# Report

2018年12月7日 10:44

## Introduce how you implement each classifier:

### KNN

(1) a description of how you formulated the problem:

Using knn algorithm, we will find n high-dimensional point which is nearst to the point we want to classify.Then, we will find out the label of most point we picke out to be the predict label.

(2) a brief description of how your program works:

This program including model_knn_train , model_knn_test and get_knn_train_result funtions. The other small funcions are called by them.

The structure of this program is:

```
model_knn_train(train_file='train-data.txt',model_file='knn-model.txt',k_range = 3,times_train = 1)
    readTrainData(train_file)
    generateTrainReport(imgTrainIds,imgTrainOrientations,imgTrainVectors,times_train,k_range)
        trainKNN(trainIds,trainOrientations,trainVectors,k)
            testKNNAccurcy(partImgTrainIds,partImgTrainOrientations,partImgTrainVectors,partImgValIds,partImgValOrientations,partImgValVectors,k)
                kNN_classify(imgTestVectors[i], imgTrainVectors, imgTrainOrientations, k)
    save_train_result_to_excel.saveTrainResult()
    generateTrainModelFile(model_file,bestK,imgTrainIds,imgTrainVectors,imgTrainOrientations)
model_knn_test(test_file='test-data.txt',model_file='knn-model.txt')
    readTrainModelFile(model_file)
    readTrainData(test_file)
    testKNNAccurcy(imgTrainIds,imgTrainOrientations,imgTrainVectors,imgTestIds,imgTestOrientations,imgTestVectors,bestK)
        kNN_classify(imgTestVectors[i], imgTrainVectors, imgTrainOrientations, k)
    generateTestOutputFile(outputFilename,predictSet)
    show_image_to_html.show_result_on_html(predictTrueSet,htmlTrueFile,predictFalseSet,htmlFalseFile)
get_knn_train_result()
    module_knn_train(train_file ,model_file ,k_range ,ratio_train)
```

(3) a discussion of any problems, assumptions, simplifications, and/or design decisions you made:

We find that if the training set is large, our training will cost a lot of time.

We design the function of 'get_knn_train_result()' to help us automaticly split the origin training set and get different training result and then generate the training result to excel file for analysis.

Besides, we save the classfication result both right and wrong in the html file to help us clearly find the patterns to the erros.

### AdaBoost

(1) a description of how you formulated the problem:

Using adaboost algorithm, we use the training set to generate several weak classifiers. In testing progress, we cascade weak classifiers to form a strong classifer. We add the predict result of weak classifiers to get a final predict result.

(2) a brief description of how your program works:

This program including model_adaboost_train , model_adaboost_test and get_adaboost_train_result funtions. The other small funcions are called by them.

The structure of this program is:

```
model_adaboost_train(train_file='train-data.txt',model_file='adaboost_model.txt',num_iteration=400,ratio_train = 1)
    demo_forest.readFile(train_file,featNum,mode)
    adaBoostTrainDS(dataArr,classLabels,num_iteration)
        buildStump(dataArr,classLabels,D)
            stumpClassify(dataMatrix,i,threshVal,inequal)
    generate_adaboost_model(model_file,weakClassArr)

model_adaboost_test(test_file='test-data.txt',model_file='adaboost_model.txt')
    demo_forest.readFile(test_file,featNum,mode)
    test_adaboost_accurcy(img_ID,img_test,labels_test,weakClassArr)
    generate_adaboost_output_file(outputFilename,img_ID,predictList)
    show_image_to_html.show_result_on_html(predictTrueSet,htmlTrueFile,predictFalseSet,htmlFalseFile)

get_adaboost_train_result()
    model_adaboost_train(train_file,model_file,num_iteration,ratio_train)
    save_train_result_to_excel.saveTrainResult(titles,trainResults)
```

(3) a discussion of any problems, assumptions, simplifications, and/or design decisions you made:

We find the design of this wek classifier can only have two categories that is ture of false.But we have four categories need to classify. So in the training and testing progress, we set all labels to -1 if they aren't 0 degree. So we get right format of labels. However, how can we get true testing result.? We can only tell whether the orientation is 0 degree or not. To solve this problem, we resize the feature vectors to square and rotate them for 90 degree or 180 degree or 270 degree. Then we put the processed image to the classifier again to get the result. If we get the predict label in the case of rotating 90 degree.That means this picture has rotated 90 degree before. So we can get multiple classifications.

Forest

(1) a description of how you formulated the problem:

To achieve a random forest, we have to build a decision tree first. To build a decision tree, we need some basic functions, for example, a function to split the dataset with some rules, a function to calculate the entropy of a dataset. We decide to store the tree in dict format, so we need a function to analyze a stored tree. After this, we can train many trees and construct a forest.

(2) a brief description of how your program works:

This program including model_forest_train and model_forest_test funtions. The other small funcions are called by them.

The structure of this program is:

```
model_forest_train(train_file='train-data.txt',model_file='forest_model.txt',featNum=32,mode='rgb',treeNum=50)
    readFile(train_file,featNum,mode)
    createTree(trainSet,labels)
        chooseBestFeatureToSplit(dataSet)
            calcEnt(dataSet)
            calcInfoGain(dataSet,i,value,baseEntropy)
        createTree(trainSet,labels)
        …….
    storeTree(randomForest,model_file,featNum,mode)
model_forest_test(test_file='test-data.txt',model_file='forest_model.txt')
    readTree(model_file)
    readFile(test_file,featNum,mode)
    voteClassify(reloadForest,testImg)
        classifyAll(decisionTree,testDataSet)
```

(3) a discussion of any problems, assumptions, simplifications, and/or design decisions you made:

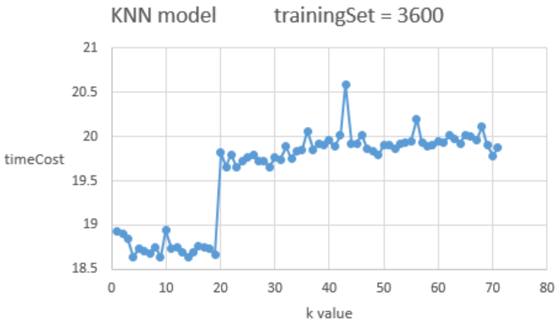
We referenced a source code of a textbook. However, what is different is that our data is int format. So we need to change the way we split the feature and divide the dataset. It cost quite a lot of time.

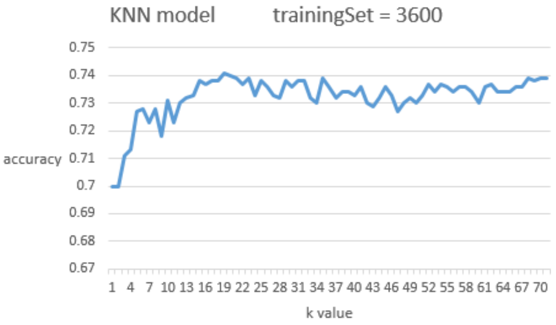
We find that the more features we pick, the higher the accuracy is, with the price of a lot of training time. It is hard to find a balance between accuracy and training time.

We also find an upper limit on accuracy. When we increased the number of trees from 50 to 100, the accuracy only increased a little. We believe that accuracy will not increase without limit.

## Present neatly-organized tables or graphs showing classication accuracies and running times as a function of the parameters you choose.

1. Using the knn way, we post the training result of with different k value when the size of training set is 3600.



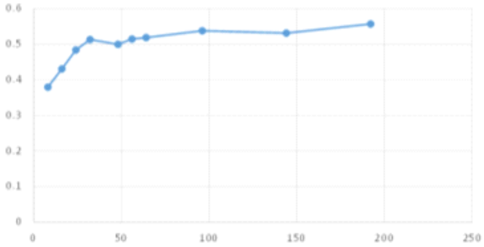


2. Using the random forest way, we need to train a decision tree first. We post the result of a single decision tree below. We use 500 pictures to train and ‘test-data.txt’ to get the result.

| FeatureNum | Accuracy rate | Training time/s |
|---|---|---|
| 8 | 0.3796394485683987 | 1.220587985477323 |
| 16 | 0.4305408271474019 | 3.421173229226042 |
| 24 | 0.4835630965005302 | 4.80559430448011 |
| 32 | 0.513255567338282 | 6.9898500861036155 |
| 48 | 0.49946977730646874 | 11.104106148221035 |

|  |  |  |
|---|---|---|
| 56 | 0.5143160127253447 | 13.94556450896016 |
| 64 | 0.5185577942735949 | 17.2981764367305 |
| 96 | 0.5376458112407211 | 29.18401044858183 |
| 144 | 0.5312831389183457 | 49.493924550410156 |
| 192 | 0.5726405090137858 | 76.28177388826771 |

We can also plot it.



Here is the result when we use 50 trees.

| FeatureNum | Accuracy rate | Training time/s |
|---|---|---|
| 16 | 0.5546129374337222 | 162.1433158650543 |
| 32 | 0.6002120890774125 | 373.5613685701355 |
| 96 | 0.6383881230116649 | 1716.581987196077 |
| 192 | 0.7020148462354189 | 4432.005874951358 |

We can find that random forest provide a quite good result when we use more features. But it also cost a lot of time to train.

## Which classiers and which parameters would you recommend to a potential client?

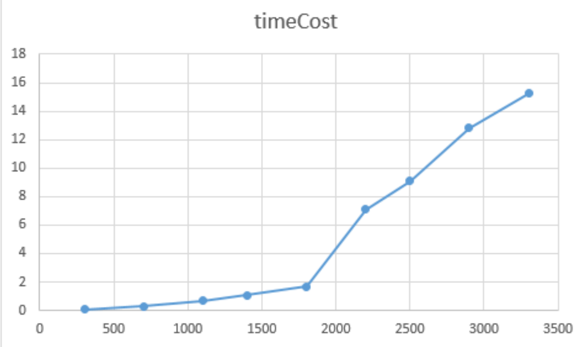
If we can use a pre-trained model, we recommend to use decision forest with 100 trees to train 192 features for every image.In our test dataset, we achieved an accuracy up to 0.7243.

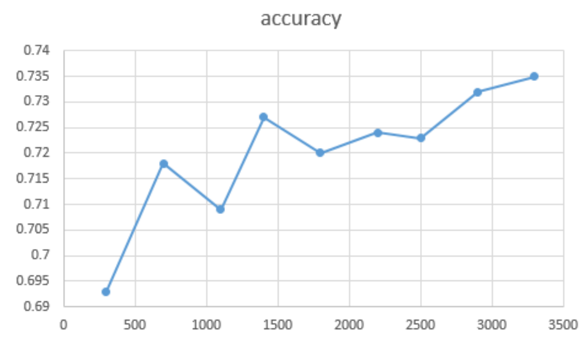
When we need to use a new training set to get model, we recommend to use KNN. Because the training process of KNN is very fast.

## How does performance vary depending on the training dataset size, i.e. if you use just a fraction of the training data?

For KNN model, we use different size of training set and get result below.

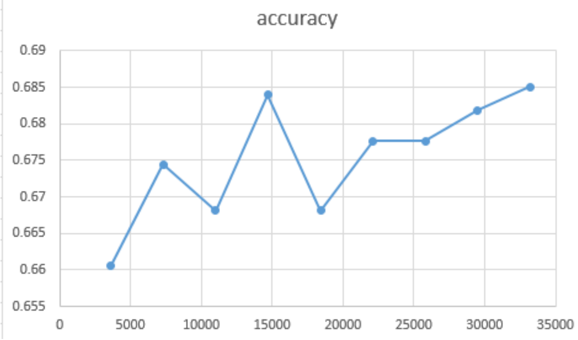
| trainingSetSize | K | accuracy | timeCost |
|---|---|---|---|
| 300 | 20 | 0.693 | 0.06 |
| 700 | 20 | 0.718 | 0.3 |
| 1100 | 20 | 0.709 | 0.67 |
| 1400 | 20 | 0.727 | 1.06 |
| 1800 | 20 | 0.72 | 1.67 |
| 2200 | 20 | 0.724 | 7.08 |
| 2500 | 20 | 0.723 | 9.07 |
| 2900 | 20 | 0.732 | 12.82 |
| 3300 | 20 | 0.735 | 15.25 |

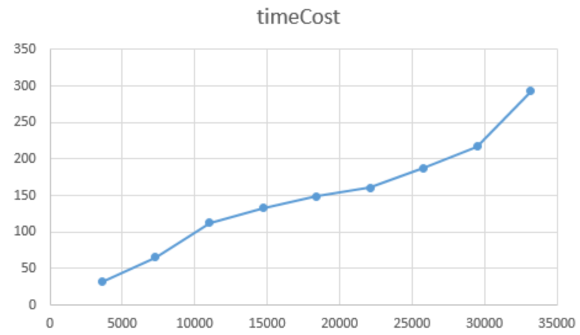




For AdaBoost model, we use different size of training set and get result below.

| trainingSetSize | num_iteration | accuracy | timeCost |
|---|---|---|---|
| 3600 | 50 | 0.660657476 | 31.54612446 |
| 7300 | 50 | 0.674443266 | 65.29101825 |
| 11000 | 50 | 0.668080594 | 112.5712681 |
| 14700 | 50 | 0.683987275 | 132.887419 |
| 18400 | 50 | 0.668080594 | 148.8502281 |
| 22100 | 50 | 0.677624602 | 160.4878871 |
| 25800 | 50 | 0.677624602 | 187.7746556 |
| 29500 | 50 | 0.681866384 | 217.0924478 |
| 33200 | 50 | 0.68504772 | 293.3234987 |

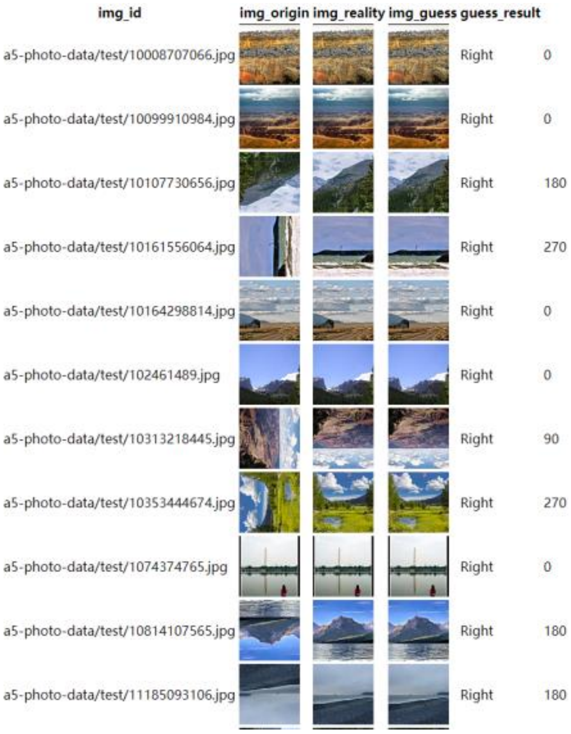




As we can see in the graph,in general,the accuracy increased when the training set gets lager.
The cost time of training process is obviously get bigger  when the training set gets lager.

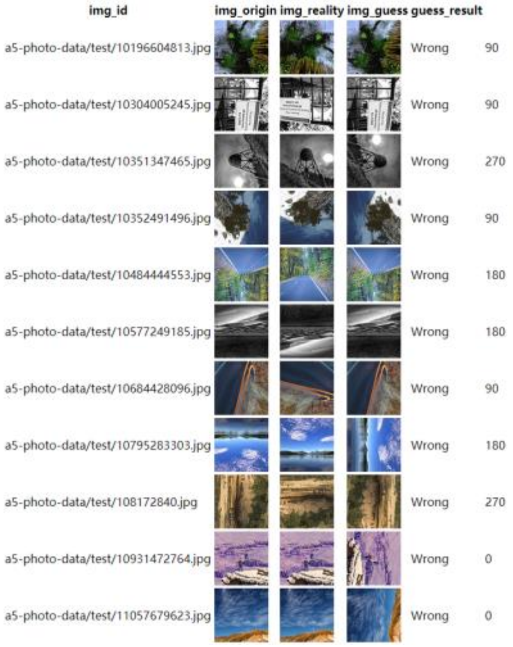
## Show a few sample images that were classied correctly and incorrectly. Do you see any patterns to the errors?

For AdaBoost model, a few sample images that were classied correctly are shown below.



| img_id | img_origin | img_reality | img_guess | guess_result | |
|---|---|---|---|---|---|
| a5-photo-data/test/10008707066.jpg | | | | Right | 0 |
| a5-photo-data/test/10099910984.jpg | | | | Right | 0 |
| a5-photo-data/test/10107730656.jpg | | | | Right | 180 |
| a5-photo-data/test/10161556064.jpg | | | | Right | 270 |
| a5-photo-data/test/10164298814.jpg | | | | Right | 0 |
| a5-photo-data/test/102461489.jpg | | | | Right | 0 |
| a5-photo-data/test/10313218445.jpg | | | | Right | 90 |
| a5-photo-data/test/10353444674.jpg | | | | Right | 270 |
| a5-photo-data/test/1074374765.jpg | | | | Right | 0 |
| a5-photo-data/test/10814107565.jpg | | | | Right | 180 |
| a5-photo-data/test/11185093106.jpg | | | | Right | 180 |

For AdaBoost model, a few sample images that were classied incorrectly are shown below.



| img_id | img_origin | img_reality | img_guess | guess_result | |
|---|---|---|---|---|---|
| a5-photo-data/test/10196604813.jpg | | | | Wrong | 90 |
| a5-photo-data/test/10304005245.jpg | | | | Wrong | 90 |
| a5-photo-data/test/10351347465.jpg | | | | Wrong | 270 |
| a5-photo-data/test/10352491496.jpg | | | | Wrong | 90 |
| a5-photo-data/test/10484444553.jpg | | | | Wrong | 180 |
| a5-photo-data/test/10577249185.jpg | | | | Wrong | 180 |
| a5-photo-data/test/10684428096.jpg | | | | Wrong | 90 |
| a5-photo-data/test/10795283303.jpg | | | | Wrong | 180 |
| a5-photo-data/test/108172840.jpg | | | | Wrong | 270 |
| a5-photo-data/test/10931472764.jpg | | | | Wrong | 0 |
| a5-photo-data/test/11057679623.jpg | | | | Wrong | 0 |

We  found that the images that were classied incorrectly has enough reason to get such result of classification. Because even for person, he can not tell the true orientation easily when the images have no obviously directionality.Those classied correctly often have sky on the top of image. The color on top of these image often has Continuity