R-Shiny Application: CIFAR-10 Classification

Jose Gonzalez

April 27th, 2016

1.0 GENERAL INFORMATION

1.1 Purpose

CIFAR-10 is an established computer-vision dataset used for object recognition. It is a subset of the 80 million tiny images dataset and consists of 60,000 32x32 color images containing one of 10 object classes, with 6000 images per class. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The purpose of this document is to discuss the application of different machine learning approaches including principal component analysis, linear discriminant analysis, k-nearest-neighbor and random forest. We will also be touching on how an R-shiny application is able to display the results from the image classification.

2.0 IMAGE CLASSIFICATION

2.1 Packages

In order to complete this project, it was important to choose a language and its respective data mining packages. I used the Python package pandas (a data analysis library) to read in the data. I decided to use scikit-learn, a Python package that provides simple and efficient data mining, machine learning and data analysis tools for the image classification. I also used the Numpy package, which is a fundamental package for scientific computing.

2.2 Data Preparation

To start, it was important to have the data formatted in a form that would be easier to work with and maintain. The images were pre-processed into their respective pixel data (rgb values) and then properly labeled in an excel document. The next step was to read in the data and break it into two pieces, the pixel data and the labels that identified the images. After that, those two pieces were broken into two additional parts, a training dataset and a testing dataset for each.

```
import pandas as pd
from sklearn.cross_validation import train_test_split

data = pd.read_csv('train1p_with_header.csv')
X=data.iloc[:,0:data.shape[1]-1].as_matrix()
y=data.iloc[:,data.shape[1]-1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

2.3 Classification

2.3.1 PCA to Random Forest Classifier

The first classifier I completed was a PCA (Principle Component Analysis) to Random Forest Classifier pipe. This approach resulted in a classifier accuracy of 32%.

```python
import numpy as np
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier

def pca_rfc(X_train,X_test,y_train,y_test):
    pca = PCA(n_components=500)
    pca.fit(X_train)
    scores = np.dot(X_train,np.transpose(pca.components_))
    rfc = RandomForestClassifier(n_estimators=500, oob_score=True, random_state = 60)
    rfc.fit(scores,y_train)
    return rfc.oob_score_

pca_rfc(X_train,X_test,y_train,y_test)
```

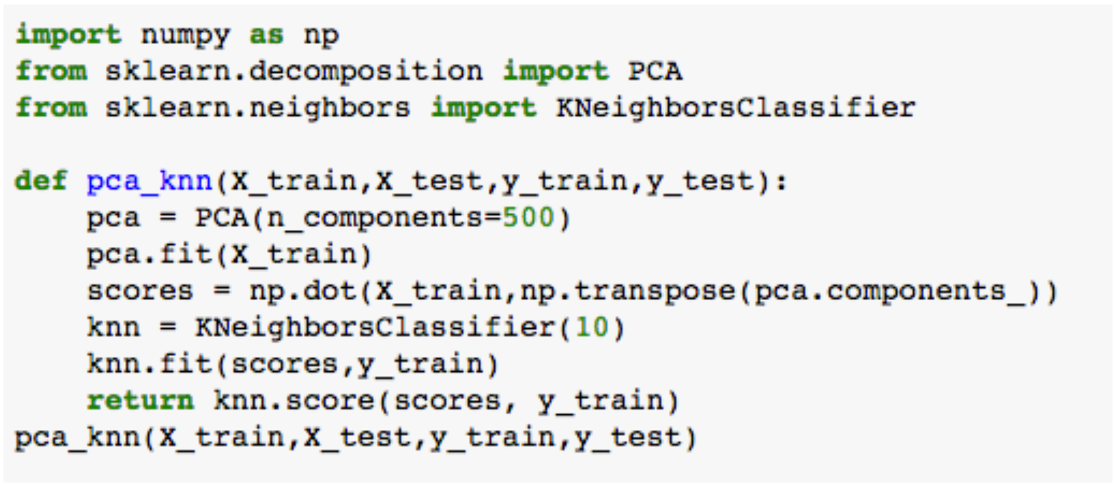
2.3.2 PCA to LDA

The second classifier I completed was a PCA (Principle Component Analysis) to LDA (Linear Discriminant Analysis) pipe. This approach resulted in a classifier accuracy of 58%.

```python
import numpy as np
from sklearn.decomposition import PCA
from sklearn.lda import LDA

def pca_lda(X_train,X_test,y_train,y_test):
    pca = PCA(n_components=500)
    lda = LDA()
    pca.fit(X_train)
    scores = np.dot(X_train,np.transpose(pca.components_))
    lda.fit(scores, y_train)
    return lda.score(scores, y_train, sample_weight=None)

pca_lda(X_train,X_test,y_train,y_test)
```

2.3.3 PCA to KNN

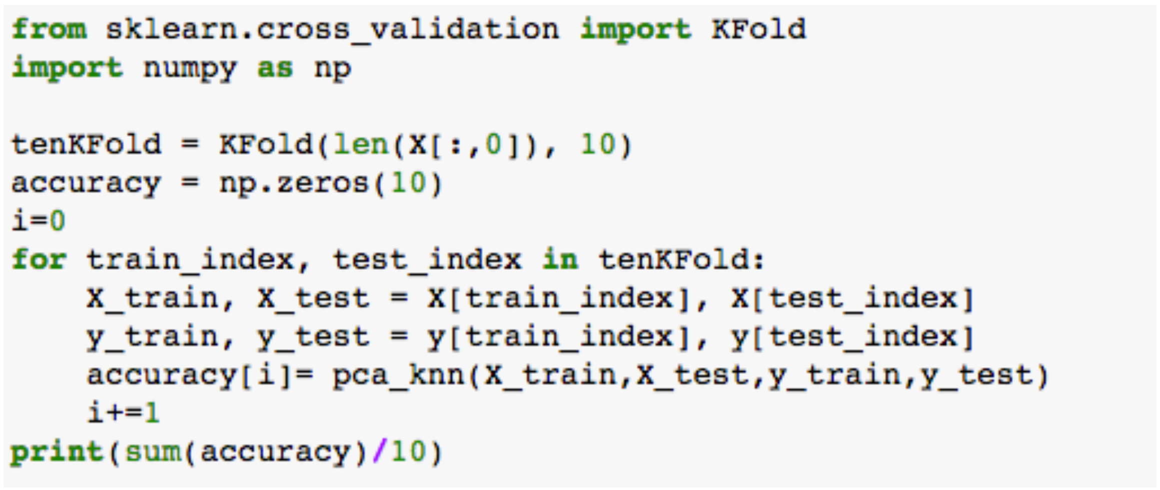
The third classifier I completed was a PCA (Principle Component Analysis) to KNN (K-Nearest-Neighbor) pipe. This approach resulted in a classifier accuracy of 38%.



```python
import numpy as np
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier

def pca_knn(X_train,X_test,y_train,y_test):
    pca = PCA(n_components=500)
    pca.fit(X_train)
    scores = np.dot(X_train,np.transpose(pca.components_))
    knn = KNeighborsClassifier(10)
    knn.fit(scores,y_train)
    return knn.score(scores, y_train)
pca_knn(X_train,X_test,y_train,y_test)
```

2.4 10-Fold Cross-Validation

Lastly, I completed a 10-fold cross-validation that can be used for any of the classifiers. In the example below, I am running a 10-fold cross-validation on the PCA to K Nearest Neighbor algorithm. This cross-validation resulted in an accuracy of



```python
from sklearn.cross_validation import KFold
import numpy as np

tenKFold = KFold(len(X[:,0]), 10)
accuracy = np.zeros(10)
i=0
for train_index, test_index in tenKFold:
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    accuracy[i]= pca_knn(X_train,X_test,y_train,y_test)
    i+=1
print(sum(accuracy)/10)
```

3.0 RSHINY APPLICATION

3.1 Layout

For the RShiny application, I decided to create an interactive page that lets the user choose what charts they would like to see. The user has the choice to view the random forest classification charts, linear discriminant analysis charts and the k-nearest neighbor charts. Each classification has two charts, a boxplot containing the data from the 10-fold cross validation and a data plot containing each cross validation accuracy vs. its cross validation iteration. The application also shows the user the five number summaries for the boxplots. The charts are created from a csv file called cvData containing the cross validation data.

3.2 Code

3.2.1 User Interface (ui.r)

```r
library(shiny)
shinyUI(fluidPage(

  titlePanel("Image Classification"),

  sidebarLayout(
    sidebarPanel(
      selectInput('class', 'Classifier', list("PCA -> Random Forest Classification" = "RFC",
                                              "PCA -> Linear Discriminant Analysis" = "LDA",
                                              "PCA -> K-Nearest Neighbor" = "KNN"))
    ),

    mainPanel(
      tabsetPanel(
        tabPanel("Box Plot",
                 plotOutput("plot1"),
                 h4("Five Number Summary: MIN, Q1, Q2, Q3, MAX"),
                 textOutput("getStats")
        ),
        tabPanel("Dot Plot",
                 plotOutput("plot2")
        ),
        tabPanel("Box Plot Comparison",
                 plotOutput("plot3")
        )
      )

    )
  )
))
```

3.2.2 Server (server.r)

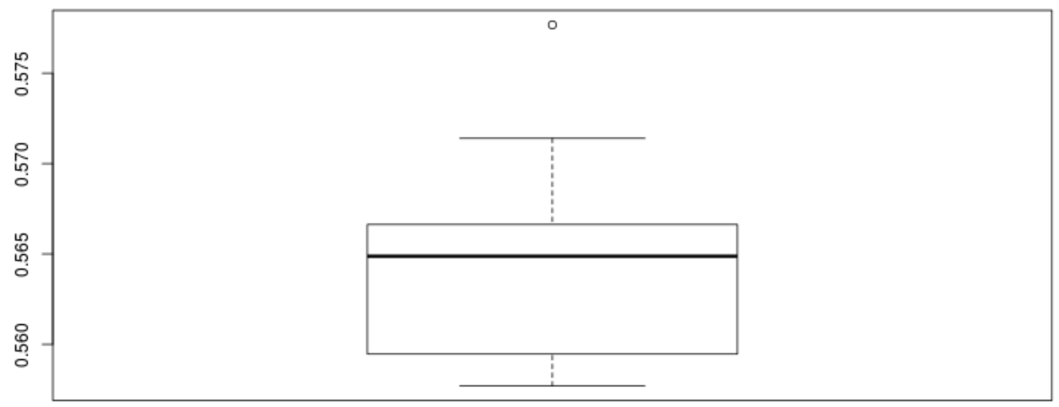
```r
library(shiny)
cvData <- read.csv(file = "/home/spring2016_csci334user13/img_app/cvData.csv")

shinyServer(function(input, output) {

  output$plot1 <- renderPlot({
    name = input$class

    if (name=="RFC") {
      boxplot(cvData$rfc)
    } else if (name=="LDA") {
      boxplot(cvData$lda)
    } else {
      boxplot(cvData$knn)
    }

  })

  output$plot2 <- renderPlot({
    name = input$class

    if (name=="RFC") {
      y <- cvData[,1]
      name = "PCA -> Random Forest Classification"
    } else if (name=="LDA") {
      y <- cvData[,2]
      name = "PCA -> Linear Discriminant Analysis"
    } else {
      y <- cvData[,3]
      name = "PCA -> K-Nearest Neighbor"
    }
    x <- 1:10
    plot(x,y,main=paste("10-Fold Cross Validation for", name),xlab="Cross-Validation Iteration",
         ylab="Accuracy",ylim=c(0,1),col = 'black')
  })

  output$plot3 <- renderPlot({
    boxplot(cvData, main="Accuracy by Classification")
  })

  output$getStats <- renderText({

    name = input$class

    if (name=="RFC") {
      rfcPlot <- boxplot(cvData$rfc)
      paste(rfcPlot$stats)

    } else if (name=="LDA") {
      ldaPlot <- boxplot(cvData$lda)
      paste(ldaPlot$stats)
    } else {
      knnPlot <- boxplot(cvData$knn)
      paste(knnPlot$stats)
    }

  })

})
```

3.2.3 Data (cvData.csv)

pca_rfc	pca_lda	pca_knn
0.326923077	0.566312997	0.374668435
0.309460654	0.564102564	0.369363395
0.324270557	0.571396994	0.361405836
0.32250221	0.559460654	0.377984085
0.315428824	0.557692308	0.363616269
0.323386384	0.561892131	0.372458002
0.317639257	0.565649867	0.366931919
0.326187845	0.577679558	0.373259669
0.316243094	0.566629834	0.372596685
0.310276243	0.558453039	0.381657459

3.3 Example Charts

3.3.1 Single Classification Boxplot



Five Number Summary: MIN, Q1, Q2, Q3, MAX

0.557692308 0.559460654 0.5648762155 0.566629834 0.571396994

Figure 1. A boxplot for the PCA->LDA classification's cross validation data.

3.3.2 Single Classification Data Plot

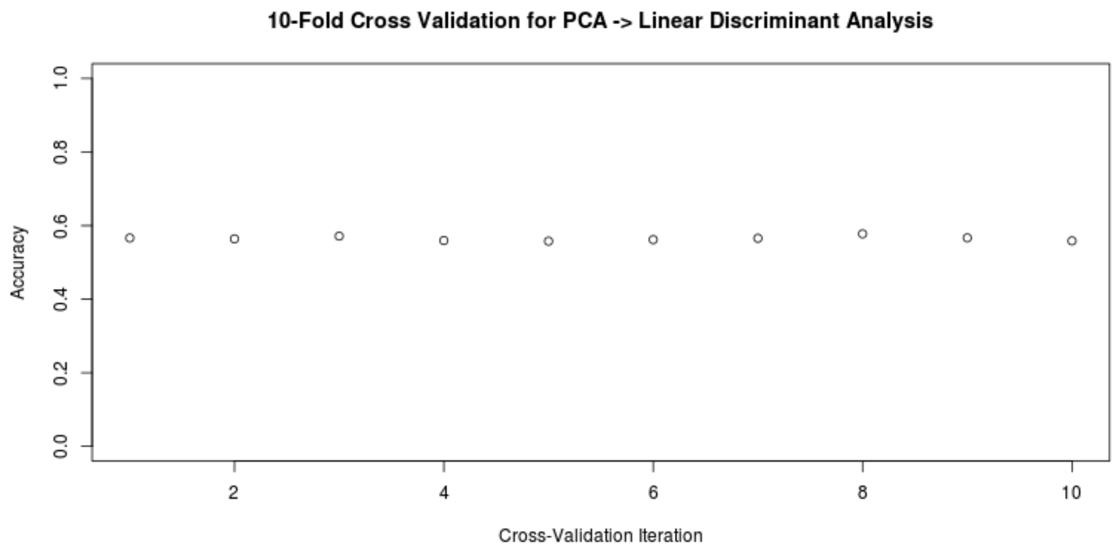


Figure 2. A data plot for the PCA->LDA classification's cross validation data.

3.3.3 Multiple Classification Boxplot

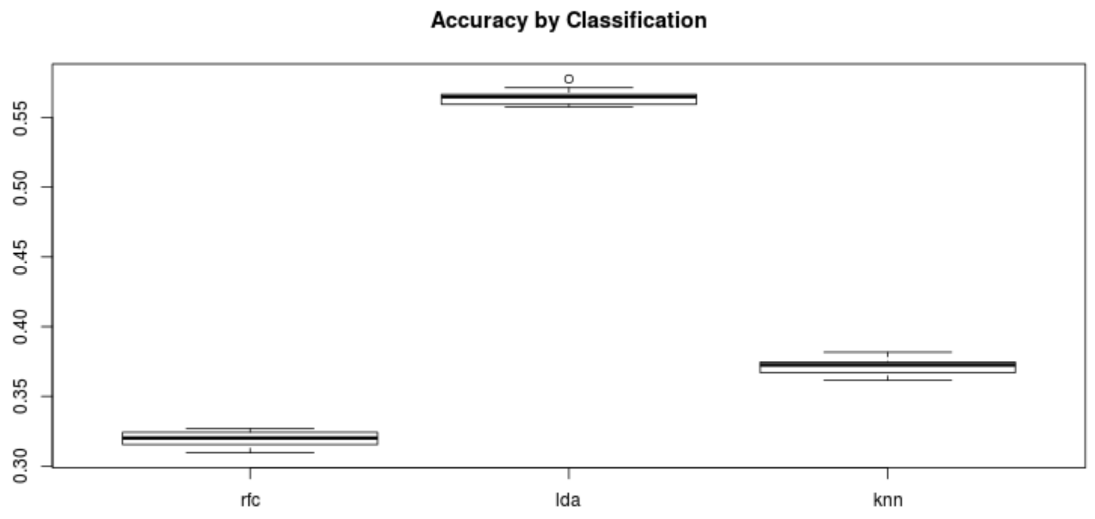


Figure 3. A boxplot for all three forms of classification.

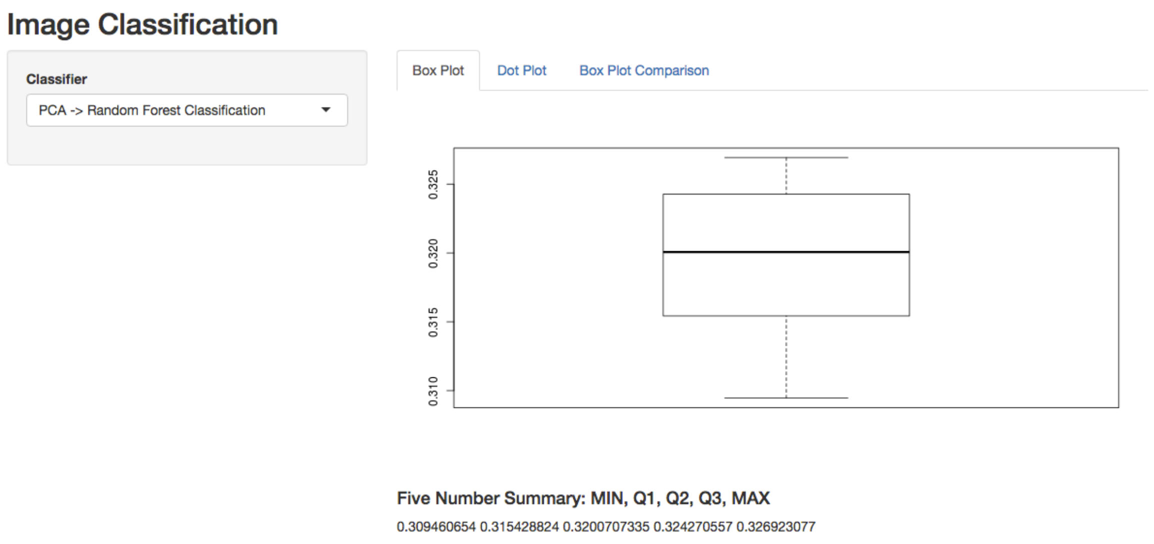
3.4 Instructions

To successfully use the RShiny application locally, you need to make sure that you have the Rshiny library installed, along with the basic R-studio packages. Alternatively, you could go to the freyja server's R-studio and start the application from there. There are a few lines of code to do this:

1. setwd("path/to/application/files")
2. library(shiny)
3. runApp("path/to/application/files", port=##your port number##, host="0.0.0.0")

After this, you can access the application by visiting:

https://freyja.cs.cofc.edu/##your username##/



Five Number Summary: MIN, Q1, Q2, Q3, MAX

0.309460654 0.315428824 0.3200707335 0.324270557 0.326923077

Figure 4. The Image Classification RShiny application.