1. **Introduce how you implement each classifier.**

**K-NEAREST NEIGHBOR:**

The intuition behind k-nearest neighbour is that the neighbours of a certain sample will mostly possess the characteristics of the sample. So in our case we seek to find out the nearest ‘k’ images in the training data with respect to the image in the test set. We compare the RGB values of each pixel of an image in the test data to all the corresponding RGB values of each pixel of the images in the training data (i.e) we compare the first pixel of the training image to the first pixel of the test data and we do this for all sixty four pixels, the distance between them is found using Euclidean distance formula, we then take the ‘k’ closest neighbours considering the distance. A max voting is done among the ‘k’ neighbours and we choose the orientation with max votes. The same is repeated for all the images in the test set. So in k-nearest neighbour algorithm we do not actually train the model but compare a single datum in the test set with all the data in the training set.

**RANDOM FOREST:**

A random forest has multiple decision trees. Each decision tree classifies the test data set into one of the orientations-0, 90, 180 and 270. Using max voting of the decision trees, we assign the orientation for the test images.

Random forest has many parameters:

n\_trees -number of trees in a forest

max\_depth- maximum depth of each tree , as the depth increases the decision tree may over fit but since there are many trees in a Random forest, overfitting does not occur

s\_size -number of samples used by each tree

min\_leaf -minimum number of samples required in each node before it is split.

We construct n\_trees number of trees with best split, each with maximum depth max\_depth and minimum number of leaves min\_leaf, s\_size number of samples that are sampled randomly. The number of features used in each tree is sqrt of the total number of features in the dataset.

**ADABOOST:**

The aim of adaboost is to boost the accuracy obtained from multiple weak classifiers. The resultant classifier will have an accuracy more than that of the individual classifiers.

Each classifier is a decision stump. A decision stump has only one test condition.

We implement 12 combinations:

0vs90, 0vs180, 0vs270 , 90vs0, 90vs180, 90vs270, 180vs0, 180vs90, 180vs270, 270vs0,270vs90,270vs180

For each combination above do below: (example taken combination 90 vs 180):

Repeat below until we get 50 decision stumps or weak classifiers:

Pick two random features say red11 and blue 11. Subtract the two features. If less than 0 then 90, if greater than equal to 0 then 180 degrees. We keep a split at 0 for all weak classifiers. Test if the number of samples for 90( less than 0 )is more than 7000 (more than half of 9244) and sum of number of samples of 90 and 180 together is more than 9244 (i.e. the sum should be more than half of 9244+9244(there are total 9244 samples in each 90 and 180 orientation)). If yes then consider red11, blue11 as one classifier.

We find 50 classifiers for each 90 vs 180, 90 vs 270 , 90 vs 0. Similarly for 0 vs 90, 0 vs 180, 0 vs 270, 180 vs 0, 180 vs 90 , 180 vs 270, 270 vs 0, 270 vs 90 and 270 vs 180. (takes about 15 mins)

We are trying all combinations i.e. even though I am doing a 90 vs 180. We do a 180 vs 90 again for two reasons:

1) There are no weak classifiers that classify both classes i.e. more than 7000 +7000 out of 9244+9244. Actually even for one class, there is only 8000 maximum. Hence I have adopted classification of one class at a time. – so a total of 12 adaboost.

2) We get three checks on each y label 0, 90 180 and 270. The accuracy not only increases but also stays constant- it is between 68 and 70 always. If we allow the classification on only one variation i.e. only 90 vs 180 and not do 180 vs 90 and at the same time choose weak classifiers split that have least disorder for 90 and 180 – the accuracy varies much more extensively from 64 to 69 percent.

3) When doing adaboost for say 90 vs 180- The weak classifier however will predict either 90 or 180 as ouput orientation, but the train dataset for the 90vs180 adaboost will have samples with orientation 90 only. This way the classifers of 90vs180 learn only about 90deg. While 180vs90 learn about 180 orientation only.

Adaboost implementation:

We combine each 50 classifiers to give one strong classifier through adaboost

Initialize the weights of the data samples to 1/N.

For each of the 50 weak classifiers:

If the predicted orientation is not same as the actual orientation then error is increased by the weight of the incorrect samples. We use data stored as dictionary to do this, so it is runs faster.

The weight of the correct samples are reduced by error/(1-error)

The weights of the samples are then normalized.

Initialize the weights of the classifier to (log(1-error)/error)

We store weights of the classifiers in the form eg. (red11,blue11)🡪 1.23 in the model file passed in the command line.

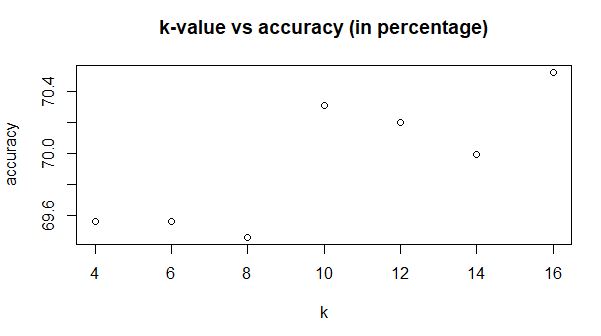
Store the weights of the classifier in the model file passed in the command line.

Testing:

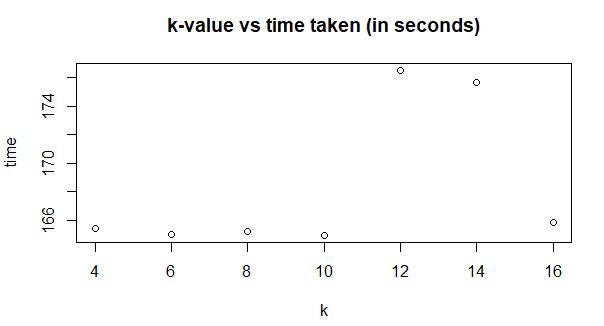
1. The weights from the model file passed in the test command line will be picked. We simply subtract the features eg. red11 and blue11 and check if it is less than 0 or more than 0. The weighted results for all orientations are compared and the orientation with maximum value is assigned to the test data.
2. The adaboost\_output.txt is also generated with the images and their corresponding predicted orientation.
3. **Present neatly-organized tables or graphs showing classification accuracies and running times as a function of the parameters you choose.**

**K-NEAREST NEIGHBOR:**

**Parameter- k value**

****

In the above plot, we can see that the accuracy percentage is lower when the k-value is less than eight and the maximum accuracy is obtained with a k-value of sixteen.

****

We can infer from the above plot that the time taken by the classifier with respect to k-value is more or less the same except for lower values of k but for higher values of k =12,14,16 the time taken increases in comparison to the lower values of k.

**RANDOM FOREST:**

|  |  |  |
| --- | --- | --- |
| Number of trees: | other parameters fixed s\_size=100,max-10,min-10 |  |
|  |  |  |
| n\_trees | Time taken (mins) | Accuracy |
| 100 | 12.3 | 60.76 |
| 500 | 30 | 61.08 |
| 700 | 45 | 62.67 |
| 900 | 78 | 63.5 |

With increase in number of trees, the accuracy and time taken increases

|  |  |  |
| --- | --- | --- |
| Depth: | other parameters fixed s\_size=100, n\_trees=300,min\_leaf=10 |  |
|  |  |  |
| depth | Time taken (mins) | Accuracy |
| 12 | 4.8 | 60.55 |
| 10 | 4.5 | 59.49 |
| 8 | 4.1 | 58.96 |
| 6 | 3.97 | 58 |

The accuracy increases with increase in depth and so does the time taken.

|  |  |  |
| --- | --- | --- |
| Min Leaves | other parameters fixed s\_size=100, n\_trees=300,depth=10 |  |
| min leaves | Time taken | Accuracy |
| 12 | 4.3 | 61.18 |
| 10 | 4.43 | 60.76 |
| 8 | 4.5 | 58.43 |
| 6 | 4.72 | 61.18 |

With increase in number of min leaves the accuracy increases

**ADABOOST:**

**The parameters we have chosen –number of weak classifiers:**

|  |  |  |
| --- | --- | --- |
| Number of decision stumps (weak classifiers) | | |
|  | | |
| Number of stumps | Time taken (mins) | Accuracy |
| 65 | 300 | 69 |
| 50 | 120 | 70.3 |
| 40 | 115 | 69 |
| 30 | 60 | 65 |

-With the increase in the number of stumps, the time taken increases. The accuracy on the other hand increases at first but then decreases beyond certain value. i.e. it increases till number of stumps=50 but decreases at 65.

-Also, in order to gain a 1 percent increase in accuracy the time taken increases 3 times. This is usually not preferable since for a small amount of accuracy the time taken is too high.

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

**K-NEAREST NEIGHBOR:**

|  |  |
| --- | --- |
| Training data size | Accuracy (in percent) |
| 18488 | 69.57 |
| 9244 | 67.87 |
| 4622 | 68.29 |
| 2311 | 65.85 |

The above table intends to compare the accuracy percentage while decreasing the training data size. The decrease in accuracy with decrease in training data size is not a surprise as that is what is generally expected. Hence we can conclude that the accuracy of a classifier is directly proportional to the size of the training data.

**ADABOOST:**

|  |  |  |
| --- | --- | --- |
| Number of samples: | |  |
|  |  |  |
| Number of samples | Time taken (mins) | Accuracy |
| 36976 | 20 | 69.5 |
| 18488 | 7.08 | 67.66 |
| 9244 | 3.36 | 68.29 |
| 4622 | 1.62 | 68.29 |
| 2311 | 0.9 | 67.55 |

The accuracy eventually increases with the increase in datasize along with the time.

**RANDOM FOREST:**

|  |  |  |
| --- | --- | --- |
| Number of samples: | other parameters fixed n\_trees=300,max-10,min-10 | |
|  |  |  |
| s\_size | Time taken (mins) | Accuracy |
| 50 | 2.4 | 57.69 |
| 100 | 4.5 | 61.9 |
| 200 | 16.99 | 61.4 |
| 300 | 62.13 | 64.16 |

The accuracy eventually increases with the increase in datasize along with the time.

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

**Incorrectly classified examples:**

7600713030- This picture is that of a light coloured waterfall.

9449235539- This picture has a dark sky and a lighter coloured land

3324514121- The picture is a bird with some lighter colours at 90 deg

**Correctly classified examples:**

10164298814, 11185093106, 180246673- are all landscapes with a clear light blue sky

There are patterns to the errors.The images with darker skies, or not so distinct sky with a lighter coloured land is not classified correctly. Also, images with birds or just a thing with some colours spread everywhere are not classified correctly. Almost all images are classified based on where the lighter colours lie. The model expects an image at 0 degrees to have lighter colours on the top of the image.

**5) Which classiers and which parameters would you recommend to a potential client?**

For the "best" classifier we used an ensemble created using max voting on the results from the other three models. This gives us an accuracy of 69.89%, even though this is lower than the accuracy of k-nearest this will still be our best classifier because the max voting ensemble greatly reduces misclassification. If you consider this as a binary classification example the max voting ensemble will reduce the false positives. Also the max voting ensemble will not suffer from the biggest drawback that the k-nearest faces (i.e) overfitting.

Hence the “best” classifier will be the classifier we would recommend to a potential client considering all of it’s advantages.