται η λειτουργία σιγμοειδούς για την επίλυση αυτού του προβλήματος.

Η λειτουργία σιγμοειδούς δίνει καμπύλη σχήματος S και κορεσμό όταν το επιχείρημά της είναι πολύ θετικό ή πολύ αρνητικό.

Η έκφραση της σιγμοειδούς συνάρτησης είναι:

$$f(x)=\frac{1}{1+e^{-k(x-xo)}}$$

Όπου το Χο,η τιμή του Χ για το sigmoid μέσο σημείο και το L η αυξητική λογιστική τάση

Ξέρουμε ότι το ΕΜΠ της λογιστικής παλινδρόμησης είναι

$\sum_{i=1}^{n} Y_i \cdot \ln(p_i) + (1 - Y_i) \cdot \ln(1 - p_i)$

Πριν βρούμε την βαθμίδα πρέπει να βρούμε το loss function του ΕΜΠ της λογιστικής όπου είναι η αρνητική συνάρτηση του ΕΜΠ από πάνω

Loss function=$-\sum_{i=1}^{n} Y_i \cdot \ln(p_i) + (1 - Y_i) \cdot \ln(1 - p_i)$

ΑΚΟΜΑ ΞΕΡΩ

$$\ln\left(\frac{p}{(1-p)}\right) = \beta_0 + \sum_{i=1}^{m} \beta_i x_i$$

$$\frac{p}{(1-p)} = \mathrm{e}^{\beta_0 + \sum_{i=1}^{m} \beta_i x_i}$$

$$p = (1-p)\mathrm{e}^{\beta_0 + \sum_{i=1}^{m} \beta_i x_i}$$

$$p + p * \mathrm{e}^{\beta_0 + \sum_{i=1}^{m} \beta_i x_i} = \mathrm{e}^{\beta_0 + \sum_{i=1}^{m} \beta_i x_i}$$

$$p = \frac{\mathrm{e}^{\beta_0 + \sum_{i=1}^{m} \beta_i x_i}}{1 + \mathrm{e}^{\beta_0 + \sum_{i=1}^{m} \beta_i x_i}} =$$

$$p_i = \frac{1}{1 + \exp(-\langle X_i, w \rangle)} =$$

$$p_i = \left(1 + \exp\left[-\mathbf{x}_i^T * \mathrm{b}\right)^{-1}$$

Το μόνο που μένει για να υπολογίσουμε την βαθμίδα είναι να βρούμε την πρώτη παράγωγο του loss function

Η αρνητική λογαριθμική πιθανοφάνεια είναι

$$L(w) = \left(-\sum_{i=1}^{N} (y_i \ln(p) + (1 - y_i)\ln(1-p))\right)$$

$$= \sum_{i=1}^{N} [y_i \ln'(p) + (1 - y_i)\ln'(1-p)]$$

$$= \sum_{i=1}^{N} \left[y_i \frac{1}{p} p' + (1 - y_i)\frac{1}{1-p}(1-p)'\right]$$

Άμα εφαρμόσουμε εδώ την sigmoid που αναφέραμε στην αρχή

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

τότε θα καταλήξουμε στο gradient της λογιστικής παλινδρόμησης όπου είναι

$$= \sum_{i=1}^{N} \left((p - y_i)x_i\right)$$

Σε πινακοειδής μορφή αυτή η ισότητα γίνεται

$$DF(X) = X^T(Y - P)$$

5)

CODE IN R language

Ας πάω στην δημιουργία του κώδικα μου που θα κάνει την παραπάνω δουλεια

Πάμε να βάλω δικά μου αληθινά δεδομένα ,θα χρησιμοποιήσω τα δεδομένα Weekly από το πακέτο ISLR οπου είναι weekly percentage returns for the S&P 500 stock index between 1990 and 2010

Α) για τον αναλυτικό υπολογισμό βαθμίδας

```
library(ISLR)#τα δεδομένα μας
library(dplyr)
#Weekly data
Weekly <- Weekly %>%
   mutate(Direction = ifelse(Direction == "Down",0,1))
n_features=5
y<-Weekly$Direction
y<-c(y)
 x1<-Weekly$Lag1
 x2<-Weekly$Lag2
 x3<-Weekly$Lag3
 x4<-Weekly$Lag4
 x5<-Weekly$Lag5
X <- cbind(rep(1,length(x1)), x1, x2, x3, x4, x5)
model.2 <- glm(y~x1+x2+x3+x4+x5, family = binomial)
theta_true <- coef(model.2)
theta_init<-rep(0,n_features+1)

# Gradient Descent Method (Analytical Solution)
```

```
# The Loss function to be minimized
Loss<-function(X,y,theta){
   L= -sum(-y*log(1 + exp(-(X%*%theta))) - (1-y)*log(1 + exp(X%*%theta)))
 return(L)
}
# Gradient Descent
gradient_descent<-function(X,y,theta,alpha,max_iter){
 L_history<-rep(0,max_iter)
 for(i in 1:max_iter){
 theta<-theta-alpha*(t(X)%*%(1/(1+exp(-X%*%theta))-y)) # gradient calculation
 L_history[i]<-Loss(X,y,theta)
 }
 results<-list(theta,L_history)
 return(results)
}
#  now Execute
alpha<-0.001
max_iter<-100
results<-gradient_descent(X,y,theta_init,alpha,max_iter)
theta<-results[[1]] # retrieve theta vector
L_history<-results[[2]] # retrieve L_history
# compare theta with theta_true
theta_true
```

```
(Intercept)          x1                  x2                  x3          x4
 0.23029037 -0.04009730  0.06015073 -0.01508114 -0.02677052
         x5
```

-0.01348731

theta

[,1]

0.23029037

x1 -0.04009730

x2 0.06015073

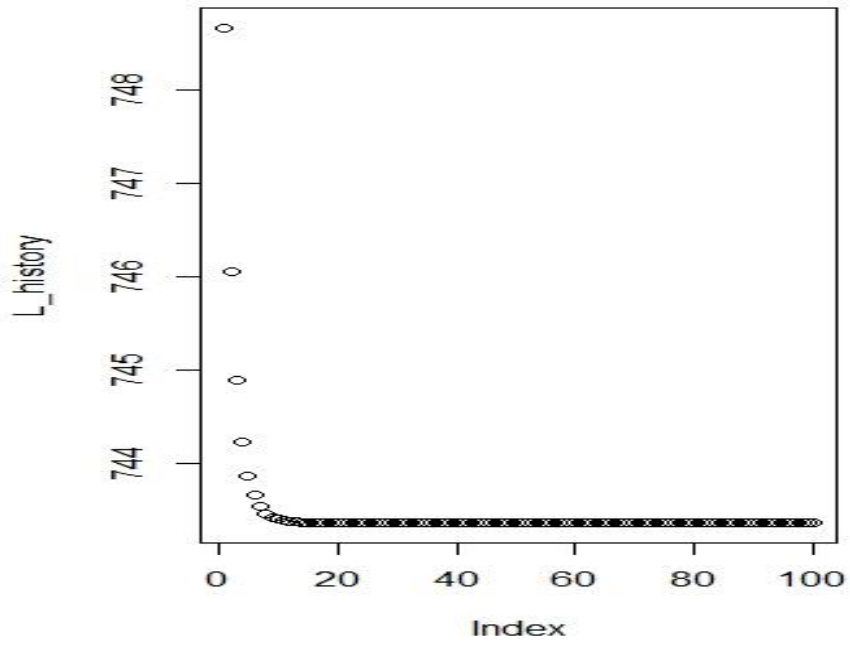
x3 -0.01508114

x4 -0.02677052

x5 -0.01348731

plot Loss function as a function of the iteration number

plot(L_history)



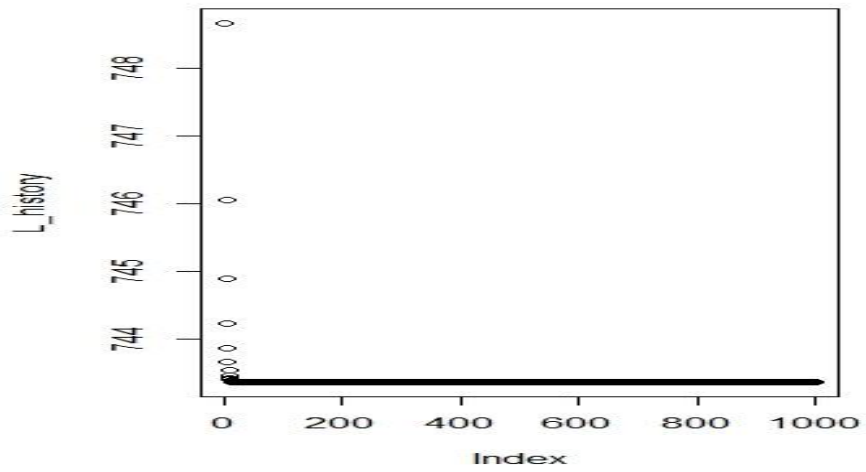
Παρατηρώ ότι συγκλίνει σχεδόν αμέσως και οι τιμές μ ταυτίζονται με τις πραγματικές πράγμα που σημαίνει ότι ο αλγόριθμος έκανε άριστη δουλεία και ηταν αρκετά γρήγορος για την εκτίμηση των παραμέτρων καθώς από την $8^{η}$ επανάληψη κατάφερε να προσεγγίσει τέλεια τις πραγματικές μας τιμες των δεδομένων μας

Όσον αφορά το ίδιο πρόβλημα αλλά με την αριθμητική προσέγγιση της βαθμίδας αυτήν την φορά

```
library(ISLR)

library(dplyr)

#Weekly data
#προσοχή άμα τρέξετε το πρόγραμμα αμέσως μετά το προηγούμενο πρέπει να γίνει clear environment αλλιώς η mutate θα τα  κάνει όλα 1
Weekly <- Weekly %>%

   mutate(Direction = ifelse(Direction == "Down",0,1))

n_features=5

y<-Weekly$Direction

y<-c(y)

 x1<-Weekly$Lag1

 x2<-Weekly$Lag2

 x3<-Weekly$Lag3

 x4<-Weekly$Lag4

 x5<-Weekly$Lag5

X <- cbind(rep(1,length(x1)), x1, x2, x3, x4, x5)

model.2 <- glm(y~x1+x2+x3+x4+x5, family = binomial)

theta_true <- coef(model.2)

theta_init<-rep(0,n_features+1)


# Gradient Descent Method (Analytical Solution)

# The Loss function to be minimized

Loss<-function(X,y,theta){

   L= -sum(-y*log(1 + exp(-(X%*%theta))) - (1-y)*log(1 + exp(X%*%theta)))
```

```
 return(L)
}
grad_vec<-function(x,y,theta)
{ delx=0.0001
grad_vec=c()
for(i in 1:length(theta))
{ x_old=theta
x_new=theta
x_old[i]=theta[i]-delx
x_new[i]=theta[i]+delx
grad_vec[i]=(Loss(x,y,x_new)-Loss(x,y,x_old))/(2*delx)
}
return(grad_vec)
}
# Gradient Descent
gradient_descent<-function(x,y,theta,alpha,max_iter){
 L_history<-rep(0,max_iter)
 for(i in 1:max_iter){
 theta<-theta-alpha*grad_vec(x,y,theta) # gradient calculation
 L_history[i]<-Loss(x,y,theta)
 }
 results<-list(theta,L_history)
 return(results)
}
# now execute
alpha<-0.001
```

```
max_iter<-1000

results<-gradient_descent(X,y,theta_init,alpha,max_iter)

theta<-results[[1]] # retrieve theta vector

L_history<-results[[2]] # retrieve L_history

# compare theta with theta_true

theta_true

(Intercept)          x1                  x2                  x3                  x4

 0.23029037 -0.04009730  0.06015073 -0.01508114 -0.02677052

          x5

-0.01348731

theta

[1]  0.23029037 -0.04009730  0.06015074 -0.01508114 -0.02677052

[6] -0.01348731

# plot Loss function as a function of the iteration number

plot(L_history)
```



Παρατηρώ ότι συγκλίνει σχεδόν αμέσως και οι τιμές μ ταυτίζονται με τις πραγματικές πράγμα που σημαίνει ότι ο αλγόριθμος έκανε άριστη δουλεία και ήταν αρκετά γρήγορος για την εκτίμηση των παραμέτρων καθώς από την $7^η$ επανάληψη κατάφερε να προσεγγίσει τέλεια τις πραγματικές τιμές των δεδομένων μας

Ευχαριστώ