############################################

# Statistical Modelling & Machine Learning #

# R Example1 #

############################################

###################################################################

# Modelling Example 1: Nonlinear model with nonconstant variance #

###################################################################

library(datasets)

?attenu

dim(attenu)

##### Linear regression #####

fit1 = lm(accel ~ mag + dist, data=attenu)

par(mfrow = c(2,2))

plot(fit1)

summary(fit1)

par(mfrow = c(1,1))

# mag, dist 둘다 유의하다. coefficient of determination 자체는 별로 높지 않다.

# 이 inference는 error term이 normality를 따른다는 가정 하에 세워진 추정이기 때문에 해당 부분에 대한 검토가 필요하다.

#error term의 normality를 확인하기 위해 residual plot을 그려보겠다. -> residual에 패턴이 존재한다. 이분산이 존재하는 것으로 추정.

#분산에 대한 조정을 하기에는 아무런 정보가 없어 mean function부터 추정해야 한다.

#선형이 아닌 회귀모형이 필요한데 아무런 정보가 없는 상태이다. 이렇게 정보가 아무것도 없을 때 사용할 수 있는 것은 GAM이다.

##### GAM #####

library(gam)

fit2 = gam(accel ~ s(mag,5) + s(dist,5),data=attenu) # Degree of Freedom = 5인 smoothing spline을 각 변수에 적용, 여기서 degrees of freedom은 데이터 얼마나 탄력적으로 반응하나의 정도인 것 같다.

fit2 = gam(accel ~ s(mag,2) + s(dist,2),data=attenu)

fit2 = gam(accel ~ s(mag,3) + s(dist,3),data=attenu)

fit2 = gam(accel ~ s(mag,4) + s(dist,4),data=attenu)

par(mfrow=c(1,2))

plot(fit2) # dist는 exponential하게 감소됨이 명확하다. mag는 linear한지 nonlinear한지 명확하지 않기 때문에 test를 해봐야 한다. anova를 해서 F-test를 해볼 수 있다.

par(mfrow=c(1,1))

fit2\_1 = gam(accel ~ mag + s(dist,5),data=attenu) #mag를 비선형으로 가정한 fit2와 선형으로 가정한 fit2\_1 비교

anova(fit2,fit2\_1) # p-value가 유의하게 나왔으므로 비선형으로 가정하는게 적절하다.

# mag: non-linear function, dist: exponential function

########## Nonlinear Model with constant variance ##########

# Y = accel, X1 = mag, X2 = dist.

# Nonlinear model: Y = beta1 + beta2\*X1 + beta3\*X1^2 + beta4\*exp(-beta5\*X2).

# GAM을 통해 얻은 정보로 parametric nonlinear regression을 세울 수 있다. 5 parameters

# beta4는 decreasing scale을 잡아주는 역할, beta5는 decreasing rate을 조절하는 역할

# mag에 대해서는 polynomial regression, dist에 대해서는 exponential regression을 취한다.

# dist가 비선형이라 LSE와 같은 잣대로 결정할 수 없다.

#추정한 모델함수 만들기

# exponential regression 파트 때문에 일반적인 linear regression technique을 통해 beta값들을 추정할 수 없다.

f = function(beta,X)

{

X1 = X[,1]; X2 = X[,2]

beta[1] + beta[2]\*X1 + beta[3]\*X1^2 + beta[4]\*exp(-beta[5]\*X2)

}

# Objective function: RSS

RSS = function(beta,Y,X) sum((Y-f(beta,X))^2)

# Gradient vector of the objective function

# RSS를 각 parameter마다 편미분

grv = function(beta,Y,X)

{

X1 = X[,1]; X2 = X[,2]

R = Y - f(beta,X)

c(-2\*sum(R), -2\*sum(R\*X1), -2\*sum(R\*X1^2), -2\*sum(R\*exp(-beta[5]\*X2)),

2\*beta[4]\*sum(R\*X2\*exp(-beta[5]\*X2)))

}

# Optimization

X = cbind(attenu$mag,attenu$dist)

colnames(X) = c('mag', 'dist')

Y = attenu$accel

ml1 = optim(par = rep(0.1,5), fn = RSS, gr=grv, method='BFGS', X=X, Y=Y) # par = initial Beta's, fn = optimize할 objective function, gr = gradient vector를 지정하는 함수. "BFGS" is a quasi-Newton method.

# gradient vector는 objective function을 각 parameter에 대해서 1차 미분을 한 것이다. 따로 지정 안해줘도 알아서 numerical하게 값을 찾아준다. 그러나 따로 gradient vector를 계산해서 넣어주면 더 잘 돌아간다. Hessian Matrix도 넣어주면 좋긴 하다.

ml1

# ml1으로 새로운 beta coefficients들을 구하였으므로 이것에 대한 residual check 역시 해야 한다.

beta.hat = ml1$par

beta.hat

# Fitted value

Yhat = f(beta.hat,X)

# Residual plot

r = Y - Yhat

par(mfrow=c(1,1))

plot(Yhat,r,ylim=c(-0.5,0.5))

lines(c(-10,10),c(0,0),col='red')

# Mean pattern은 없어지는 것으로 나온다.

# Linearly increasing variance pattern.

#residual의 분산이 점점 커지는 패턴을 가짐을 확인할 수 있음. 등분산 X

# 이럴 경우 Linear Variance Function을 사용하는게 적절하다.

######### Nonlinear model with nonconstant variance ##########

# To check whether a matrix is singular or not

# install.packages('matrixcalc')

library(matrixcalc)

# Objective function for mean function: Genearalized least square method.

obj.mean = function(beta,Y,X,S) t(Y-f(beta,X)) %\*% solve(S) %\*% (Y-f(beta,X)) # solve(S) = the inverse of S

# S: Covariance matrix

#5주차 1차시 52분 19초에 나오는 식을 구현한 것

# Gradient vector of the objective function

gr.mean = function(beta,Y,X,S)

{

sigma2 = diag(S)

X1 = X[,1]; X2 = X[,2]

R = Y - f(beta,X)

c(-2\*sum(R/sigma2), -2\*sum(R\*X1/sigma2), -2\*sum(R\*X1^2/sigma2),

-2\*sum(R\*exp(-beta[5]\*X2)/sigma2),

2\*beta[4]\*sum(R\*X2\*exp(-beta[5]\*X2)/sigma2))

}

# Linear variance function: |r| = gam1 + gam2\*Yhat. 각 r이 variance-covariance matrix의 diagonal term이 된다.

# For linear variance function, we can consider absolute residuals,

# instead of squared residuals.

# gam.hat = (Z^T W Z)^(-1) Z^T W |r|.

beta.new = ml1$par # initial parameter.

W = diag(rep(1,length(Y))) # W에 대한 정보는 따로 없으므로 등분산 가정

mdif = 100000

while(mdif > 0.000001)

{

Yhat = f(beta.new,X)

r = Y - Yhat #보통 squared residuals를 사용하는데 이 예제에서는 absolute residuals를 사용하였다.

Z = cbind(1,Yhat)

gam.hat = solve(t(Z) %\*% W %\*% Z) %\*% t(Z) %\*% W %\*% abs(r)

sigma = Z %\*% gam.hat

S = diag(as.vector(sigma^2))#objective mean function에 사용될 sigma

if (is.non.singular.matrix(S)) W = solve(S)

else W = solve(S + 0.000000001\*diag(rep(1,nrow(S)))) #variance function구할 때의 weights

ml2 = optim(beta.new, obj.mean, gr=gr.mean,method='BFGS', Y=Y, X=X, S=S)

beta.old = beta.new

beta.new = ml2$par

mdif = max(abs(beta.new - beta.old))

}

beta.new

Yhat = f(beta.new,X)

sigma = Z %\*% gam.hat

r = (Y - Yhat)/sigma

# Residual plot

plot(Yhat,r,ylim=c(-4,4))

lines(c(0,10),c(0,0),col='red')

#random error의 형태를 띔.

# 최종 모델은 이 variance function을 포함한 모델이다.

##### Linear regression with nonconstant variance #####

# lmvar:

# Linear mean function

# Linear variance function: log(sigma) = X\*beta

# install.packages('lmvar')

library(lmvar)

X\_s = cbind(attenu$mag, attenu$dist)

colnames(X\_s) = c('mag', 'dist')

fit3 = lmvar(attenu$accel, X, X\_s)

summary(fit3)

ms = predict(fit3, X\_mu=X, X\_sigma=X\_s)

r1 = (Y - ms[,1])/ms[,2]

plot(ms[,1],r1)

lines(c(-10,10),c(0,0),col='red')

#5주차 1차시 끝

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

######################################################

# Modelling Example 2: Model with time correlations #

######################################################

tsdat = read.table('tsdat.txt',header=T)

fit = lm(y ~ x, data=tsdat)

summary(fit)

par(mfrow=c(2,2))

plot(fit)

# 등분산, normality를 만족하는 것으로 보인다.

#하지만 time series이기 때문에 linear test를 해봐야 한다. Durbin-Watson Test

# Durbin-Watson test

# install.packages('lmtest')

library(lmtest)

dwtest(fit) # time correlation이 있음이 명확하다.

# Check ACF & PACF

# install.packages('astsa')

library(astsa)

# AR(p): ACF: Exponentially decreasing; PACF: Non-zero values at first p lags.

# MA(q): ACF: Non-zero values at first q lags; PACF: Exponentially decreasing.

# ARMA(p,q): ACF: Similar to ACF of AR(p); PACF: Similar to PACF of MA(q).

acf2(residuals(fit)) # acf는 exponentially decreasing하다는 것을 알 수 있고, pacf는 1 이후로 correlation이 0에 가깝다는 것을 알 수 있다.

# lag = 1 이면 어지간한 time correlation을 잡을 수 있겠다는 추정이 가능 => AR(1)모델 사용

ar1 = sarima (residuals(fit), p = 1,d = 0,q = 0, no.constant=T) #AR(1)

# p = AR의 차수, q = MA의 차수, no.constant = T : mean = 0 가정, d는 어차피 시험에 나오지도 않을거임

ar1$fit #time dependency가 사라짐.

# 그럼 이걸 이용해서 어떻게 beta를 추정할 것인가의 문제를 풀면 된다.

# MLE: Multivariate normal distribution

X = cbind(1,tsdat$x)

Y = tsdat$y

n = length(Y)

S = diag(rep(1,n)) # initial covariance matrix

mdif = 1000000

beta.old = rep(100000,2) # Y의 scale에 따라 Y보다는 어느 정도 크게 잡아야 하는 것 같다.

while(mdif > 0.0000001)

{

beta.new = as.vector(solve(t(X) %\*% solve(S) %\*% X) %\*%t(X) %\*% solve(S) %\*% Y) # Weighted Least Squares

r = as.vector(Y - (X %\*% beta.new))

ar1 = sarima (r, 1,0,0, no.constant=T, details=F)

alpha = ar1$fit$coef # AR(1)모델의 coefficient

sigma2 = ar1$fit$sigma2 # AR(1)모델의 sigma squared

mdif = max(abs(beta.new - beta.old))

beta.old = beta.new

# Construct covariance matrix

S = matrix(nrow=n,ncol=n)

for (i in 1:n)

{

for (j in 1:n)

{

if (i == j) S[i,j] = 1

if (i != j) S[i,j] = alpha^(abs(i-j))

}

}

S = (sigma2 / (1-alpha^2)) \* S # updating covariance matrix

}

round(beta.new,4)

# MLE: Product of conditional distribution (Approximation)

# Y\_t | Y\_t-1 ~ N(X\_t\*beta + alpha\*epsilon\_t-1, sigma^2)

fit = lm(y ~ x, data=tsdat)

Yt = tsdat$y[2:n]

Xt = tsdat$x[2:n]

et = residuals(fit)[1:(n-1)] # Y\_t | Y\_t-1 ~ N(X\_t\*beta + alpha\*epsilon\_t-1, sigma^2)이니까 n-1까지

mdif = 10000

b.old = rep(0,3)

while(mdif > 0.0000001)

{

fit.temp = lm(Yt ~ Xt + et)

b.new = fit.temp$coefficient

mdif = max(abs(b.new[1:2] - b.old[1:2]))

et = (Y - X %\*% b.new[1:2])[1:(n-1)]

b.old = b.new

}

round(b.new,4)

# Built-in function

# cochrane.orcutt => f: linear model, error: AR(p) process.

# install.packages("orcutt")

library(orcutt)

fit = lm(y ~ x, data=tsdat)

cochrane.orcutt(fit)

#########################################################

# Modelling Example 3: Model with spatial correlations #

#########################################################

# install.packages('spdep')

library(spdep)

data(oldcol)

?COL.OLD

# 'COL.nb' has the neighbors list.

# 2-D Coordinates of observations

crds = cbind(COL.OLD$X, COL.OLD$Y)

# Compute the maximum distance

mdist = sqrt(sum(diff(apply(crds, 2, range))^2))

# All obs. between 0 and mdist are identified as neighborhoods.

dnb = dnearneigh(crds, 0, mdist)

# Compute Euclidean distance between obs.

dists = nbdists(dnb, crds)

# Compute Power distance weight d^(-2)

glst = lapply(dists, function(d) d^(-2))

# Construct weight matrix with normalization

# style='C': global normalization; 'W': row normalization

lw = nb2listw(dnb, glist=glst, style='C')

# Spatial Autoregressive Model

fit = lagsarlm(CRIME ~ HOVAL + INC, data=COL.OLD, listw=lw) #listw : normalized weight matrix를 넣는 곳인 것 같은데

summary(fit)

#ƒ

# install.packages('spatialreg')

library(spatialreg)

# Fitted values

predict(fit)

###################################################

# Modelling Example 4: Generalized Linear Models #

###################################################

########## Cumulative logit model ##########

# install.packages('ordinal')

library(ordinal)

?wine

?clm

fit = clm(rating ~ temp + contact, data=wine, link = 'logit')

summary(fit)

########## Poisson regression model ##########

# For data

# install.packages('lme4')

library(lme4)

data(grouseticks)

?grouseticks

head(grouseticks)

hist(grouseticks$TICKS,breaks=0:90)

fit = glm(TICKS ~ HEIGHT\*YEAR, data = grouseticks, family=poisson)

summary(fit)

########## Negative binomial regression model ##########

library(MASS)

fit1 = glm.nb(TICKS ~ HEIGHT\*YEAR, data = grouseticks, link=log)

summary(fit1)

########## Proportional hazard model ##########

library(survival)

# For data

# install.packages('carData')

library(carData)

?Rossi

Surv(Rossi$week, Rossi$arrest)

fit = coxph(Surv(week,arrest) ~ fin + age+ race + wexp + mar + prio,

data=Rossi)

summary(fit)

# Estimated survival function

plot(survfit(fit),ylim=c(0.6,1),xlab="Weeks", ylab="Prop.of Not Rearrested")

# Estimated survival functions for financial aid

Rossi.fin = with(Rossi, data.frame(fin=c(0, 1), age=rep(mean(age), 2),

race=rep(mean(race=='other'),2),

wexp=rep(mean(wexp=="yes"),2),

mar=rep(mean(mar=="not married"),2),

prio=rep(mean(prio),2)))

plot(survfit(fit,newdata=Rossi.fin), conf.int=TRUE,

lty=c(1, 2), ylim=c(0.6, 1), col=c('red','blue'),

xlab="Weeks", ylab="Prop. of Not Rearrested")

legend("bottomleft", legend=c("fin = no","fin = yes"),

lty=c(1 ,2),col=c('red','blue'))