Model Selection

Contents

Step 0: specify directories.

Set up the directories. Please download weights.h5 and put it in the output folder.

```
#setwd('/Users/jinruxue/Documents/ADS/spr2017-proj3-group3/')
setwd('./')
#train_img_dir <- 'data/raw_images/'
baseline_train_features <- 'data/sift_features.csv'
advanced_train_orig_features <- 'output/cnn_features.csv'
advanced_train_features <- 'output/cnn_features_350.csv'
train_labels <- 'data/labels.csv'
test_img_dir <- 'data/test/'
test_sift_features <- 'data/test_sift_features.csv'
orig_test_cnn_features <- 'output/orig_test_cnn_features.csv'
test_cnn_features <- 'output/test_cnn_features.csv'
weights_url <- 'https://s3.amazonaws.com/doodle-vs-friedchicken/weights.h5?response-content-disposition=
```

Step 1: set up controls for 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 test set
- (T/F) run evaluation on an independent test set

```
run.cv=TRUE # run cross-validation on the training set
K <- 5  # number of CV folds
run.feature.train=FALSE # process features for training set
run.evaluation=TRUE #if true, split training set into training and test sets, use cv and grid search to
run.test=TRUE # run evaluation on an independent test set
run.feature.test=TRUE # process features for test set
```

Step 2: construct visual features

Here we extract features from a CNN network. We use Python to process it so make sure you have following python packages installed with python2.7:

I recomment you to to lib directory in command line run

```
source('train.R')
if(run.feature.train){
  cat('Have you run computeCNNFeatures.py?')
  tm_select_features_train <- system.time(pca_model <-selectFeatures(advanced_train_orig_features,advan
  save(pca_model,file = 'output/pcaModel.RData')
  #save(pca_model[[2]],file = 'index.RData')
}
```

After this step you should be able to access reduced cnn features stored in 'output/cnn_features_350.csv'.

Step 3: Train baseline model

```
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
## Distribution not specified, assuming bernoulli ...
```

Step 4: Train advanced models

Since we use ensemble model as our best model, here we train svm, xgboost, randomforest, logistic regression and later on ensemble them for prediction.

```
library("glmnet")
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```
library(e1071)

n.features <- 350
tm_advanced_train <- NA

tm_advanced_train1 <- system.time(svm_fit <- svm_adv(advanced_train_features,n.features,run.cv))
save(svm_fit,file = 'output/svmModel.RData')

tm_advanced_train2 <- system.time(lr_fit <- lr_adv(advanced_train_features,n.features,run.cv))
save(lr_fit,file = 'output/lrModel.RData')

tm_advanced_train <- tm_advanced_train1 + tm_advanced_train2
```

Step 5: Make prediction

If run.evaluation is true, make prediction on separated training set(not used in training process), if run.test is true, make prediction on the new test set.

```
source('test.R')
# load trained models
load('output/baselineModel.RData')
load('output/svmModel.RData')
load('output/lrModel.RData')
```

```
load('output/pcaModel.RData')
#load('output/index.RData')
n.bl.features <- 5000
n.adv.features <- 350

if(run.feature.test){
  # compute cnn features for test data
  # use pca to get 350 out out 2048 features
  pca.mod <- pca_model[[1]]
  train.sd <- pca_model[[2]]

  tm_select_features <- system.time(selectFeatures(orig_test_cnn_features,test_cnn_features,run.feature

}

if(run.evaluation){
  # process and split test data
  base.test.data <- split.train(baseline_train_features)
  adv.test.data <- split.train(advanced_train_features)
  # use test.R to make prediction with gbm_fit
  tm_bl_pred <- NA
  tm_bl_pred <- system.time(base.test.pred <- test(gbm_fit,base.test.data,n.bl.features))
  # or with best.fits(several models) and ensemble their results
  tm_adv_pred <- NA
  # tm_adv_pred <- system.time(adv.test.pred <- test(svm_fit,adv.test.data,n.adv.features))
  tm_adv_pred1 <- system.time(adv.test.pred1 <- test(svm_fit,adv.test.data,n.adv.features))
  tm_adv_pred2 <- system.time(adv.test.pred2 <- test(lr_fit,adv.test.data,n.adv.features))
  tm_adv_pred <- tm_adv_pred1 + tm_adv_pred2
  adv.test.pred <- (adv.test.pred1+adv.test.pred2)/2
  # give evaluation(accuracy)
  acc.bl <- sum((base.test.pred>0.5)==base.test.data$label)/dim(base.test.data)[1]
  acc.adv <- sum((adv.test.pred>0.5)==adv.test.data$label)/dim(adv.test.data)[1]
  cat('baseline model accuracy is',acc.bl,'\nadvanced model accuracy is',acc.adv)
}
```

```
## baseline model accuracy is 0.7125
## advanced model accuracy is 0.9525
```

```
if(run.test){
  # process test data
  base.test.data <- process.test(test_sift_features)
  adv.test.data <- process.test(test_cnn_features)
  # use test.R to make prediction with gbm_fit
  tm_bl_pred <- NA
  tm_bl_pred <- system.time(base.test.pred <- test(gbm_fit,base.test.data,n.bl.features))
  # or with best.fits(several models) and ensemble their results
  tm_adv_pred <- NA
  tm_adv_pred1 <- system.time(adv.test.pred1 <- test(svm_fit,adv.test.data,n.adv.features))
  tm_adv_pred2 <- system.time(adv.test.pred2 <- test(lr_fit,adv.test.data,n.adv.features))
  tm_adv_pred <- tm_adv_pred1 + tm_adv_pred2
  adv.test.pred <- (adv.test.pred1+adv.test.pred2)/2
  # save the pred results into 2 csv
  result.bl <- as.integer(base.test.pred>0.5)
  result.adv <- as.integer(adv.test.pred>0.5)
```

```
  save.pred <- function(pred,images,output_name){
    output <- data.frame(predict_label=as.numeric(pred>0.5))
    output$image <- images
    output$labradoodle <- pred
    output$friedChicken <- 1-pred
    fwrite(output,file=output_name)
  }
  save.pred(base.test.pred,base.test.data$image,'output/baseline_test_pred.csv')
  save.pred(adv.test.pred,adv.test.data$image,'output/advance_test_pred.csv')
  cat('baseline model prediction for test set is(0 for fried chicken, 1 for labradoodle) ', result.bl)
  cat('advanced model prediction for test set is(0 for fried chicken, 1 for labradoodle) ', result.adv)
}
```

```
## baseline model prediction for test set is(0 for fried chicken, 1 for labradoodle)  0 0 1 0 0 0 1 1 1
```

```
# summarize training time for baseline and advanced models
# summarize test time for feature-preprocess, baseline and advanced models
```