York University

Image Classification Using SVM &Random Forest Classification

Ravish Kamath

213893664

MATH 3333

Professor Xin Gao

Date: April 22, 2022

Abstract

In this paper, we will discuss two different methods of classification for a whole dataset of images that came from the Columbia Photographic Images and Photorealistic Computer Graphic Dataset. Those two methods would be Support Vector Machine (SVM) and Random Forest classification. In this paper we will give a description and the process of each classification method. Furthermore, we will be comparing its effectiveness with the Logistic Regression method, by using the Receiver Operating Characteristic (ROC) function to see which method of classification works best for the dataset of images. When comparing with the results with the Logistic Regression, we see that both methods do better, however the random forest classification seems to be able to do a better job at classification than the other two. Finally, we will do some cross validation as well to compares the different methods and conclude.

Introduction

With the world going into a more digital age, the advent of big data has become a lucrative commodity for many businesses as well as governments. One part of big data is image processing and classification. We can use image classification for urban planning, natural disaster identification, criminal identification, self-driving cars, categorizing pictures in our phones etc. As we can see from the list, image classification has a wide array of uses. However, image classification can only be done through computers and algorithms, due to the billions of gigabytes that millions of pictures contain. To overcome this, statisticians, data analyst and data scientists use various machine learning algorithms to classify pictures into different categories, so that people can use the data to identify quickly any other data. Some machine learning algorithms that these individuals use are neural networks, random forest, support vector machine (SVM), linear discriminant analysis (LDA) etc. Our purpose in this report is try to use SVM and random forest method to see if they are good algorithms to classify the pictures compared to the Logistic Regression method.

Methodology

Data

The data set that we are using was provided by Professor Gao, which is called Columbia Photographic Images and Photorealistic Computer Graphic Dataset. In this data set we can see there are different categories for each JPEG file. Those categories are:

- Artificial
- Indoor-light
- Outdoor-dan-dusk
- Outdoor-night
- Indoor-dark
- Natural
- Outdoor-day
- Outdoor-rain-snow

Furthermore in our dataset, we have the location of each photo, which are from New York or Boston. All the code for the dataset is provided in Appendix 1.1. Based on the figure 1 below we can see that the data classification can be

tough to identify, hence why we would need to use methods such as SVM and random forest, to better classify the images.

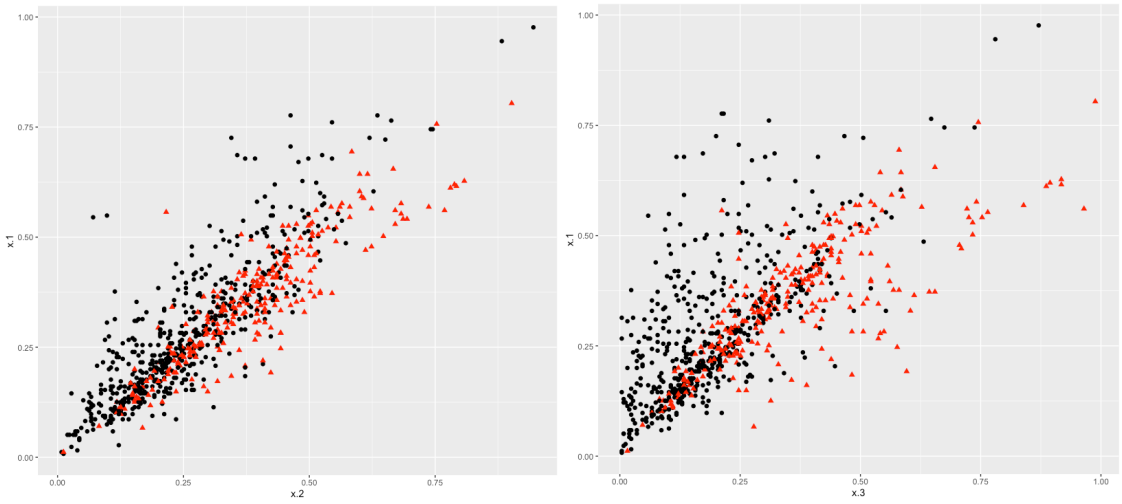

Figure 1: (Appendix 1.2)

Logistic Regression

Logistic regression builds of linear regression except the response variable(what we are interested in) is binary (0 or 1). Hence the response variable can either be true or false. The use of Logistic is helpful when it comes to problems such as profit or loss, loans that have been defaulted or not etc. Here is the formula for a logistic model:

$$\log\left(\frac{p}{1-p}\right) = a + \sum b_i x_i$$

Although the logistic model Is a great machine learning tactic, it may not be helpful for classification for the images we are using in the dataset. The code and

output for the logistic Regression is provided in Appendix 2.1. We will discuss this further in the analysis and result sections.

SVM

SVM is a special classification algorithm where it tries to find a good separation line and adds 2 parallel lines of equal distance to that line so that each parallel line passes through the data point that is closest to the line of separation. The distance between the 2 parallel lines is how we measure how good the separation is. If the distance is large, the better the classifier. Sometimes the separation line could be linear but also nonlinear. In the case for nonlinear, SVM method would transform the data into higher dimensions (through projections) where the separation is possible. We do this by using the kernel function. The purpose of the kernel function is to calculate the distance between 2 points in the new projected higher dimension space, using a new dot product. We will discuss the effectiveness of this method in the analysis and result sections.

Random Forest

Random Forest is a method that comes from decision trees. A decision tree is a model which maps from the space of the predictors to the space of response, usually done through binary. Random Forests is one of the 3 ensemble methods

that many data analysts use. “At each node, an individual decision tree determines the split on the basis of a smaller, random selection of attributes, and not from the set of all attributes. Each tree in the forest of trees then votes on the classification on a new item, and the most popular class is returned as the ensemble solution.” (Ledolter). We will discuss the analysis and results of this method in the next couple sections.

Analysis

When conducting the logistic regression function, the output we get is listed in on Appendix 2.1 Here we can see that the intercept, X1 and X3 are significant since their p-value is less than 5%. Furthermore, to verify whether this method is effective, we applied the method to an ROC. Figure 2 indicates the effectiveness of it.

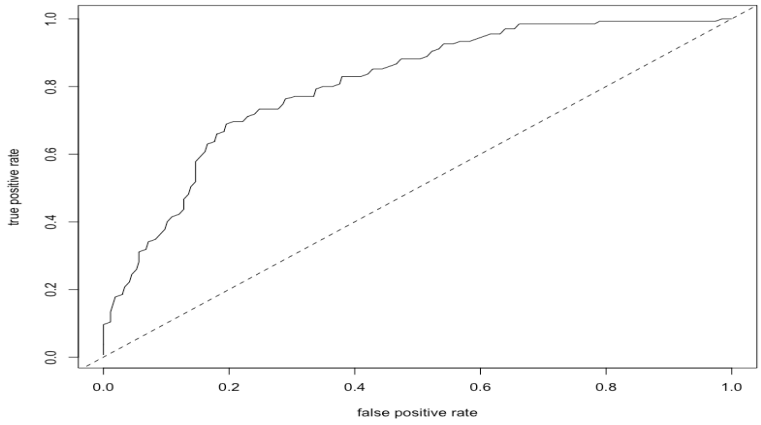

Figure 2. Logistic ROC (Appendix 2.2)

Next, we conducted the SVM method. We tried to use different types of kernels to see how effective by seeing the AUC values. This unfortunately did not change the AUC value at all. The code for the SVM is indicated in Appendix 3.1. Furthermore, we have Figure 3 that shows visually how the result is when we apply the SVM for our data set.

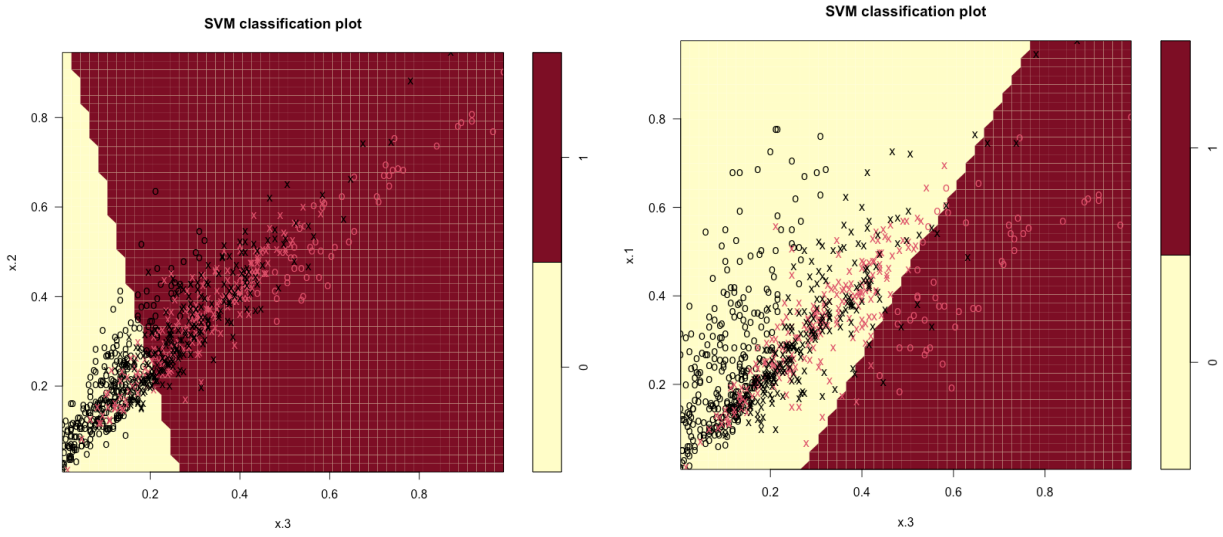

Figure 3. SVM plots (Appendix 3.1)

Finally for our final method, random forest, the code is provided in Appendix 4.1. Here we just used the basic random forest method and was able to come up with a visual of the decision tree, as seen in Figure 4.

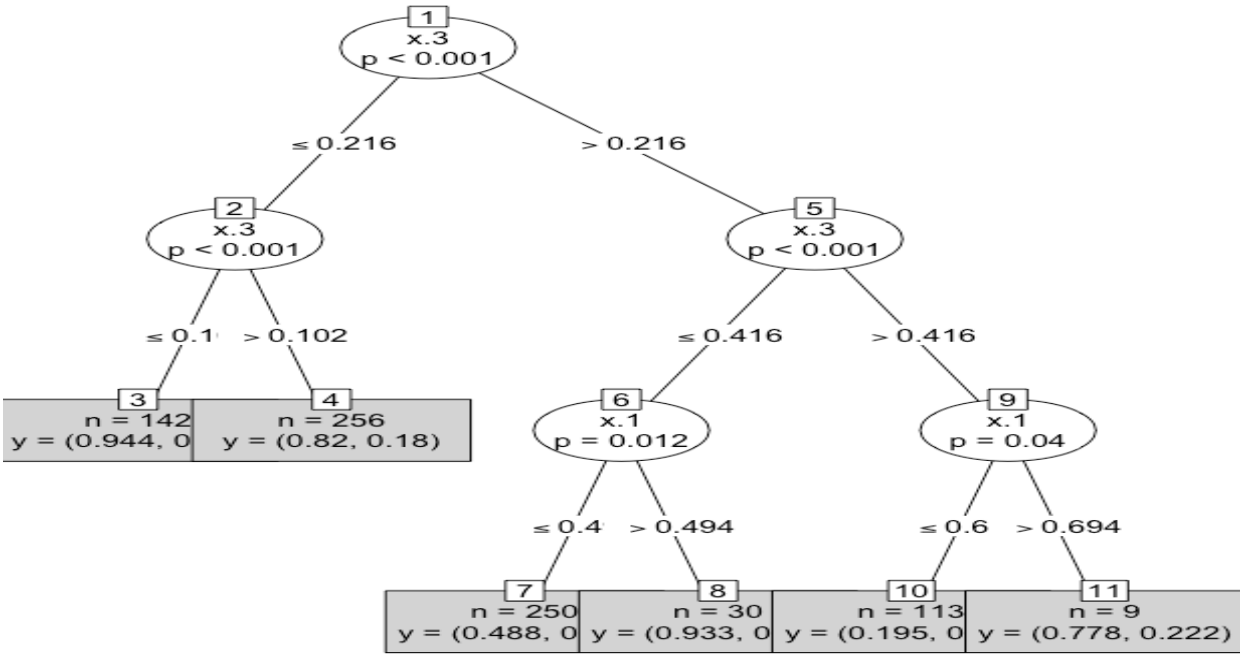

Figure 4 (Appendix 4.1)

Results

Recall that for each method, we will be measuring its effectiveness using the AOC and area of under the curve. For the logistic regression, we have the AUC to be around 80%. For the SVM method we had the AUC to be slightly better at an 81%. Finally for the random forest, we do not have any results for the AUC, however we can use the cross validations too see how successful it is. We will be comparing it to the SVM model. In the case for SVM, we get cross validation mean error to be 0.2435 while for the Random Forest, we got it to be 0.24276. The SVM cross validation code is shown in Appendix 3.3 while for the Random Forest, it is

in Appendix 4.2. It seems like the random forest and the SVM have very similar results for the cross validation, although we only ran it 10 times due to processing issues occurring. Based on the results it seems like both these methods are more successful than the logistic regression, however by not a big margin as expected. One recommendation would be to reduce the images into smaller partitions in order to get more data for these algorithms to work. However due to its complexity, we avoided that tactic, and just focused on the raw dataset that was given.

Conclusion

To conclude, using the SVM and Random Forest gave us slightly better results compared to the Logistic Regression analysis. We used ROC and cross validation as measurements for their effectiveness, with both methods showing very similar results. A small note that was discussed in the analysis was the use of partitioning the images, to get better results, however due to the complexity, this was avoided. These methods are important for image classification for different applications such as photo identification or self-driving cars. As the world get more digital, the use of image classification will be more important than ever before.

Appendix

Appendix 1

1.1 Dataset

```
pm = read.csv('/Users/ravishkamath/Desktop/University/2.
        York Math/1 MATH/1. Statistics /MATH 3333/3.
        Assessments/Final Project/Supporting files for
        final project-20220307/photoMetaData.csv')
n <- nrow(pm)
trainFlag <- (runif(n) > 0.5)
y <- as.numeric(pm$category == "outdoor-day")

X <- matrix(NA, ncol=3, nrow=n)
for (j in 1:n) {
  img <- readJPEG(paste0("/Users/ravishkamath/Desktop/University/2.
                York Math/1 MATH/1. Statistics /MATH 3333/3.
                Assessments/Final Project/Supporting files
                for final project-20220307/columbiaImages//",pm$name[j]))
  X[j,] <- apply(img,3,median)
  print(sprintf("%03d / %03d", j, n))
}
dat = data.frame(x=X, y=as.factor(y))
```

1.2 Dataset Plots

```
ggplot(data = dat, aes(x = x.2 , y = x.1, color = y, shape = y)) +
  geom_point(size = 2) +
  scale_color_manual(values=c("#000000", "#FF0000")) +
  theme(legend.position = "none")

ggplot(data = dat, aes(x = x.3 , y = x.1, color = y, shape = y)) +
  geom_point(size = 2) +
  scale_color_manual(values=c("#000000", "#FF0000")) +
  theme(legend.position = "none")
```

Appendix 2

2.1 Logistic Regression

```
out <- glm(y ~ X, family=binomial, subset=trainFlag)
out$iter
summary(out)
```

```
Call:
glm(formula = y ~ X, family = binomial, subset = trainFlag)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.3252  -0.7762  -0.4713   0.8911   2.0175

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.0141     0.3185  -6.323 2.57e-10 ***
X1           -8.6245     2.2607  -3.815 0.000136 ***
X2            3.4050     3.3609   1.013 0.311004
X3           11.4934     2.4563   4.679 2.88e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 519.51  on 398  degrees of freedom
Residual deviance: 393.38  on 395  degrees of freedom
AIC: 401.38

Number of Fisher Scoring iterations: 5
```

2.2 Logistic ROC/AUC

```
pred <- 1 / (1 + exp(-1 * cbind(1,X) %*% coef(out)))
y[order(pred)]
y[!trainFlag][order(pred[!trainFlag])]

mean((as.numeric(pred > 0.5) == y)[trainFlag])
mean((as.numeric(pred > 0.5) == y)[!trainFlag])

roc <- function(y, pred) {
  alpha <- quantile(pred, seq(0,1,by=0.01))
  N <- length(alpha)

  sens <- rep(NA,N)
  spec <- rep(NA,N)
  for (i in 1:N) {
    predClass <- as.numeric(pred >= alpha[i])
    sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
    spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
```

```
 }
 return(list(fpr=1- spec, tpr=sens))
}

r <- roc(y[!trainFlag], pred[!trainFlag])
plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
abline(0,1,lty="dashed")

#AUC#
auc <- function(r) {
 sum((r$fpr) * diff(c(0,r$tpr)))
}
glmAuc <- auc(r)
glmAuc
```

```
> mean((as.numeric(pred > 0.5) == y)[trainFlag])
[1] 0.7443609
> mean((as.numeric(pred > 0.5) == y)[!trainFlag])
[1] 0.7406484
>
> roc <- function(y, pred) {
+   alpha <- quantile(pred, seq(0,1,by=0.01))
+   N <- length(alpha)
+
+   sens <- rep(NA,N)
+   spec <- rep(NA,N)
+   for (i in 1:N) {
+     predClass <- as.numeric(pred >= alpha[i])
+     sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+     spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+   }
+   return(list(fpr=1- spec, tpr=sens))
+ }
>
> r <- roc(y[!trainFlag], pred[!trainFlag])
> plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
> abline(0,1,lty="dashed")
>
> #AUC#
> auc <- function(r) {
+   sum((r$fpr) * diff(c(0,r$tpr)))
+ }
> glmAuc <- auc(r)
> glmAuc
[1] 0.8069897
```

Appendix 3

3.1 SVM

```
y = factor(y)
svm_model1 = svm(y~., type = 'C',  kernel = 'linear', data = dat)
summary(svm_model1)
plot(svm_model1, dat, x.2~x.3)
plot(svm_model1, dat, x.1~x.3)
```

```
Call:
svm(formula = y ~ ., data = dat, type = "C", kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  455

 ( 227 228 )


Number of Classes:  2

Levels:
 0 1
```

3.2 SVM ROC/AUC

```
#ROC#
pred <- 1 / (1 + exp(-1 * cbind(1,X) %*% coef(svm_model1)))
y[order(pred)]
y[!trainFlag][order(pred[!trainFlag])]

mean((as.numeric(pred > 0.5) == y)[trainFlag])
mean((as.numeric(pred > 0.5) == y)[!trainFlag])

roc <- function(y, pred) {
  alpha <- quantile(pred, seq(0,1,by=0.01))
  N <- length(alpha)

  sens <- rep(NA,N)
  spec <- rep(NA,N)
  for (i in 1:N) {
    predClass <- as.numeric(pred >= alpha[i])
    sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
    spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
  }
  return(list(fpr=1- spec, tpr=sens))
}

r <- roc(y[!trainFlag], pred[!trainFlag])
plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
abline(0,1,lty="dashed")
```

```
# auc
auc <- function(r) {
  sum((r$fpr) * diff(c(0,r$tpr)))
}
glmAuc <- auc(r)
glmAuc
```

3.3 SVM Cross Validation

```
# Cross Valdiation #
n = nrow(X)
nt = 600
rep = 10
error_SVM = dim(rep)
neval = n - nt
for (i in 1: rep) {
  training = sample(1:n, nt)
  trainingset = dat[training,]
  testingset = dat[-training,]

  #Random Forest Analysis
  rf_classifier = randomForest(y ~., data = trainingset, type = classification,
                               ntree = 100, mtry = 2, importance = TRUE)
  prediction_RF = predict(rf_classifier, testingset)
  table_RF = table(yPrime, prediction_RF)
  error_RF[i] = (neval - sum(diag(table_RF)))/neval
}
mean(error_RF)
```

```
[1] 0.2435
```

Appendix 4

4.1 Random Forest Method

```
rf_classifier = randomForest(y ~., data = dat, type = classification,
                             ntree = 100, mtry = 2, importance = TRUE)
rf_classifier

m <- ctree(y ~ ., data=dat)
m
plot(m, type="simple")
table(predict(m), dat$y)
Call:
 randomForest(formula = y ~ ., data = dat, type = classification,      ntree = 100, mtry = 2, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 2

        OOB estimate of  error rate: 30.63%
Confusion matrix:
    0   1 class.error
0 419 104   0.1988528
1 141 136   0.5090253
>
> m <- ctree(y ~ ., data=dat)
> m

               Conditional inference tree with 6 terminal nodes

Response:  y
Inputs:  x.1, x.2, x.3
Number of observations:  800

1) x.3 <= 0.2156863; criterion = 1, statistic = 181.121
  2) x.3 <= 0.1019608; criterion = 1, statistic = 15.355
    3)*  weights = 142
  2) x.3 > 0.1019608
    4)*  weights = 256
1) x.3 > 0.2156863
  5) x.3 <= 0.4156863; criterion = 1, statistic = 27.244
    6) x.1 <= 0.4941176; criterion = 0.988, statistic = 8.232
      7)*  weights = 250
    6) x.1 > 0.4941176
      8)*  weights = 30
  5) x.3 > 0.4156863
    9) x.1 <= 0.6941176; criterion = 0.96, statistic = 6.093
      10)*  weights = 113
    9) x.1 > 0.6941176
      11)*  weights = 9
> plot(m, type="simple")
> table(predict(m), dat$y)

     0   1
 0 379  58
 1 144 219
```

4.2 Random Forest Cross Validation

```
# Cross Valdiation #
n = nrow(X)
nt = 600
rep = 10
error_SVM = dim(rep)
neval = n - nt
for (i in 1: rep) {
  training = sample(1:n, nt)
  trainingset = dat[training,]
  testingset = dat[-training,]

  #Random Forest Analysis
  rf_classifier = randomForest(y ~., data = trainingset, type = classification,
                               ntree = 100, mtry = 2, importance = TRUE)
  prediction_RF = predict(rf_classifier, testingset)
  table_RF = table(y, prediction_RF)
  error_RF[i] = (neval - sum(diag(table_RF)))/neval
}
mean(error_RF)
```

```
[1] 0.24276
```

References

1. *Support Vector Machine*. Support Vector Machine · UC Business Analytics R Programming Guide. (n.d.). Retrieved April 23, 2022, from http://uc-r.github.io/svm

2. data_hacks, Harshita_Dudhe, & dilshad4816129. (2015, August 7). *How to plot a sample tree from random forest in R*. Data Science, Analytics and Big Data discussions. Retrieved April 23, 2022, from https://discuss.analyticsvidhya.com/t/how-to-plot-a-sample-tree-from-random-forest-in-r/2800

3. Ledolter, J. (2013). *Data Mining and business analytics with R*. New Jersey.

4. *Image classification*. Image Classification - an overview | ScienceDirect Topics. (n.d.). Retrieved April 23, 2022, from https://www.sciencedirect.com/topics/engineering/image-classification