201111774 응용통계학과 박성진

(homework)

L1 , L2 , Huber loss 를 사용한 model 생성후 hyper-parameter Tuning

compare the model performance when the losses are square, least absolute deviation and Huber with AIC and scaled error

1. 실습데이터 old.sam.for.reg.fit.csv, old.sam.for.reg.pred.csv 활용

- simple linear regression을 위해 Response variable 인 sensitivity와 독립변수 V1을 사용

sensitivity	type	pressure	V1	V2
-35.0045100125574	A	Low	0.388554459691911	0.178254322088359
-19.0083036559359	A	Low	1.32924963453161	2.1765764989388
-22.9386687875601	C	Low	-0.0780379592228282	-1.03532541534334
-25.8408598630724	B	Low	1.06786222554256	-0.963499083868921
-31.666275032004	A	Low	-0.647135580065748	0.832282595264365
-6.15286069954251	A	Low	1.44726576684543	-0.854812020215949
10.028102927778	C	Low	-0.295068879423137	-1.90167408935838
-8.40775040321761	A	Low	-0.257069923898074	-0.717552123159869
-16.9065410662865	B	Low	-0.939155151050187	-0.539411824773062
-32.7561323090383	A	Low	-0.617662880791346	-0.471152320105937

- train 데이터와 test 데이터 load 후 , 세가지 모델 비교할 matrix 생성

```
# data load
# train_data
df<-read.csv(file="old.sam.for.reg.fit.csv")
tr.df = df[,c(1,4)]
#test_data
test.df<-read.csv(file="old.sam.for.reg.pred.csv")
te.df =test.df[,c(1,4)]
# result matrix
rst.mat = matrix(NA,2,3)
colnames(rst.mat)=c("lm","rq","hub")
rownames(rst.mat)=c("AIC","scaled_error")
rst.mat

#                lm rq hub
#AIC            NA NA  NA
#scaled_error NA NA  NA
```

2. 모델 적합 - square loss(L2), absolute deviation loss(L1), Huber loss

```
#models = simple linear regression + L2, L1, huber
library(robustreg)
library(quantreg)
#####################################################################3
#L2
lm.fit = lm(formula = sensitivity~V1,data=tr.df)
#L1
rq.fit=rq(formula = sensitivity~V1,data=tr.df)
#Huber
rb.fit=robustRegH(formula = sensitivity~V1,data=tr.df)
#####################################################################
```

3. AIC와 scaled_error 구현 후 세가지 모델 평가

```r
# implement AIC and Scaled Error
aic.fun <- function(loss){
  sse = sum((loss)^2)
  return(log(sse/length(loss))+2*2/length(loss))
}

scaled_error.fun <- function(loss){
  sqrt(sum((loss)^2)/length(loss))

}
```

---

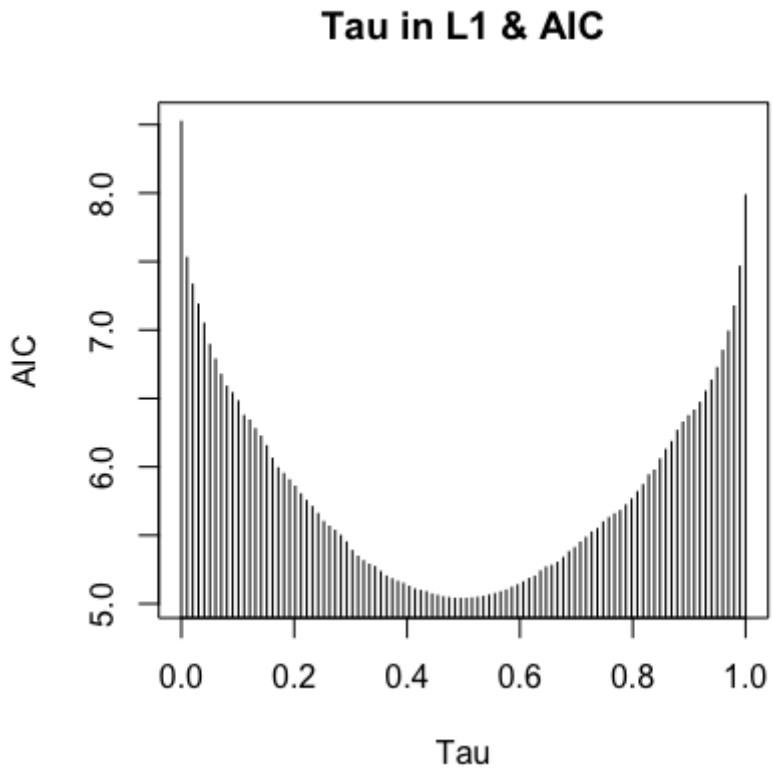
	lm	rq	hub
AIC	5.036964	5.036357	5.035965
scaled_error	12.164014	12.160321	12.157939

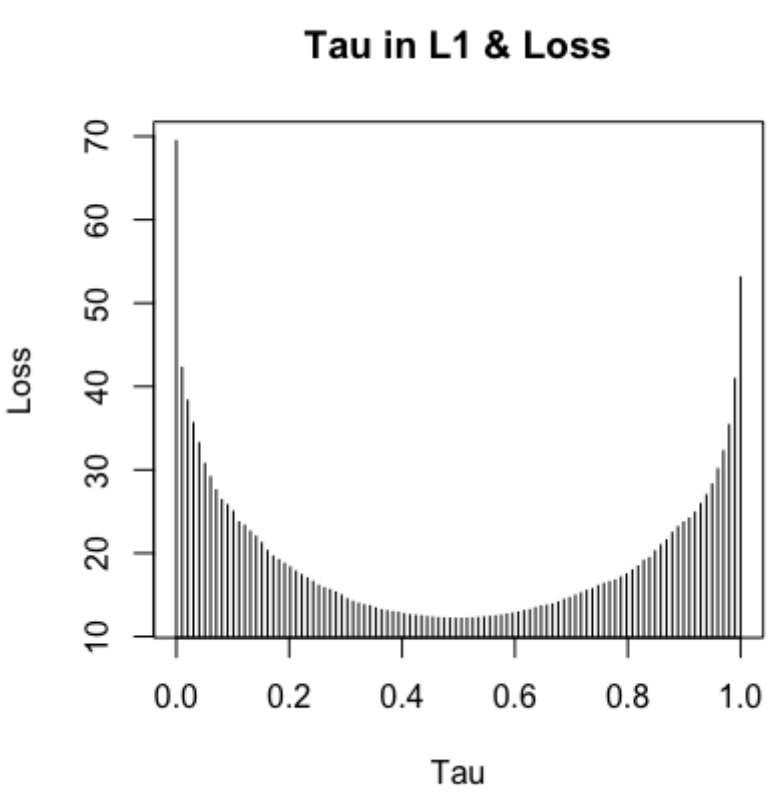
- lm 경우, 모델 성능에 영향을 미치는 hyper parameter가 존재하지 않지만,
- rq나 huber은 모델 성능에 영향을 미치는 hyper parameter가 존재

4. absolute deviation loss의 Tau, Huber loss의 tune 값을 조정

1) absolute deviation loss - Tau

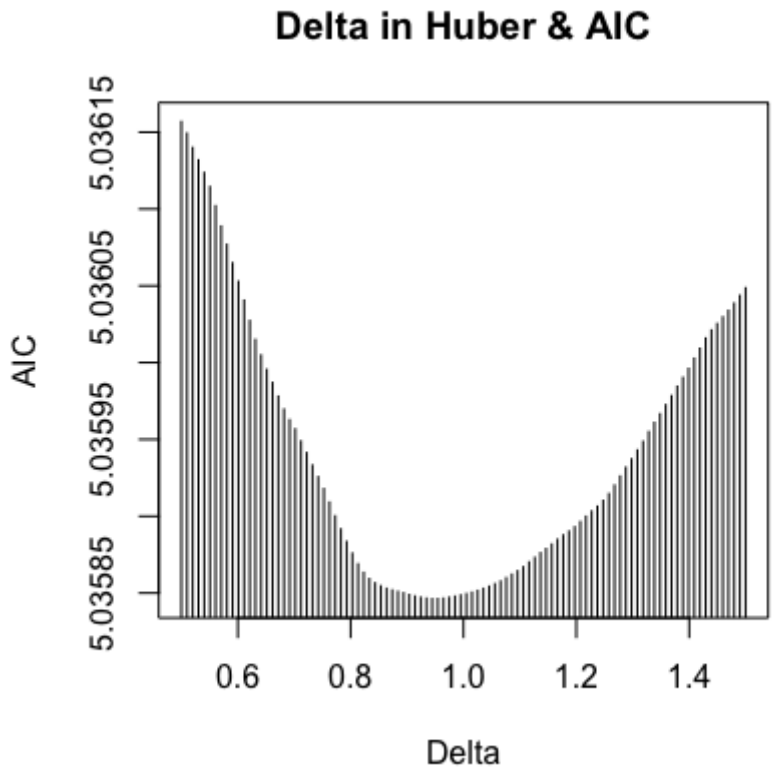
```
# grid search for [L1] & Huber
L1.mat = matrix(NA,3,100)
rownames(L1.mat)=c("Tau","AIC","Prediction")
# tau
tau.tune=seq(0,1,length.out = 100)

for(j in 1:100){
    L1.mat[1,j] = tau.tune[j]
    rq.fit=rq(formula = sensitivity~V1,tau=tau.tune[j] ,data=tr.df)

    test_pred=cbind(1,te.df[,2])%*%coef(rq.fit)
    loss=te.df[,1]-test_pred
    aic.rq = aic.fun(loss)
    L1.mat[2,j]=aic.rq

    square_loss=sqrt(sum((loss)^2)/dim(te.df)[1])
    L1.mat[3,j]=square_loss
}
```

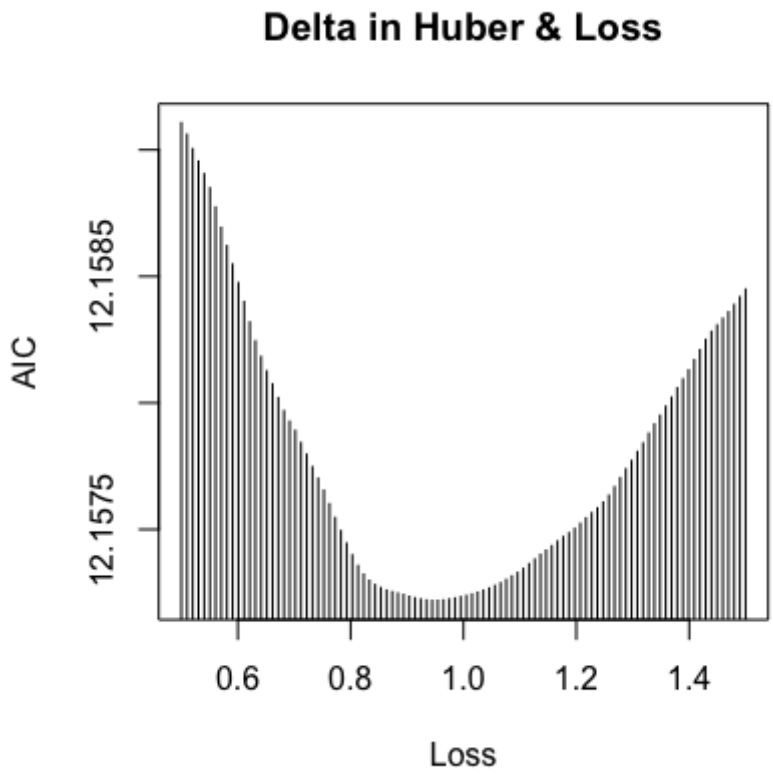



Tau in L1 & Loss
Loss
10
20
30
40
50
60
70
0.0
0.2
0.4
0.6
0.8
1.0
Tau

1) Huber loss - Delta

```
# grid search for L1 & [Huber]
hub.mat = matrix(NA,3,100)
rownames(hub.mat)=c("delta","AIC","Prediction")
#delta
hub.tune=seq(0.5,1.5,length.out = 100)

for(j in 1:100){
  hub.mat[1,j] = hub.tune[j]
  rb.fit=robustRegH(formula = sensitivity~V1,tune=hub.tune[j] ,data=tr.df)

  test_pred=cbind(1,te.df[,2])%*%coef(rb.fit)
  loss=te.df[,1]-test_pred
  aic.hub = aic.fun(loss)
  hub.mat[2,j]=aic.hub

  square_loss=sqrt(sum((loss)^2)/dim(te.df)[1])
  hub.mat[3,j]=square_loss

}
```





5. result

	lm	rq	hub
best_Tau	*NA*	0.4949495	*NA*
Best_delta	*NA*	*NA*	0.9444444
AIC	5.036964	5.0359300	5.0358462
scaled_error	12.157939	12.1577278	12.1572186

- 모델을 평가할 measure로 AIC, scaled_error를 쓸때 huber loss 의 delta를 0.9444로 할때의 performance 가 가장 좋다