#CV Generic Function

# Author: Kris seo

# Partner: Klara Mairentia

loss\_function <- function (y, y\_pred) {

return (mean(y != y\_pred))

}

qdaClassifier <- function(train, valid){

fit.qda <- qda(currFormula, data=train)

pred.qda <- predict(fit.qda, valid)

return(pred.qda$class)

}

logistic <- function(train,valid) {

glm.fit = glm(currFormula, data = train ,family=binomial)

glm.prob = predict(glm.fit,valid,type="response")

glm.pred <- rep(0, length(glm.prob))

glm.pred[glm.prob >= 0.5] <- 1

glm.pred[glm.prob < .5] <- -1

return (glm.pred)

}

svm <- function(train, valid) {

svm.fit <- svm(currFormula, data = train, scale = FALSE, kernel = "radial", cost = 2)

pred <- predict(svm.fit,valid)

return(pred)

}

CV\_generic <- function (data, features,classifier = logistic, folds, loss = loss\_function) {

# Data has features for training features and labels for training labels.

# classifier for lda,qda, logistic regressoin and svm.

# folds # of folds

library(caret)

folds <- createFolds(data,folds)

for (j in 1:length(folds)) {

CVVal <- data[-folds[[j]]]

CVTrain <- data[folds[[j]]]

train\_feat <- paste(features, collapse = "+")

currFormula <- as.formula(paste("label", "~", train\_feat, sep = ""))

CVVal <- droplevels(subset(ldply(CVVal, data.frame), label!=0))

CVTrain <- droplevels(subset(ldply(CVTrain, data.frame), label!=0))

if (classifier == logistic) {

CV\_loss <- loss\_function(CVVal$label,logistic(CVTrain,CVVal))

return (CV\_loss)

} else if (classifier == svm){

CV\_loss <- loss\_fucntion(CVVal$label,svm(CVTrain,CVVal))

} else {

fit <- classifier(currFormula, data = CVTrain)

pred <- predict(fit, CVVal)

CV\_loss <- loss\_function(CVVal$label,pred$class)

return (CV\_loss)

}

}

}

s---

title: "Proj2"

output: pdf\_document

author: "Kris Seo (3033384291),Klara Mairentia (3033356003)"

---

```{r}

library(ggplot2)

library(dplyr)

library(GGally)

library(MASS)

library(caret)

library(plyr)

library(zoo)

library(klaR)

library(e1071)

```

##1. Data Collection and Exploration

```{r}

img1 <- read.table("image\_data/image1.txt", header = FALSE)

img2 <- read.table("image\_data/image2.txt", header = FALSE)

img3 <- read.table("image\_data/image3.txt", header = FALSE)

colnames(img1) <- c("y\_coord", "x\_coord", "label", "NDAI","SD","CORR", "DF", "CF", "BF", "AF","AN")

colnames(img2) <- c("y\_coord", "x\_coord", "label", "NDAI","SD","CORR", "DF", "CF", "BF", "AF","AN")

colnames(img3) <- c("y\_coord", "x\_coord", "label", "NDAI","SD","CORR", "DF", "CF", "BF", "AF","AN")

```

```{r}

img1$label <- as.factor(img1$label) # Changing label variable as a factor variable.

img2$label <- as.factor(img2$label)

img3$label <- as.factor(img3$label)

all\_img <- rbind(img1, img2, img3) # Combine all three of images

all\_img <- cbind("index"= 1: nrow(all\_img), all\_img) # giving index

```

1 b) Plot well-labeled beautiful maps

There is no obvious pattern and trend among not cloud, cloud and unlabeled data points.

The dataset is not temporal, but spatial.

An i.i.d assumption for the samples is not justified for this spatial dataset because this is the case when training and testing samples are randomly drawn from the same image. Then, the dependence caused by overlap between two sets will be artiificailly improved by the spatial information processing methods, resulting a bias. Pixel data points are interdependent. If pixel data points were independent, then the processing image would have not been the same as before.

```{r}

ggplot(img1,aes(x\_coord,y\_coord)) + geom\_point(aes(color = label)) + scale\_color\_manual(name="label", labels=c("not cloud","unlabeled","cloud"), values=c("gray","black","white")) + labs(title = "Image 1") + scale\_y\_reverse()

ggplot(img2,aes(x\_coord,y\_coord)) + geom\_point(aes(color = label)) + scale\_color\_manual(name="label", labels=c("not cloud","unlabeled","cloud"), values=c("gray","black","white")) + labs(title = "Image 2") + scale\_y\_reverse()

ggplot(img3,aes(x\_coord,y\_coord)) + geom\_point(aes(color = label)) + scale\_color\_manual(name="label", labels=c("not cloud","unlabeled","cloud"), values=c("gray","black","white")) + labs(title = "Image 3") + scale\_y\_reverse()

```

Summarize the data, i.e., % of pixels for the different classes.

```{r}

prop.table(table(img1$label))

prop.table(table(img2$label))

prop.table(table(img3$label))

prop.table(table(all\_img$label))

```

1(c) Perform a visual and quantitative EDA of the data set

(i) Pairwise relationship between the features themselves

```{r}

ggpairs(all\_img[,5:7])

```

For overall data points, not considering labels, NDAI and SD have a positive correlationship about .631 between each other, NDAI and Corr also have a positive correlationship about .403 between each other.

However, the correlationship between CORR and SD is relatively lower than other correlationships.

```{r}

img\_unlabeled <- all\_img %>% filter(label == 0)

img\_no <- all\_img %>% filter(label == -1)

img\_yes <- all\_img %>% filter(label == 1)

ggpairs(all\_img, columns = 5:7, ggplot2::aes(colour=label))

```

To figure out the differences between two labels for three features, we ggpairs for all\_img and color on label.

NDAI value for cloud free data point tends to be lower than NDAI value for cloud data point as the paper said higher NDAI indicates the higher chance of presence of cloud.

There are lots of data points labeled as no-cloud in lower interval for CORR.

```{r}

par(mfrow=c(2,5))

boxplot(all\_img$x\_coord ~ all\_img$label, xlab ='label', ylab = 'x')

boxplot(all\_img$y\_coord ~ all\_img$label, xlab ='label', ylab = 'y')

boxplot(all\_img$NDAI~all\_img$label, xlab = 'label', ylab = 'NDAI')

boxplot(all\_img$SD~all\_img$label, xlab = 'label', ylab = 'SD')

boxplot(all\_img$CORR~all\_img$label, xlab = 'label', ylab = 'CORR')

boxplot(all\_img$DF~all\_img$label, xlab = 'label', ylab = 'DF')

boxplot(all\_img$CF~all\_img$label, xlab = 'label', ylab = 'CF')

boxplot(all\_img$BF~all\_img$label, xlab = 'label', ylab = 'BF')

boxplot(all\_img$AF~all\_img$label, xlab = 'label', ylab = 'AF')

boxplot(all\_img$AN~all\_img$label, xlab = 'label', ylab = 'AN')

```

(ii) Relationship between the expert labels with the individual features

```{r}

not\_cloud <- filter(all\_img, label==-1)

ggpairs(not\_cloud[, 8:12])

unlabeled <- filter(all\_img, label==0)

ggpairs(unlabeled[, 8:12])

cloud <- filter(all\_img, label==1)

ggpairs(cloud[, 8:12])

```

##2. Preparation

####2(a) Split the entire data into training, validation, and test

```{r}

split <- function (num\_block, data) {

threshold\_x <- round(seq(min(data$x\_coord), max(data$x\_coord), length.out = round(sqrt(num\_block))+1))

threshold\_y <- round(seq(min(data$y\_coord), max(data$y\_coord), length.out = round(sqrt(num\_block))+1))

split <- list()

k <-1

for (i in 1:(round(sqrt(num\_block)))) {

for (j in 1:(round(sqrt(num\_block)))) {

split[[k]] <- data %>% filter(data$x\_coord %in% (threshold\_x[i]:threshold\_x[i+1]-1), data$y\_coord %in% (threshold\_y[j]:threshold\_y[j+1]-1))

k <- k+1

}

}

return(split)

}

```

```{r}

# Dropping unlabeled data

all\_img <- all\_img %>% filter(!(label == 0))

img1 <- img1 %>% filter (!(label == 0))

img2 <- img2 %>% filter (!(label == 0))

img3 <- img3 %>% filter (!(label == 0))

```

\*\*First way\*\*

```{r}

set.seed(1234)

num\_block <- 25

split1\_img1 <- split(num\_block, img1)

split1\_img2 <- split(num\_block, img2)

split1\_img3 <- split(num\_block, img3)

split1 <- c(split1\_img1, split1\_img2, split1\_img3)

testSize <- floor((num\_block\*3)\*0.2)

valSize <- floor((num\_block\*3)\*0.1)

testIndex <- sample((1:(num\_block\*3)), testSize)

valIndex <- sample((1:(num\_block\*3))[-testIndex], valSize)

trainIndex <- (1:(num\_block\*3))[-c(testIndex, valIndex)]

testImg <- split1[testIndex]

valImg <- split1[valIndex]

trainImg <- split1[trainIndex]

```

\*\*Second Way\*\*

```{r}

set.seed(1234)

num\_block2 <- 100

split2\_img1 <- split(num\_block2, img1)

split2\_img2 <- split(num\_block2, img2)

split2\_img3 <- split(num\_block2, img3)

split2 <- c(split2\_img1, split2\_img2, split2\_img3)

testSize <- floor((num\_block2\*3)\*0.2)

valSize <- floor((num\_block2\*3)\*0.1)

set.seed(123)

testIndex2 <- sample((1:(num\_block2\*3)), testSize)

valIndex2 <- sample((1:(num\_block2\*3))[-testIndex2], valSize)

trainIndex2 <- (1:(num\_block2\*3))[-c(testIndex2, valIndex2)]

testImg2 <- split2[testIndex2]

valImg2 <- split2[valIndex2]

trainImg2 <- split2[trainIndex2]

```

\*\*Third Way\*\*

Averaging out 4 x 4 points and do the split.

```{r}

img1\_y <- img1 %>% filter(label == 1)

img1\_n <- img1 %>% filter(label == -1)

mean\_pixel1 <- as.data.frame(rollmean(img1\_y[,-3],4,align="center")) # Averaging out 4 x 4 points

mean\_pixel1$label <- 1

mean\_pixel2 <- as.data.frame(rollmean(img1\_n[,-3],4, align = "center"))

mean\_pixel2$label <- -1

mean\_img1 <- rbind(mean\_pixel1,mean\_pixel2)

mean\_img1$label <- as.factor(mean\_img1$label)

img2\_y <- img2 %>% filter(label == 1)

img2\_n <- img2 %>% filter(label == -1)

mean\_pixel21 <- as.data.frame(rollmean(img2\_y[,-3],4,align="center")) # Averaging out 4 x 4 points

mean\_pixel21$label <- 1

mean\_pixel22 <- as.data.frame(rollmean(img2\_n[,-3],4, align = "center"))

mean\_pixel22$label <- -1

mean\_img2 <- rbind(mean\_pixel21,mean\_pixel22)

mean\_img2$label <- as.factor(mean\_img2$label)

img3\_y <- img3 %>% filter(label == 1)

img3\_n <- img3 %>% filter(label == -1)

mean\_pixel31 <- as.data.frame(rollmean(img3\_y[,-3],4,align="center")) # Averaging out 4 x 4 points

mean\_pixel31$label <- 1

mean\_pixel32 <- as.data.frame(rollmean(img3\_n[,-3],4, align = "center"))

mean\_pixel32$label <- -1

mean\_img3 <- rbind(mean\_pixel31,mean\_pixel32)

mean\_img3$label <- as.factor(mean\_img3$label)

```

```{r}

num\_block <- 25

msplit1\_img1 <- split(num\_block, mean\_img1)

msplit1\_img2 <- split(num\_block, mean\_img2)

msplit1\_img3 <- split(num\_block, mean\_img3)

msplit1 <- c(msplit1\_img1, msplit1\_img2, msplit1\_img3)

testSize <- floor((num\_block\*3)\*0.2)

valSize <- floor((num\_block\*3)\*0.1)

set.seed(123)

testIndex <- sample((1:(num\_block\*3)), testSize)

valIndex <- sample((1:(num\_block\*3))[-testIndex], valSize)

trainIndex <- (1:(num\_block\*3))[-c(testIndex, valIndex)]

mtestImg <- msplit1[testIndex]

mvalImg <- msplit1[valIndex]

mtrainImg <- msplit1[trainIndex]

```

```{r}

mtrain11 <- ldply(mtrainImg, data.frame)

mtest11 <- ldply(mtestImg, data.frame)

```

2(b) Report the accuracy of a trivial classifier which sets all labels to -1 (cloud free) on the validation set and on the test set. In what scenarios will such a classifier have high average accuracy?

```{r}

valSet <- ldply(valImg, data.frame)

testSet <- ldply(testImg, data.frame)

valSet$classifier <- -1

testSet$classifier <- -1

mean(valSet$classifier == valSet$label)

mean(testSet$classifier == testSet$label)

```

A classifier have high average accuracy when there are more sets selected in cloud free label parts.

```{r}

#eliminating class label=0

trainSet <- ldply(trainImg, data.frame)

trainSet <- droplevels(subset(trainSet, label!=0))

testSet <- ldply(testImg, data.frame)

testSet<- droplevels(subset(testSet, label!=0))

valSet <- ldply(valImg, data.frame)

valSet<- droplevels(subset(valSet, label!=0))

```

2(c) Assuming the expert labels as the truth, without using fancy classification method, suggest three of the "best" features using quantitative and visual justification

```{r}

```

PC1: corresponds to AF and SD (?)

PC2: corresponds to DF and x\_coord

2(d) Write a generic CV function that takes a generic classifier, training features, training labels, number of folds K and a loss function

```{r}

train <- c(trainImg, valImg)

```

```{r}

set.seed(1234)

features <- c("NDAI", "SD","CORR")

CV\_generic(train,features, classifier = qda, folds = 4,loss)

#train\_feat <- paste(features, collapse = "+")

#currFormula <- as.formula(paste("label", "~", train\_feat, sep = ""))

#print(currFormula)

```

##3. Modeling

QDA

```{r}

set.seed(1234)

fit.qda <- qda(currFormula, data=trainSet)

pred.qda <- predict(fit.qda, testSet)

# test error rate

acc <- mean(pred.qda$class == testSet$label)

acc

#partimat(label ~ NDAI + SD + CORR, data=train11, method= "qda")

```

LDA

```{r}

lda.fit = lda(label ~ NDAI + SD + CORR , data = ,family = binomial)

lda.pred = predict(lda.fit, test11)

mean(lda.pred$class != test\_label)\*100

partimat(label ~ NDAI + SD + CORR, data=train11, method= "lda")

```

Logistic regression

```{r}

fitlog <- glm(label ~ NDAI + SD + CORR, data=train11, family = binomial)

predicted\_log <- predict(fitlog, test11, type = "response")

sum((predicted\_log> 0.5)\*1 != test\_label)/nrow(test11) \*100

fit.qda <- qda(label ~ NDAI + SD + CORR, data=train11)

pred.qda <- predict(fit.qda, test11)

test\_label <- test11$label

# test error rate

acc <- mean(pred.qda$class != test\_label)

acc

```

SVM

```{r}

svm.fit = svm(label ~ NDAI + SD + CORR , data = trainSet)

svm.pred = predict(svm.fit, testSet)

#mean($class != testSet$label)

```