```
####################################
# CS B551 Fall 2017, Assignment #4
#
# Your names and user ids:
# Name: Yingnan Ju; ID: yiju
# Name: Yue Chen; ID: yc59
#
####################################
```

(1) a description of how you formulated the problem, including precisely defining the abstractions;

We just used the data that the professor provided. The problem could be summarized to be predict the orientation of a picture through detect the color of (all or some) pixels of the thumbs.

KNN, or nearest: regarding the color of each picture as a vector and compare the vector of the picture to be predicted with all vectors in the training set and find K vectors that are most similar with that vector (two vectors that have min distance between them)

$$RSS = \sum_{i=1}^{n} (Y_i - f(x_i))^2$$

Adaboost: or Adaptive Boosting. A boost classifier is a classifier in the form

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

Each f_t(x) is a weak learner that takes an object x as input and returns a value indicating the class of the object [1]. In this case, it is a four-class problem and we get four classifiers eventually and each tells us if the picture is in this direction or not. If there are more positive answers or all answers are negative, we will compare the confidences of each classifier. The sign of the weak learner output identifies the predicted object class and the absolute value gives the confidence in that classification [1]. In this case, as the homework pdf said, we use simple decision stumps that simply compare the color of two different pixel. We will describe in the following part that how we chose the different pixels.

Nnet: or neural network. We defined a three layers neural network. Input layer, hidden layer and output layer. The number of nodes in input layer is same with the input features. The number of nodes in output layer is same with the number of output results. In this case we have four different answers so we defined four outputs, similar with the method in adaboost part. Similar, if there are more positive answers or all answers are negative, we will compare the confidences of each classifier. The number of nodes in the hidden layer was a parameter and in the following part we will describe how to decide it. The relation between input and output of each layer is:

$$I_j = \sum_i w_{ij} O_i$$

$$O_j = sigmod(I_l) = \frac{1}{1 + e^{-Il}}$$

This is the feed-forward process. We trained the network with backpropagation algorithm.

$$E_j = sigmod'(O_i) * \sum_k E_k w_{jk}$$

$$w_{jk} = w_{ij} + \lambda E_j O_i + \mu C_{ij}$$

Best: same with KNN or nearest.

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

KNN, or nearest:

Training part: simply copy all content train file into the model file (nearest_model.txt).

Test part: for each picture in the test set, compare it with all vectors in the training set and find out which was the most common answer in the k nearest vectors.

Adaboost: or Adaptive Boosting:

Training part: take the color in index 172 (blue) and 3 (green) as a simple stump. Create four classifiers and each of them indicate one orientation of a picture. Repeat the iteration to train the classifiers in limited times or the error rate is 0, which is almost impossible.

Test part: for each picture in the test set we took the color in index 172 (blue) and 3 (green) and input them in each classifier and get the most possible answer as the prediction.

Nnet: or neural network.:

Training part: take the color in index 172 (blue) and 3 (green) as the input so the input layer of the network could be simplified as two nodes. Create four networks and each of them indicate one orientation of a picture. Repeat the iteration to train the networks in limited times with different parameters to find the best ones.

Test part: for each picture in the test set we took the color in index 172 (blue) and 3 (green) and input them in each classifier and get the most possible answer as the prediction.

Best: same with KNN or nearest.

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

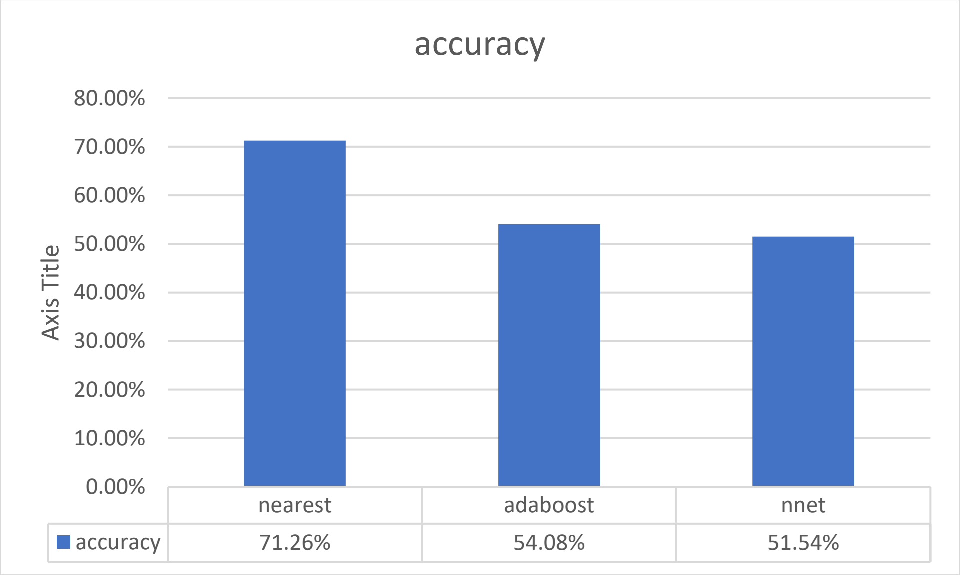
Most classifiers or networks could only have two labels and the ones for multi labels are much more complicated. So, we use multi classifiers or networks to get the similar result. If there are more positive answers or all answers are negative, we will compare the confidences of each classifier or network.

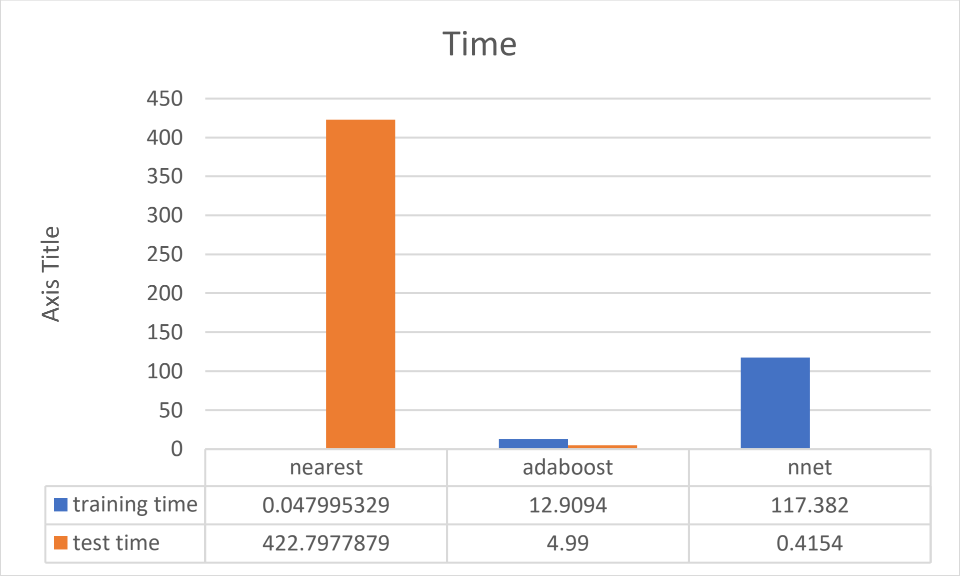
(4) answers to any questions asked below in the assignment.

In your report, present neatly-organized tables or graphs showing classification accuracies and running times as a function of the parameters you choose.
Table:

	nearest	adaboost	nnet
accuracy	71.26%	54.08%	51.54%
training time	0.047995	12.9094	117.382
test time	422.7978	4.99	0.4154

Graph:





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

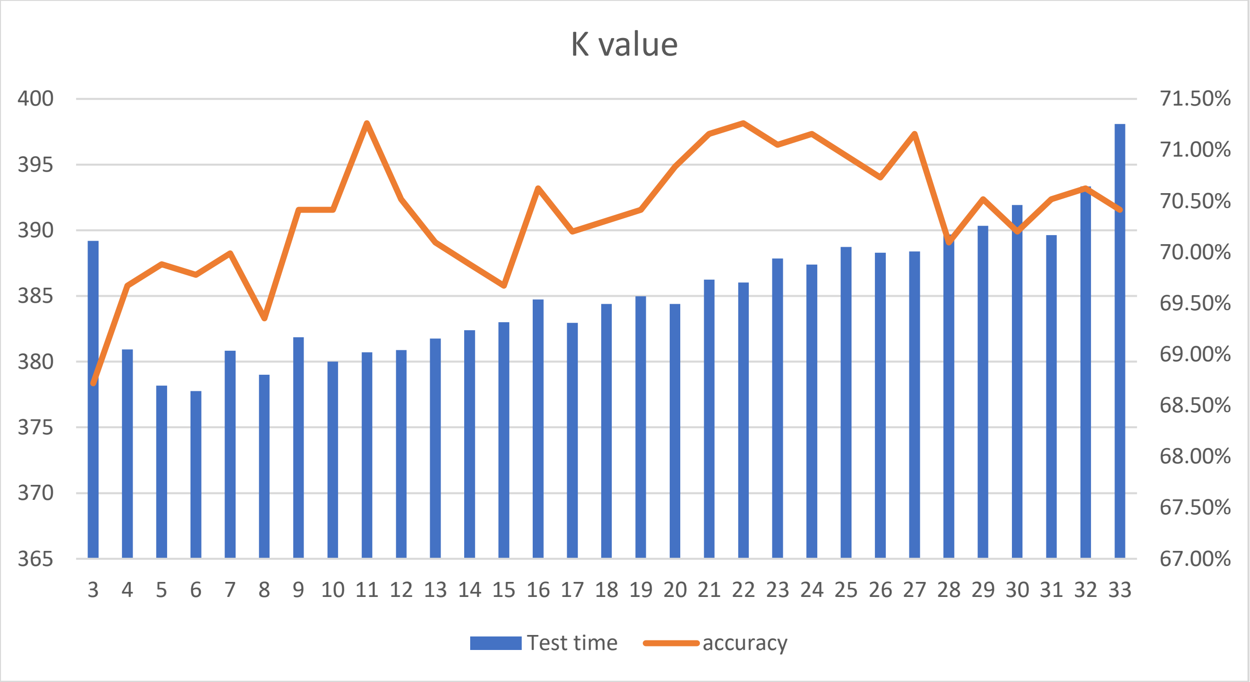
Knn: for those occasion that is not time limited, I would recommend Knn, which is easy to implement and the result is rather good. The disadvantage is that it costs a lot of test time.
If it is time limited, I would recommend adaboost, which has acceptable training time and test time.

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

KNN, nearest:

Full training set.

Here is the graph showing the relation between the accuracy and K value:

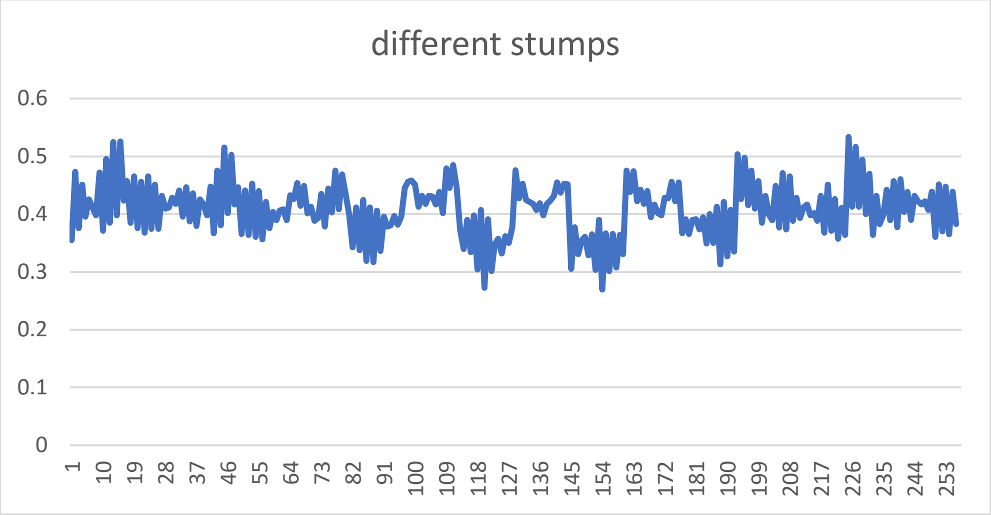


We can see that as k increases, test time increases. However, we get the highest accuracy at k=11.

Adaboost:

Full training set with simple stump.

We tried 16^2=256 kinds of stump and the result is:



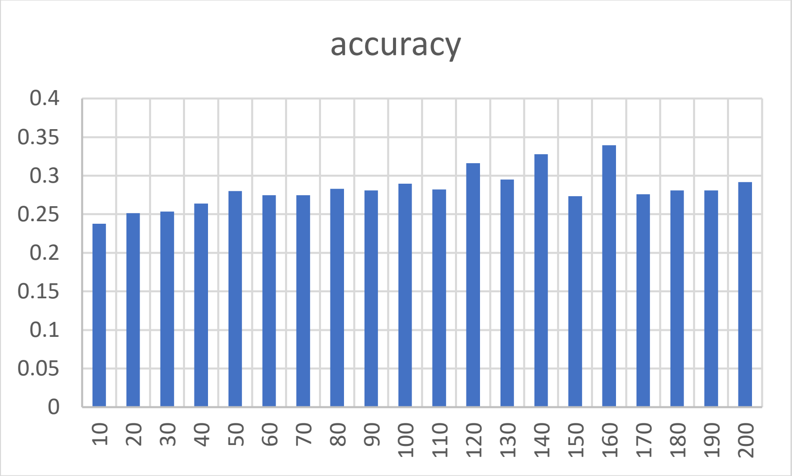
The best answer is point (172,3).

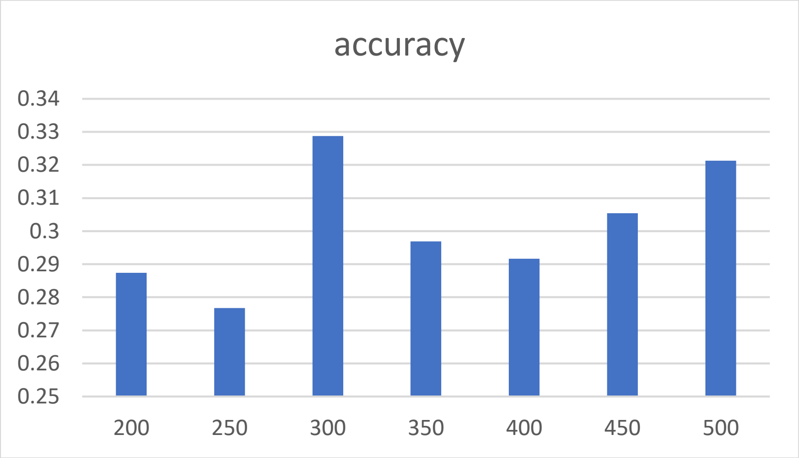
NNet:

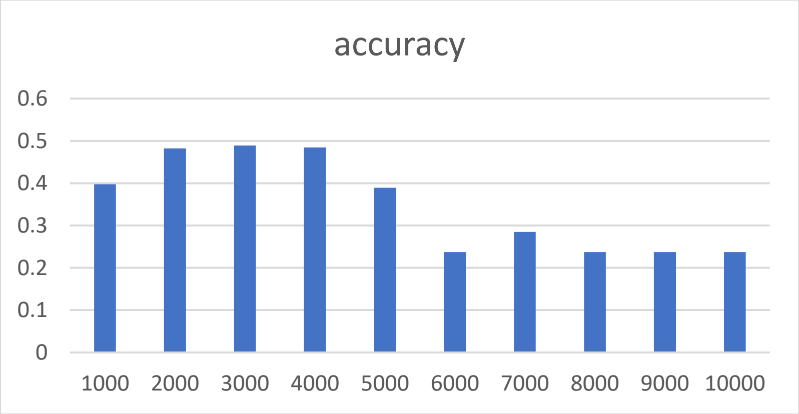
We don’t use all training set and the relation between size of training set and the accuracy is:

input	accuracy	input	accuracy	input	accuracy
10	0.23754	200	0.287381	1000	0.397667
20	0.251326	250	0.276776	2000	0.482503
30	0.253446	300	0.328738	3000	0.488865
40	0.264051	350	0.296925	4000	0.484624
50	0.279958	400	0.291622	5000	0.389183
60	0.274655	450	0.305408	6000	0.23754
70	0.274655	500	0.321315	7000	0.28526
80	0.283139			8000	0.23754
90	0.281018			9000	0.23754
100	0.289502			10000	0.23754
110	0.282078				
120	0.316013				
130	0.294804				
140	0.327678				
150	0.273595				
160	0.339343				
170	0.275716				
180	0.281018				
190	0.281018				
200	0.291622				

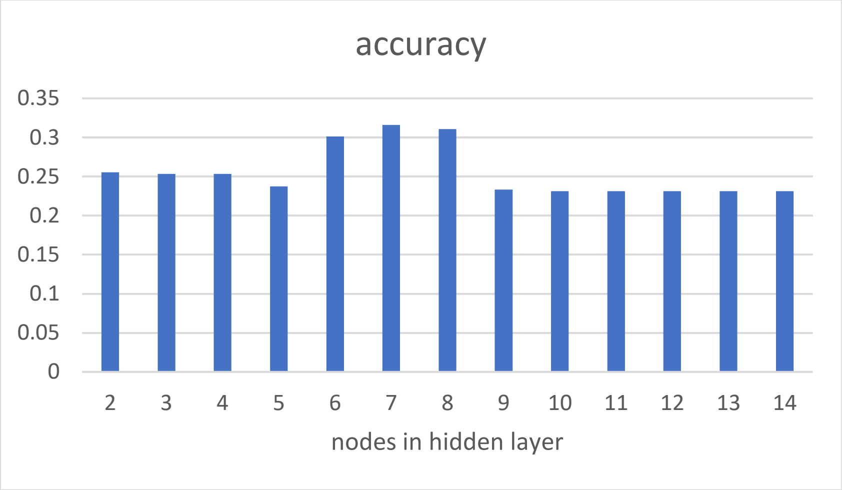
The graph is:

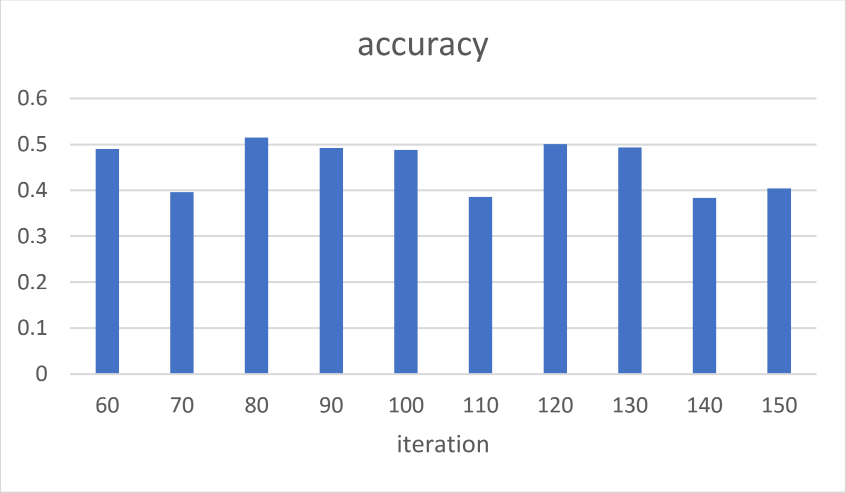


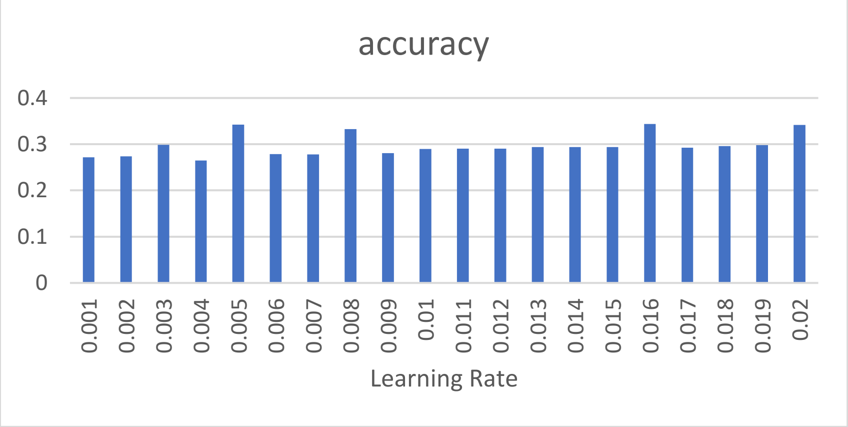


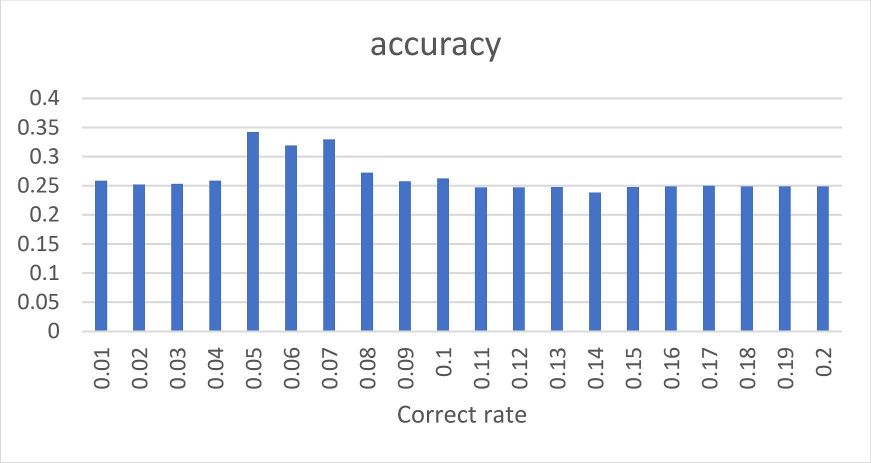


Other different parameters with different accuracy:


accuracy
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
2
3
4
5
6
7
8
9
10
11
12
13
14
nodes in hidden layer

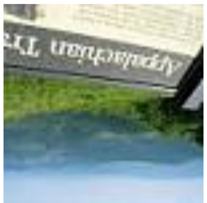

accuracy
0.6
0.5
0.4
0.3
0.2
0.1
0
60
70
80
90
100
110
120
130
140
150
iteration

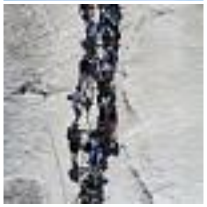

accuracy
0.4
0.3
0.2
0.1
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
0.011
0.012
0.013
0.014
0.015
0.016
0.017
0.018
0.019
0.02
Learning Rate

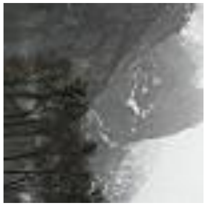


Therefore, we choose the number of nodes in the hidden layer 7, the size training set 3000, iteration = 80, learning rate = 0.005 and correct rate 0.05.

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







Maybe there are big grey/white or low contrast areas in these pictures.

Reference:

[1] https://en.wikipedia.org/wiki/AdaBoost

[2] http://blog.csdn.net/zjsghww/article/details/71485677

[3] http://www.cnblogs.com/Finley/p/5946000.html

[4] http://blog.csdn.net/miangangzhen/article/details/51281989

[5] http://blog.csdn.net/ly_ysys629/article/details/72842067

[6] https://ask.hellobi.com/blog/guodongwei1991/6709