**Série Sélection d'attributs et régression (Ridge, Lasso, ACP).**

**2CS: SITG1,SITG4,SILG1.**

**Mme Hamdad, ESI.**

**1- Approche filtre**:

Extraire les attributs pertinents en utilisant le gain d'information sur les datasets Glass, Shuttle, Sonar en suivant le code suivant:.

library(FSelector)

**Exemple sur Iris**

data=iris

Gain d'information:

weights <- information.gain(Species~., data)

print(weights)

prendre les k meilleurs (k=2)

subset <- cutoff.k(weights, 2)

ou prendre un % d'attributs

subset1 <- cutoff.k.percent(weights,0.5)

utiliser cette formule dans le classifieur

f <- as.simple.formula(subset, "Species")

print(f)

- Utiliser une autre évaluation (gain.ratio, symmetrical.uncertainty,......).

- Effectuer une classification par SVM avant sélection et après sélection. Que constatez vous en terme de performance et de temps d'exécution.

- Refaire sur Shuttle, Glass et Sonar.

- A chaque fois, comptabiliser le temps d'exécution selon:

Le temps d'exécution :

T1<-Sys.time()

svm=svm(Class~.,data)

T2<-Sys.time()

T2-T1

T1<-Sys.time()

svm1=svm(f,data)

T2<-Sys.time()

T2-T1

**2-Approche enveloppante:**

Dans cette approche, la sélection est partie prenante de la classification supervisé:

- Comprendre le script suivant et déterminer les attributs pertinents:

evaluator <- function(subset) {

#k-fold cross validation

k <- 5

splits <- runif(nrow(iris))

results = sapply(1:k, function(i) {

test.idx <- (splits >= (i - 1) / k) & (splits < i / k)

train.idx <- !test.idx

test <- iris[test.idx, , drop=FALSE]

train <- iris[train.idx, , drop=FALSE]

tree <- rpart(as.simple.formula(subset, "Species"), train)

error.rate = sum(test$Species != predict(tree, test, type="c")) / nrow(test)

return(1 - error.rate)

})

print(subset)

print(mean(results))

return(mean(results))

}

subset <- best.first.search(names(iris)[-5], evaluator)

f <- as.simple.formula(subset, "Species")

print(f)

- Changer best.first.search par exhaustive.search, forward.search, Backward.search. Quelle est la différence entre les résultats de ces différentes questions?

- A l'intérieur de la procédure, remplacer rpart par svm. Comparer les résultats?

- Refaire la même chose avec Sonar.

**Exercice3**: sur mlbench, considérer le dataset BostonHousing:

- Faire une ACP normée.

- Retenir les composantes principales dont le %d'inertie est >80%

- Faire une régression linéaire, en considérant ces composantes principales

- Comparer l'erreur obtenue avec celle de la régression linéaire sur les variables initiales.

**Exercice4**: **Régression Ridge:**

**Comme il s'agit de la régression nous allons travailler sur airquality ou bien housing price.**

data(BostonHousing)

data= BostonHousing

summary(BostonHousing)

dim(BostonHousing)

names(BostonHousing)

cor(BostonHousing)

- Effectuer un régression linéaire. Quelles sont les variables non significatives?

mod=lm(medv~.,data)

- Supprimer ces variables et refaire la régression

Que constatez vous?

- Calculer l'erreur

sse =sum((data$medv − mod$fitted)^2)

sse

- Nous effectuons une régression pinalisée Ridge!

library(MASS)

mod.ridge=lm.ridge(medv~.,data, lambda=seq(0,20,0.1))

par(mfrow=c(2,1))

plot(mod.ridge)

matplot(t(mod.ridge$coef),lty=1:3,type="l",col=1:10)

legend("top",legend=rownames(mod.ridge$coef), col=1:10,lty=1:3)

- Détection de la pénalité optimal par cross validation

plot(mod.ridge$lambda,mod.ridge$GCV)

select(mod.ridge).

Refaire mod.ridge en mettant la valeur optimal de Lambda.

mod.ridge$coef

Comparaison avec MSE

mod.ridge<-lm.ridge(medv~crim+zn+chas+nox+rm+dis+rad+tax+ptratio+b+lstat,data, lambda=4.3)

X.matrix <- cbind(rep(1,length=length(data$medv)),data$crim, data$zn, data$chas, data$nox, data$rm, data$dis, data$rad, data$tax, data$ptratio, data$b, data$lstat)

X.matrix1=as.matrix(data[,-9])

fitted.vals <- X.matrix %\*% c(34.918471472 , -0.104315656, 0.043678727, 2.747912270, -16.683126171, 3.851764685 ,-1.426638997 , 0.272261020 ,-0.010629921 , -0.935344659, 0.009278324 ,-0.516975577 )

sse.ridge =sum((data$medv − fitted.vals)^2)

**Exercice5: LASSO**

library(lasso2)

lasso <- l1ce(medv ~ .,data, bound=(1:100)/100,absolute.t=FALSE)

coefficients=coef(lasso)

plot(lasso,col=1:11,lty=1:3,type="l")

legend("topleft",legend=colnames(coefficients), col=1:11,lty=1:3)

######

On supprime le terme constant

penalite\_relative=c(1:100)/100

matplot(penalite\_relative,coefficients[,-1],

+ lty=1:3,type="l",col=1:10)

vc=gcv(lasso)

crit.vc=vc[,"gcv"]

bound\_opt=vc[which.min(crit.vc),"rel.bound"]

lasso.opt <- l1ce (medv ~ .,data, bound=bound\_opt)

coef=coef(lasso.opt)

coef

fit=fitted(lasso.opt)

mean((fit-data[,"medv"])^2)

Comparer l"erreur avec celle De MCO.