Project: Dogs, Fried Chicken or Blueberry Muffins?

Team #4

Summary:

In this project, we created a classifier for images of puppies, fried chickens and blueberry muffins.

Install Packages

```
# packages.used=c("gbm", "caret","DMwR" ,"nnet","randomForest","EBImage","e1071","xgboost")
#
# # check packages that need to be installed.
# packages.needed=setdiff(packages.used,
#                         intersect(installed.packages()[,1],
#                                   packages.used))
# # install additional packages
# if(length(packages.needed)>0){
#   install.packages(packages.needed, dependencies = TRUE)
# }
```

Read in SIFT feature data

```
sift_train0 <- read.csv("./data/sift_train.csv", header=F)
label_train0 <- read.csv("./data/label_train.csv", header=F)
source("./lib/eco2121_train_gbm_baseline.r")
source("./lib/pca_features.r")
#source("./lib/new_xgboost_sift_pca100.r")

sift <- sift_train0[, -1]
```

Use PCA to reduce dimension

```
set.seed(500)
# data <- pca_features(sift, 100)
#
# # selected data with labels
# pca_train_data <- cbind(data, label_train0[,2])
# colnames(pca_train_data)[ncol(pca_train_data)] <- "label"
# pca_train_data<-as.data.frame(pca_train_data)

sift_pca<-read.csv("./data/feature_pca100.csv",header = T, as.is = T)
label<-read.csv("./data/label_train.csv",header = T,as.is = T)
dat<-cbind(label[,2],sift_pca[,-1])
colnames(dat)[1]<-"label"
```

Train and Validate set

```r
set.seed(500)
# Train and test split
train_index<-sample(1:nrow(dat),0.7*nrow(dat))

xgb_variables<-as.matrix(dat[,-1]) # Full dataset
xgb_label<-dat[,1] # Full label

# Split train data
xgb_train<-xgb_variables[train_index,]
train_label<-xgb_label[train_index]
train_matrix<-xgb.DMatrix(data = xgb_train, label=train_label)

# Split test data
xgb_test<-xgb_variables[-train_index,]
test_label<-xgb_label[-train_index]
test_matrix<-xgb.DMatrix(data = xgb_test, label=test_label)
```

Baseline Model: GBM + SIFT

```r
sift_train = read.csv("./data/sift_train.csv")
label = read.csv("./data/label_train.csv")
data = data.frame(label[,2], sift_train[,2:ncol(sift_train)])
colnames(data)[1] = "label"

set.seed(123)
index = sample(1:nrow(data), size=0.7*nrow(data))
train_data = data[index,]
test_data = data[-index,]

## To run the baseline model uncomment the following ##

# dat_train = training features
# label_train = labels
# K = number of folds
# d = a certain interaction depth
# system.time(result<-gbm_train(train_data[,2:ncol(train_data)],train_data$label))
# result

#
```

Our Model: XGBoost + PCA + SIFT

```r
# Basic model
basic = xgboost(data = train_matrix,
                max.depth=3,eta=0.01,nthread=2,nround=50,
                objective = "multi:softprob",
                eval_metric = "mlogloss",
                num_class = 3,
                verbose = F)
```

```r
# Tune the model
xgb_params_3 = list(objective="multi:softprob",
                    eta = 0.01,
                    max.depth = 3,
                    eval_metric = "mlogloss",
                    num_class = 3)

# fit the model with arbitrary parameters
xgb_3 = xgboost(data = train_matrix,
                params = xgb_params_3,
                nrounds = 100,
                verbose = F)

# cross validation
xgb_cv_3 = xgb.cv(params = xgb_params_3,
                  data = train_matrix,
                  nrounds = 100,
                  nfold = 5,
                  showsd = T,
                  stratified = T,
                  verbose = F,
                  prediction = T)

# set up the cross validated hyper-parameter search
xgb_grid_3 = expand.grid(nrounds=c(100,250,500),
                         eta = c(1,0.1,0.01),
                         max_depth = c(2,4,6,8,10),
                         gamma=1,
                         colsample_bytree=0.5,
                         min_child_weight=2,
                         subsample = 1)

# pack the training control parameters
xgb_trcontrol_3 = trainControl(method = "cv",
                               number = 5,
                               verboseIter = T,
                               returnData = F,
                               returnResamp = "all",
                               allowParallel = T)

# train the model for each parameter combination in the grid

ptm <- proc.time() ## start the time
```

```r
xgb_train_3 = train(x=train_matrix, y=train_label,
                    trControl = xgb_trcontrol_3,
                    tuneGrid = xgb_grid_3,
                    method = "xgbTree")
```

```r
ptm2 <- proc.time()
ptm2- ptm ## stop the clock1
```

```
##    user  system elapsed
##  545.75   83.74  397.39
```

```r
# ## Time for training: 350.92s
#
head(xgb_train_3$results[with(xgb_train_3$results,order(RMSE)),],5)
```

```
##     eta max_depth gamma colsample_bytree min_child_weight subsample
## 21 0.10         4     1              0.5                2         1
## 20 0.10         4     1              0.5                2         1
## 9  0.01         6     1              0.5                2         1
## 19 0.10         4     1              0.5                2         1
## 24 0.10         6     1              0.5                2         1
##    nrounds      RMSE  Rsquared       MAE      RMSESD RsquaredSD
## 21     500 0.5110378 0.6141201 0.4175375 0.009210265 0.01961039
## 20     250 0.5116581 0.6133756 0.4187183 0.009530715 0.01967853
## 9      500 0.5134024 0.6246922 0.4314855 0.010377619 0.02250139
## 19     100 0.5137137 0.6108225 0.4225101 0.010866963 0.02121169
## 24     500 0.5138771 0.6147863 0.4187986 0.016686129 0.02848466
##          MAESD
## 21 0.007929596
## 20 0.007746908
## 9  0.004287987
## 19 0.009226362
## 24 0.010063041
```

```r
# get the best model's parameters
xgb_train_3$bestTune
```

```
##    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 21     500         4 0.1     1              0.5                2         1
```

```r
# # best model
bst = xgboost(data=train_matrix,max.depth=4,eta=0.1,nthread=2,nround=250,colsample_bytree=0.5,min_child_

pred = predict(bst, test_matrix)
prediction<-matrix(pred,nrow = 3,ncol = length(pred)/3) %>%
  t() %>%
  data.frame() %>%
  mutate(label=test_label+1,max_prob=max.col(.,"last"))

# ## confusion matrix of test set
confusionMatrix(factor(prediction$label),factor(prediction$max_prob),mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 294   5   7
##          2  13 243  52
##          3  19  62 205
##
## Overall Statistics
##
##                Accuracy : 0.8244
##                  95% CI : (0.798, 0.8488)
##     No Information Rate : 0.3622
##     P-Value [Acc > NIR] : < 2e-16
```

```
##
##                   Kappa : 0.7363
##  Mcnemar's Test P-Value : 0.01881
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9018   0.7839   0.7765
## Specificity            0.9791   0.8898   0.8726
## Pos Pred Value         0.9608   0.7890   0.7168
## Neg Pred Value         0.9461   0.8868   0.9039
## Precision              0.9608   0.7890   0.7168
## Recall                 0.9018   0.7839   0.7765
## F1                     0.9304   0.7864   0.7455
## Prevalence             0.3622   0.3444   0.2933
## Detection Rate         0.3267   0.2700   0.2278
## Detection Prevalence   0.3400   0.3422   0.3178
## Balanced Accuracy      0.9405   0.8369   0.8246
```

```
# ## Accuracy: 82.67%
# ## Parameters: max.depth=4, eta=0.1, nthread=2, nround=250
```