Homework 13

Showroop Pokhrel

12/9/2018

```
library(dplyr, quietly = T)
library(caret, quietly = T)
library(nnet, quietly = T)
mtrain <- read.csv("~/Desktop/HW13/Data/mnist_train.csv", header = T, strings
AsFactors = F)
mtest <- read.csv("~/Desktop/HW13/Data/mnist_test.csv",header = T, stringsAsF
actors = F )
```

DATA PREPROCESING

```
three <- mtrain$X5==3
mtrain$isThree <- factor(as.numeric(three))

#Subsetting to the first 1000 rows
mtrain <- mtrain[1:1000,]

#Variables
response <- "isThree"
predictors <- setdiff(names(mtrain),c(names(mtrain)[1],response))

# Data Preparation
y <- mtrain$isThree
x <- dplyr::select(mtrain, predictors)
names(x) <- NULL
x <- x/255

#Given a dataframe with columns containing atcual and predicted values, the f
ollowing function returns the accuracy of the model
getAccuracy <- function(df)
{
  logical <- df$TrueValue == df$Prediction
  sum <- sum(logical)
  pred_accuracy <- sum/nrow(df)
  return(pred_accuracy)
}
```

MODELING

Part I : Changing only the size of the nodes

```
tuning_df1 <- data.frame(size=1, decay=0)
tuning_df2 <- data.frame(size=2, decay=0)
tuning_df3 <- data.frame(size=3, decay=0)
tuning_df <- data.frame(size=1:20, decay=0)

fitControl1 <- trainControl(method = "none")
fitControl <- trainControl(method = "repeatedcv",number = 2,repeats = 5)

model1 <- caret::train(x=x, y=y, method="nnet",trControl = fitControl1, tuneG
rid=tuning_df1, maxit=1000000000)
model2 <- caret::train(x=x, y=y, method="nnet",trControl = fitControl1, tuneG
rid=tuning_df2, maxit=1000000000)
model3 <- caret::train(x=x, y=y, method="nnet",trControl = fitControl1, tuneG
rid=tuning_df3, maxit=1000000000)
model4 <- caret::train(x=x, y=y, method="nnet",trControl = fitControl, tuneGr
id=tuning_df, maxit=1000000000)

#Predictions
x_test <- mtest[,-1]
names(x_test) <- NULL
prediction1 <- predict(model1,x_test)
prediction2 <- predict(model2,x_test)
prediction3 <- predict(model3, x_test)

fit_df1 <- data.frame(TrueValue=as.numeric(mtest$X7==3), Prediction=predictio
n1)
fit_df2 <- data.frame(TrueValue=as.numeric(mtest$X7==3), Prediction=predictio
n2)
fit_df3 <- data.frame(TrueValue=as.numeric(mtest$X7==3), Prediction=predictio
n3)


accuracy1 <- getAccuracy(fit_df1)
accuracy2 <- getAccuracy(fit_df2)
accuracy3 <- getAccuracy(fit_df3)

df <- data.frame(Nodes=c(1,2,3), Accuracy=c(accuracy1,accuracy2, accuracy3))
knitr::kable(df,caption = "Accuracy in the Validation Dataset")
knitr::kable(model4$results, caption = "Number of nodes and corresponding acc
uracy of model in Training Set")
```

Accuracy in the <u>Validation Dataset</u>

Nodes	Accuracy
1	0.8989899
2	0.8989899
3	0.8989899

Number of nodes and corresponding accuracy of model in <u>Training Set</u>

size	decay	Accuracy	Kappa	AccuracySD	KappaSD
1	0	0.9070007	0	0.0009804	0
2	0	0.9070007	0	0.0009804	0
3	0	0.9070007	0	0.0009804	0
4	0	0.9070007	0	0.0009804	0
5	0	0.9070007	0	0.0009804	0
6	0	0.9070007	0	0.0009804	0
7	0	0.9070007	0	0.0009804	0
8	0	0.9070007	0	0.0009804	0
9	0	0.9070007	0	0.0009804	0
10	0	0.9070007	0	0.0009804	0
11	0	0.9070007	0	0.0009804	0
12	0	0.9070007	0	0.0009804	0
13	0	0.9070007	0	0.0009804	0
14	0	0.9070007	0	0.0009804	0
15	0	0.9070007	0	0.0009804	0
16	0	0.9070007	0	0.0009804	0
17	0	0.9070007	0	0.0009804	0
18	0	0.9070007	0	0.0009804	0
19	0	0.9070007	0	0.0009804	0
20	0	0.9070007	0	0.0009804	0

PART II: Setting the nodes to 1 and changing the decay

```
tuning <- data.frame(size=rep(1,21), decay=seq(0,2,0.1))
fitControl <- trainControl(method = "repeatedcv",number = 2,repeats = 2)
model <- caret::train(x=x, y=y, method="nnet",trControl = fitControl, tuneGri
d=tuning, maxit=1000000000)

prediction <- predict(model,x_test)
fit_df <- data.frame(TrueValue=as.numeric(mtest$X7==3), Prediction=prediction
)
accuracy <- getAccuracy(fit_df)
knitr::kable(model$results, caption = "Varying decay and corresponding accura
cy of model in Training Set")
print(c("The accuracy of the best model (size=1, decay=2) in validation set i
s ", accuracy))
```

Accuracy of model in Training Set

size	decay	Accuracy	Kappa	AccuracySD	KappaSD
1	0.0	0.9070008	0	0.0010528	0
1	0.1	0.9070008	0	0.0010528	0
1	0.2	0.9070008	0	0.0010528	0
1	0.3	0.9070008	0	0.0010528	0
1	0.4	0.9070008	0	0.0010528	0
1	0.5	0.9070008	0	0.0010528	0
1	0.6	0.9070008	0	0.0010528	0
1	0.7	0.9070008	0	0.0010528	0
1	0.8	0.9070008	0	0.0010528	0
1	0.9	0.9070008	0	0.0010528	0
1	1.0	0.9070008	0	0.0010528	0
1	1.1	0.9070008	0	0.0010528	0
1	1.2	0.9070008	0	0.0010528	0
1	1.3	0.9070008	0	0.0010528	0
1	1.4	0.9070008	0	0.0010528	0
1	1.5	0.9070008	0	0.0010528	0
1	1.6	0.9070008	0	0.0010528	0
1	1.7	0.9070008	0	0.0010528	0
1	1.8	0.9070008	0	0.0010528	0

1	1.9	0.9070008	0	0.0010528	0
1	2.0	0.9070008	0	0.0010528	0

```
## [1] "The accuracy of the best model (size=1, decay=2) in validation set is
## [2] "0.898989898989899"
```

Observations:

1) No change in predictive accuracy in as a result of node increment. Maybe the effect of increasing nodes is more pronounced in a larger sample size and not as much when n=1000
2) Models with nodes = 1, 2 and 3 were tested in the validation dataset and all yielded an accuracy of 89.9%. Therefore, model with node=1 is the most parsimonious.
3) Next, I set node = 1 and vary the decay from 0-2 in crements of 0.1. All the models have the same accuracy in the training set. (Same results when both the decay and nodes are varied. But not shown here).
4) R autoselects node=1 and decay=2 as the best, but the model has the same predictive accuracy in the validation set as PART I (node=1, decay =0)
5) Given all of this information I think the optimal model in this context would be either (node=1, decay=0) or (node=1, decay=2), whichever is more parsimonious.