Homework 4

EDUARDO GALEOTE ESCALERA

A20552496

CSP 571

Data Preparation and Analysis

November 2023

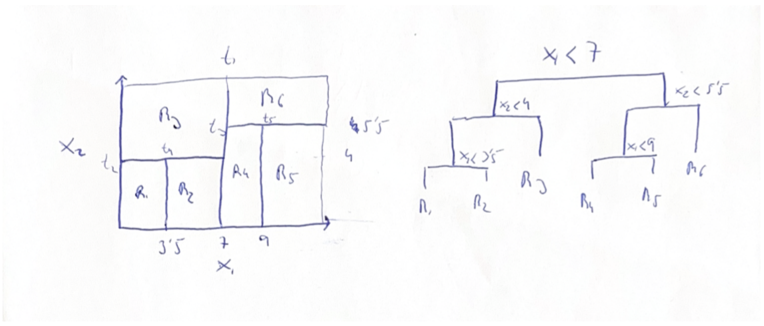
Contents

1 Recitation Exercises

1.1 Chapter 8

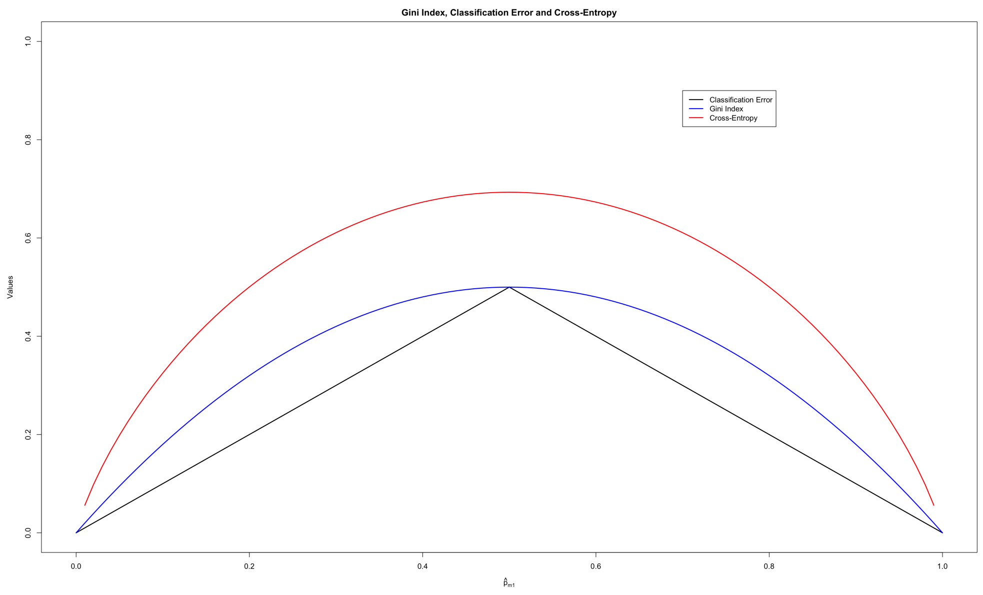
1.1.1 Excercise 1

Draw an example (of your own invention) of a partition of two-dimensional feature space that could result from recursive binary splitting. Your example should contain at least six regions. Draw a decision tree corresponding to this partition. Be sure to label all aspects of your figures, including the regions R1, R2, . . ., the cutpoints t1,t2,..., and so forth.



1.1.2 Excercise 3

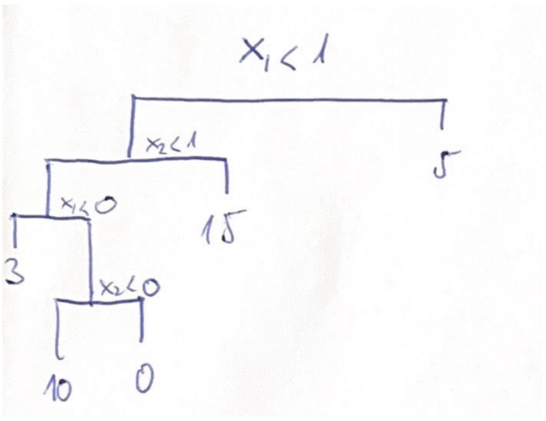
Consider the Gini index, classification error, and entropy in a simple classification setting with two classes. Create a single plot that displays each of these quantities as a function of $\hat{p}_{m1}$. The x-axis should display $\hat{p}_{m1}$, ranging from 0 to 1, and the y-axis should display the value of the Gini index, classification error, and entropy.



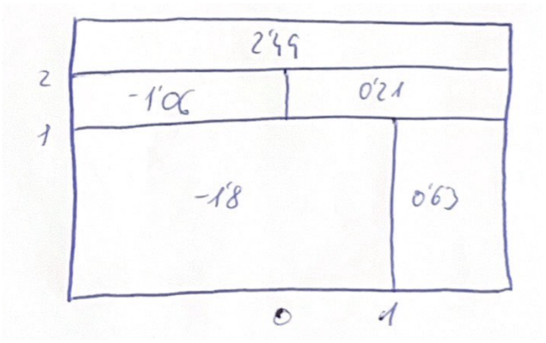
1.1.3 Excercise 4

This question relates to the plots in Figure 8.14.

a) *Sketch the tree corresponding to the partition of the predictor space illustrated in the left-hand panel of Figure 8.14. The numbers inside the boxes indicate the mean of Y within each region.*



b) *Create a diagram similar to the left-hand panel of Figure 8.14, using the tree illustrated in the right-hand panel of the same figure. You should divide up the predictor space into the correct regions, and indicate the mean for each region.*



1.1.4 Excercise 5

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of P(Class is Red—X):

0.1,0.15,0.2,0.2,0.55,0.6,0.6,0.65,0.7, and 0.75.

There are two common ways to combine these results together into a single class prediction. One is the majority vote approach discussed in this chapter. The second approach is to classify based on the average probability. In this example, what is the final classification under each of these two approaches?
Majority voting for classification: Count of P(Class is Red — X) less than 0.5 = 4 and P(Class is Red — X) $\geq$ 0.5 = 6. So X is classified as red.
Average probability: Average probability that P(Class is Red — X) is 4.5/10 = 0.45. Therefore, X is classified as green.

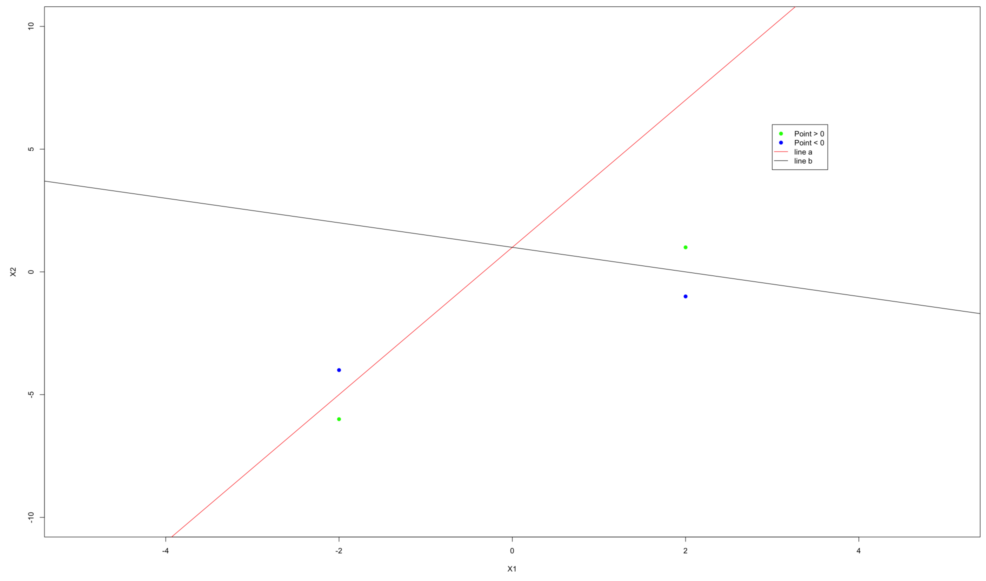
1.2 Chapter 9

1.2.1 Excercise 1

This problem involves hyperplanes in two dimensions.

a) *Sketch the hyperplane* $1+3X_1-X_2=0$. *Indicate the set of points for which* $1+3X_1-X_2>0$, *as well as the set of points for which* $1+3X_1-X_2<0$.
See below.

b) *On the same plot, sketch the hyperplane* $-2+X_1+2X_2=0$. *Indicate the set of points for which* $-2+X_1+2X_2>0$, *as well as the set of points for which* $-2+X_1+2X_2<0$.



1.2.2 Excercise 2

We have seen that in $p=2$ *dimensions, a linear decision boundary takes the form* $\beta_0+\beta_1X_1+\beta_2X_2=0$. *We now investigate a non-linear decision boundary.*

a) *Sketch the curve*

$$(1+X_1)^2+(2-X_2)^2=4.$$

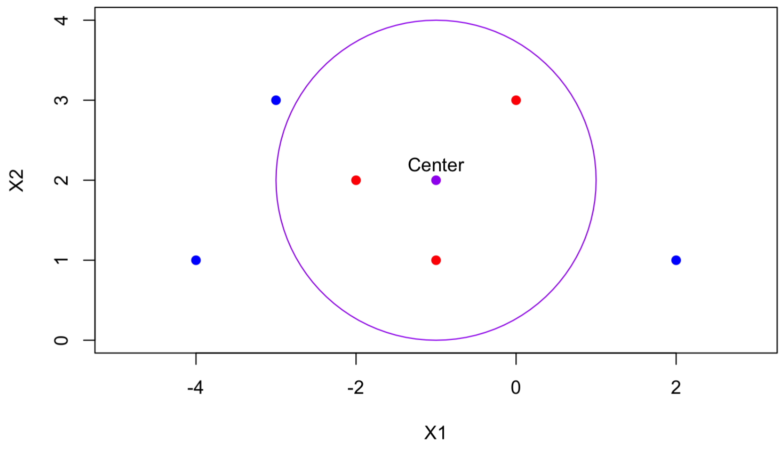
See below.

b) *On your sketch, indicate the set of points for which (BLUE)*

$$(1+X_1)^2+(2-X_2)^2>4,$$

as well as the set of points for which (RED)

$$(1+X_1)^2+(2-X_2)^2<4.$$



c) *Suppose that a classifier assigns an observation to the blue class if*

$$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$

and to the red class otherwise. To what class is the observation $(0, 0)$ *classified?* $(-1, 1)$*?* $(2, 2)$*?* $(3, 8)$*?*
(0,0) : Blue , (-1,1) : Red , (2,2) : Blue , (3,8) : Blue

d) *Argue that while the decision boundary in (c) is not linear in terms of* X_1 *and* X_2*, it is linear in terms of* X_1*,* X_1^2*,* X_2*, and* X_2^2*.*

$$(1 + 2X_1 + X_1^2) + (4 - 4X_2 + X_2^2) = 4$$
$$1 + 2X_1 - 4X_2 + X_1^2 + X_2^2 = 0$$

The resulting equation shows that the circular boundary is linear in the feature space (X_1^2, X_2^2, X_1, X_2). In other words, the 2D equation of a circle when expanded to include X_1^2,X_2^2 becomes a linear equation.

1.2.3 Excercise 3

Here we explore the maximal margin classifier on a toy data set.

a) *We are given* $n = 7$ *observations in* $p = 2$ *dimensions. For each observation, there is an associated class label.*

Obs.	X_1	X_2	Y
1	*3*	*4*	*Red*
2	*2*	*2*	*Red*
3	*4*	*4*	*Red*
4	*1*	*4*	*Red*
5	*2*	*1*	*Blue*
6	*4*	*3*	*Blue*
7	*4*	*1*	*Blue*

Sketch the observations.
See below.

b) *Sketch the optimal separating hyperplane, and provide the equation for this hyperplane (of the form (9.1)).*
Given equation of a line: $y = mx + c$ And given X1, X2, m=1 and c=-0.5, we have: $X_2 - X_1 + 0.5 = 0$

c) *Describe the classification rule for the maximal margin classifier. It should be something along the lines of "Classify to Red if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$, and classify to Blue otherwise." Provide the values for β_0, β_1, and β_2.*
Class Red if $X_2 - X_1 + 0.5 > 0$ and Blue otherwise. $\beta_0 = 0.5; \beta_1 = -1; \beta_2 = 1$

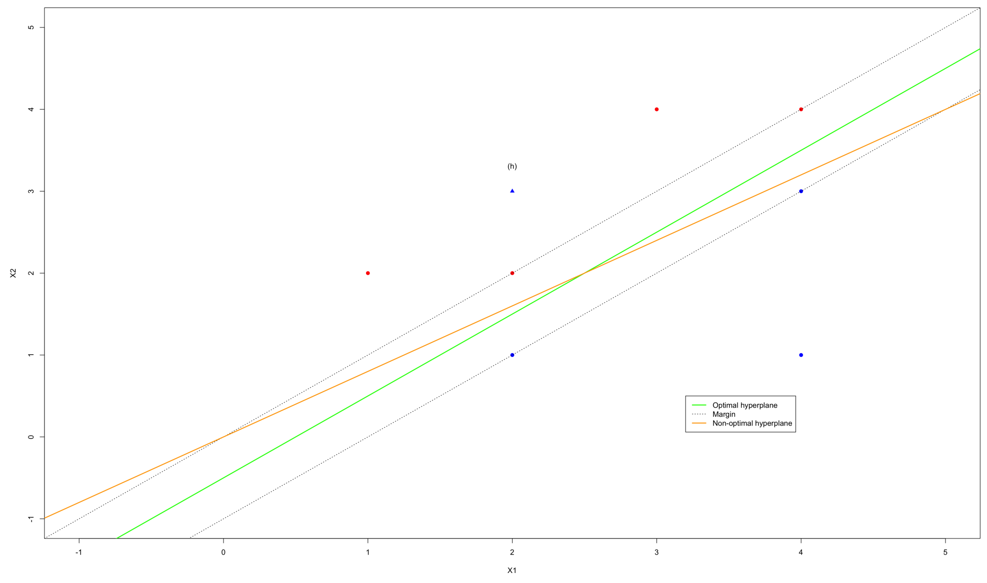
d) *On your sketch, indicate the margin for the maximal margin hyperplane.*
See below.

e) *Indicate the support vectors for the maximal margin classifier.*
See below.

f) *Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.*
Observation seven is not a support vector, and is also located far from the current support vectors. Moving it slightly towards the margin will not make it a support vector, and given that only the support vectors can affect the maximal margin classifier, it holds that observation seven won't affect the classifier.

g) *Sketch a hyperplane that is not the optimal separating hyperplane, and provide the equation for this hyperplane.*
See below.

h) *Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane.*
See below.



2 Practicum Exercises

2.1 Problem 1

Simulate a binary classification dataset with a single feature via a mixture of normal distributions using R (Hint: Generate two data frames with the random number and a class label, and combine them together). The normal distribution parameters (using the function rnorm) should be (5,2) and (-5,2) for the pair of samples - you can determine an appropriate number of samples. Induce a binary decision tree (using rpart), and obtain the threshold value for the feature in the first split. How does this value compare to the empirical distribution of the feature? How many nodes does this tree have? What is the entropy and Gini at each? Repeat with normal distributions of (1,2) and (-1,2). How many nodes does this tree have? Why? Prune this tree (using rpart.prune) with a complexity parameter of 0.1. Describe the resulting tree.
The threshold value for the first split is 0.06. The splitting value should be 0. The tree has 3 nodes.
When using mean 1 and -1, the number of nodes vary a lot. This new tree has 15 nodes and much more complex than the other. We can deduce that when there is not a clear boundary between the data, this algorithm isn't as good. When pruning it, it leaves us a tree with only 3 nodes. The threshold value is now 1.5. The data is divided in data samples of 75% and 25%, very distant from the reality which is 50 50.

```r
n <- 100 # Number of samples per class
data1 <- data.frame(feature = rnorm(n, 5, 2), label = 0)
data2 <- data.frame(feature = rnorm(n, -5, 2), label = 1)
data <- rbind(data1, data2)

library(rpart)
library(rpart.plot)
tree <- rpart(label ~ feature, method="class", data=data)
rpart.plot(tree)

pruned_tree <- prune(tree, cp=0.1)
```

2.2 Problem 2

Load the Wine Quality sample dataset from the UCI Machine Learning Repository (winequality-red.csv and winequality-white.csv) into R using a dataframe. Create an 80/20 test-train split of each wine dataframe, and use the rpart package to induce a decision tree of both the red and white wines, targeting the quality output variable. Visualize the tree using the rpart.plot library, and use the caret package confusionMatrix method to determine the decision tree accuracy on the test set. Compare the decision trees for red and white wine - what differences in terms of tree structure and variables of interest can be noted? Use the randomForest package to repeat the fit with a random forest tree model, and compare the resulting test accuracy against the original single tree model.
Accuracy of red wine: 0.5573.
Accuracy of white wine: 0.5409.
We can see the first variable to split the data is the same on both trees, level of alcohol. Volatile acidity is another common variable. Both trees only classify the new data into 3 levels of quality, which are the most common. These leaves out many possible classifications and leads to error.
In the random forest the accuracy increases, being 0.6972 for the red wine and 0.683 for the white wine.

```
red_wine <- read.csv('winequality-red.csv', sep = ';')
white_wine <- read.csv('winequality-white.csv', sep = ';')
red_wine$quality=as.factor(red_wine$quality)
white_wine$quality=as.factor(white_wine$quality)
library(caret)
set.seed(123)
trainIndexRed <- createDataPartition(red_wine$quality, p = .8, list = FALSE,
    times = 1)
trainRed <- red_wine[trainIndexRed, ]
testRed <- red_wine[-trainIndexRed, ]

trainIndexWhite <- createDataPartition(white_wine$quality, p = .8, list = FALSE,
     times = 1)
trainWhite <- white_wine[trainIndexWhite, ]
testWhite <- white_wine[-trainIndexWhite, ]

library(rpart)
treeRed <- rpart(quality ~ ., data = trainRed, method = "class")
treeWhite <- rpart(quality ~ ., data = trainWhite, method = "class")
library(rpart.plot)
rpart.plot(treeRed)
rpart.plot(treeWhite)

predictionsRed <- predict(treeRed, testRed, type = "class")
predictionsWhite <- predict(treeWhite, testWhite, type = "class")

confusionMatrix(predictionsRed, testRed$quality)
confusionMatrix(predictionsWhite, testWhite$quality)

library(randomForest)
rfRed <- randomForest(quality ~ ., data = trainRed)
rfWhite <- randomForest(quality ~ ., data = trainWhite)

predictionsRFRed <- predict(rfRed, testRed)
predictionsRFWhite <- predict(rfWhite, testWhite)

confusionMatrix(predictionsRFRed, testRed$quality)
confusionMatrix(predictionsRFWhite, testWhite$quality)
```

2.3 Problem 3

Load the SMS Spam Collection sample dataset from the UCI Machine Learning Repository (smsspamcollection.zip) into R using a dataframe (Note: The column names will need to be set manually). Use the tm package to create a Corpus of documents (Hint: Construct the corpus using a VectorSource of the text column). Apply the following transformations from the tm package to the corpus in order prepare the data: a) Convert lowercase, b) Remove stopwords, c) Strip whitespace, and d) Remove punctuation. Use findFreqTerms to contruct features from words occuring more than 10 times and proceed to split the data into a training and test set - for each create a DocumentTermMatrix. Finally convert the DocumentTermMatrix train/test matrices to a Boolean representation (counts greater than zero are converted to a 1) and fit a SVM using the e1071 package. Report your training and test set accuracy.

Test set accuracy: 0.88.

Training set accuracy: 0.86.

```
data <- read.csv("smsspamcollection.txt", sep="\t", header=FALSE)
colnames(data) <- c("label", "text")


pacman::p_load(tm)
```

```r
corpus <- Corpus(VectorSource(data$text))
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, stripWhitespace)
dtm <- DocumentTermMatrix(corpus)
findFreqTerms(dtm, lowfreq = 10)

set.seed(123)  # for reproducibility
index <- sample(seq_len(nrow(dtm)), size = 0.8 * nrow(dtm))
train_dtm <- dtm[index, ]
test_dtm <- dtm[-index, ]

convert_to_boolean <- function(dtm) {
  dtm_bool <- dtm
  dtm_bool$dimnames$Terms <- NULL
  dtm_bool <- as.matrix(dtm_bool)
  dtm_bool[dtm_bool > 0] <- 1
  return(as.data.frame(dtm_bool))
}

train_dtm_bool <- convert_to_boolean(train_dtm)
test_dtm_bool <- convert_to_boolean(test_dtm)


library(e1071)
train_labels <- data$label[index]
test_labels <- data$label[-index]
svm_model <- svm(train_dtm_bool, as.factor(train_labels))
predictions <- predict(svm_model, test_dtm_bool)
accuracy <- sum(predictions == as.factor(test_labels)) / length(test_labels)
print(paste("Test set accuracy:", accuracy))

predictions2 <- predict(svm_model, train_dtm_bool)
accuracy2 <- sum(predictions == as.factor(train_labels)) / length(train_labels)
print(paste("Training set accuracy:", accuracy2))
```