Project 3 - Labradoodle or Fried Chicken? In Blakc and White

Group 5

March 24, 2017

In this project, we created a classification engine for grayscale images of poodles versus images of fried chickens. The baseline model uses gradient boosting machine with decision stumps on 5000 SIFT features. Our advanced models include linear and kernel SVM (support vector machine), KNN(k-nearest neighbors) on both the HoG-features and the SIFT features. Our best performance baseline model achieved an accuracy of 74% and 88% for the advanced model.

```
install.packages("caret")
install.packages("gbm")
install.packages("xgboost")
install.packages("e1071")

library(gbm)

library(caret)

library(xgboost)
library(e1071)
```

Step 0: Import visual features and prepare training and testing data

First, set the working directory to the folder where features are stored. HoG features are generated from MATLAB and we keep the first 200 principal components to avoid overfitting. Specify the training and the testing set to create your own testing data by random subsampling. In order to obain reproducible results, set.seed() whenever randomization is used. If you are interested in reading raw images, please follow the path ../spr2017-proj3-group5/data/raw_images. We introduced HoG features into our model because it typically works in a sliding window. When we scan the images, we realized that most fired chicken pictures share similar background and we can avoid the overfitting problem by processing hog_features with PCA. We made a cross validation and decided to keep only the first 200 PCs.

```
setwd("/Users/apple/Documents/R/spr2017-proj3-group5/data")
img_sift_feature_raw <- read.csv("sift_features.csv")
img_sift_feature <- data.frame(t(img_sift_feature_raw))
img_hog_feature_raw <- read.csv("hog_features.csv",header = FALSE)
img_hog_feature_raw <- img_hog_feature_raw[,1:200]
img_hog_feature <- data.frame(img_hog_feature_raw)
#set test and training data
set.seed(1)
test_ind <- sample(1:5,2000,replace = T)
hog_test <- img_hog_feature[which(test_ind==5),]
```

```
hog_train <-img_hog_feature[which(test_ind!=5),]
sift_test <- img_sift_feature[which(test_ind==5),]
sift_train <-img_sift_feature[which(test_ind!=5),]
save(test_ind, file="test_ind.RData")
```

Step 1: set up controls for evaluation experiments.

In this chunk, we have a set of controls for the evaluation experiments.

- (T/F) cross-validation on the training set
- (number) K, the number of CV folds
- (T/F) process features for training set
- (T/F) run evaluation on an independent train set
- (T/F) run evaluation on an independent test set

```
# run cross-validation on the training set
run.cv=TRUE
# number of CV folds
K <- 5
#run evaluation on an independent train set
run.train=TRUE
 # run evaluation on an independent test set
run.test=TRUE
```

Using cross-validation and independent test set evaluation, we compare the performance of different classifiers or classifiers with different specifications. In this example, we use GBM with different `n.trees`. In the following chunk, we list, in a vector, setups (in this case, `n_trees`) corresponding to models that we will compare. If you want to change the parameter range you want to test, you can just edit them here.

```
#for baseline gbm
depth <- 1   #set interaction.depth
n_trees <- seq(1000, 2000, 100)
shrinkage <- 0.01
model_label_gbm = paste("GBM with n_trees =", n_trees)

#for advanced svm_radial
cost <- seq(5,20,5)
gamma <- seq(0.001,0.009,0.002)
model_labels_svm = paste("svm with cost=",cost," gamma=",gamma)
```

Step 2: import images class labels.

In this step, you can import class labels for train and test subset and factorize them for classification use.

```
setwd("/Users/apple/Documents/R/spr2017-proj3-group5-2/data")
img_label <- read.csv("labels.csv")
label_test<-img_label[which(test_ind==5),]
label_train<-img_label[which(test_ind!=5),]
```

```
label_train <- factor(label_train)
label_test <- factor(label_test)
```

Step 3: Train a classification model with training images

Call the train model and test model from library.

`train.R` and `test.R` should be wrappers for all your model training steps and your classification/prediction steps. + `train.R`

- Input: a path that points to the training set features.
- Input: an R object of training sample labels.
- Output: an RData file that contains trained classifiers in the forms of R objects: models/settings/links to external trained configurations.
- `test.R`
- Input: a path that points to the test set features.
- Input: an R object that contains a trained classifiers.
- Output: an R object of class label predictions on the test set. If there are multiple classifiers under evaluation, there should be multiple sets of label predictions.

```
source("../lib/train.R")
source("..lib/test.R")
```

Model selection with cross-validation and Choose the "best" parameter value

- Do model selection by choosing among different values of training model parameters, that is, the interaction depth for GBM and cost for advanced model(SVM).
- Train the model with the entire training set using gbm baseline model and advanced svm model.

```
if(run.train){
  train_result <- fit_train(best_ntree)
  save(train_result, file="../output/fit_train.RData")
}
```

Step 5: Make prediction

Feed the final training model with the completely holdout testing data.

```
tm_test=NA
if(run.test){
  test_result <- fit_test()
  load(file=paste0("../output/feature_", "zip", "_", "test", ".RData"))
  load(file="../output/fit_train.RData")
}
```

Summarize Running Time

Prediction performance matters, here we give the predict error and run time of our baseline model and advanced svm models.

```
#{r running_time,echo=FALSE}

## Time for training baseline gbm model= 268.985 s

## Time for training advanced svm model= 4.114 s

## Time for making prediction using baseline gbm model 0.126 s

## Time for making prediction using advanced svm model 0.307 s

## The test error of baseline gbm model= 0.2916667 s

## The test error of advanced svm model= 0.1372549 s
```