Assignment - 4

Course : CSCI - 551 Elements of Artificial Intelligence

Submitted by:

Charith Musku

Bhargav Kandlagunta

Bhagya Reddy

1.K- Nearest Neighbours Algorithm

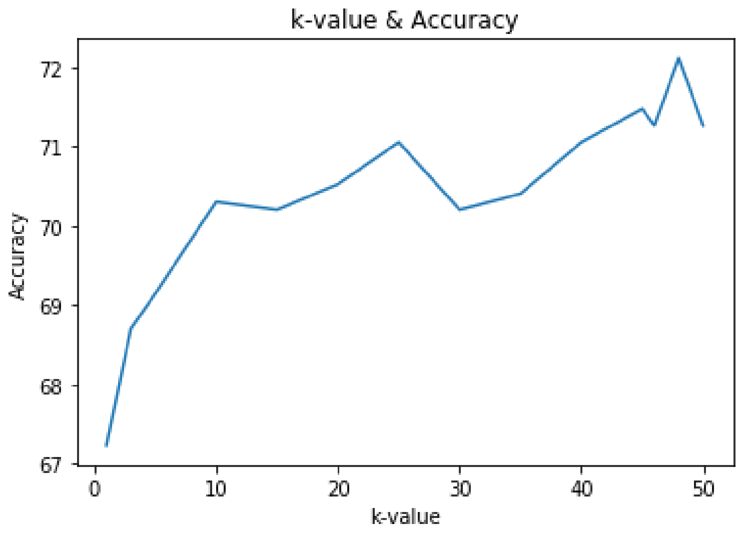
KNN algorithm is one of the simplest classification algorithm. KNN is a non-parametric, lazy learning algorithm. So for this, we have taken the data from the files given as numpy arrays namely x_train, y_train, x_test, y_test. Then, we have used the Euclidean distance as the distance metric for calculating the distances between the points.

For each test instance in x_test the distance to each of the x_train is calculated and stored into a matrix of size n*m where n is the length of x_test and m is the length of x_train. After we found the distances we sorted each row in the matrix using argsort(). Depending on the k value chosen we selected the initial k distances in each row of the sorted distances.

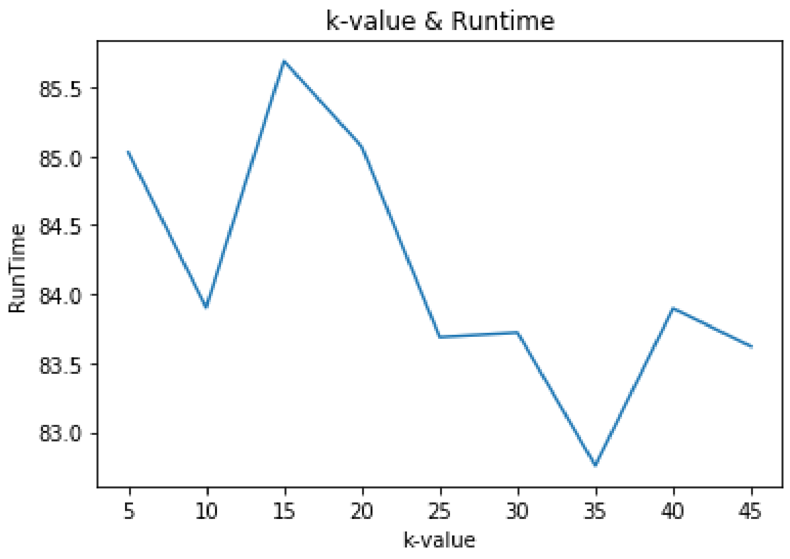
From the selected first k nearest neighbours, we chose the most common label and predict that as the label for the test instance and append it to the predictions array. The accuracy score gives the accuracy obtained.

We have obtained the maximum accuracy at k=48 i.e., 72.11%

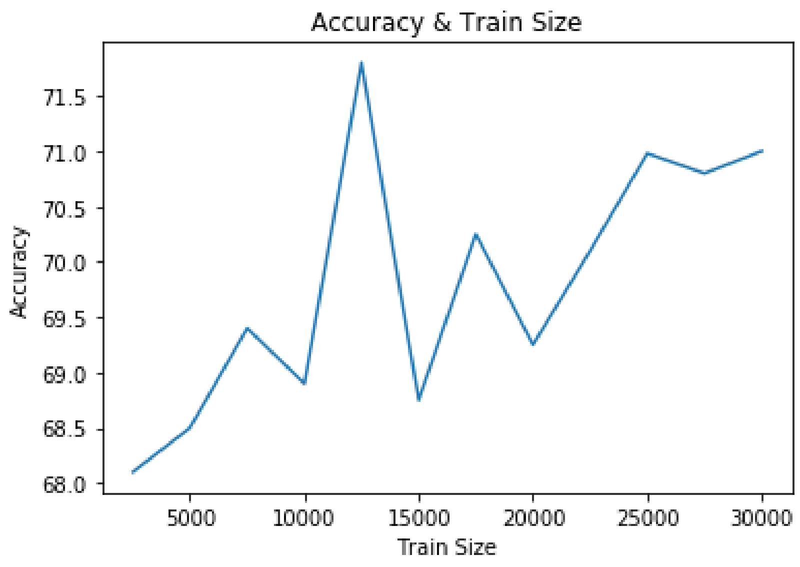
Graph 1: K-value vs Accuracy



Graph 2: Runtime plot



Graph 3: Accuracy vs training size



2.AdaBoost Algorithm

To handle mutli class classification problem we implemented a slightly different approach from that of binary classification adaboost algorithm.

Algorithm :

- Initialize weights as '1/n' for each of the data point(image in this case), where n is the total number of training images.
- We need to consider some weak classifiers as decision stumps to build decisions and then update the weights corresponding to the error in each iteration. Here we are comparing pixel values at two indices randomly selected in each image sample as decision stump.
- For each of the decision stump we consider :
    - The whole training images are split based on the decision stump.
        - One, if the selected first pixel values is greater than or equal to second one.
        - Two, if it is the other way.
    - Then for each of the split the decision prediction is taken as the mode of orientation of all the images in the split.
        - Suppose if the decision stump is based on pixel values at 10,81 indices.
        - Then training images set s split into two sets based on the above pixels.
        - And if 90 is the orientation that is most frequent in first split, then rest of the image samples in the set with different orientation are considered as errors.
    - Error is calculated based on the number of mis interpreted image based on the predictions set for both categories.

    - Error is the summation of all the weights corresponding to each mis predicted image.
    - A constraint is put on the error to be less than 0.75 because of the four different classes. If this constraint is not meant, current hypothesis/decision stump is discarded.
    - Then alpha value is calculated based on the error found in the previous step using :
        - a = log((1-error)/error) + log(K-1)

      where K is numbber of classes, here K = 4
    - Then the weights for all the mis interpreted images have to be updated, to have higher weights for the next hypothesis.
        - w_i = w_i * exp(a)
    - Now the weights are to be normalized to be used for the next iteration.
    - Current hypothesis and weight are appended to dictionaries to be stored in model file later on after all the iterations are done.

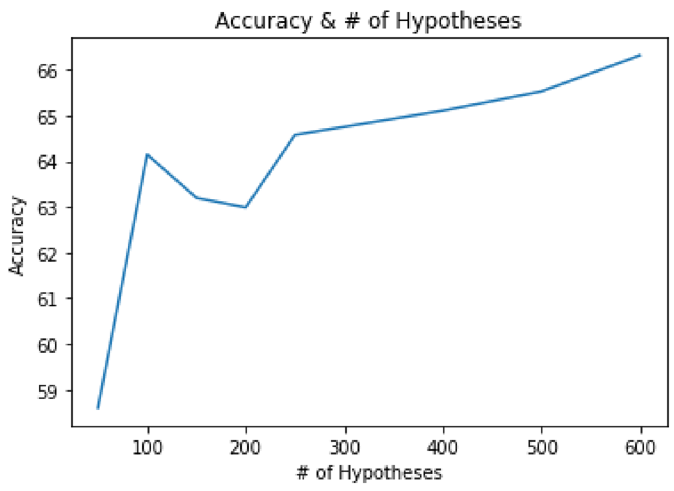
Testing :

- Hypotheses and weights stored for adaboost are loaded from adaboost_model.txt file
- For each test image
    - Initialize a prediction vector [0,0,0,0], each value corresponding to chance of current image falling into [0, 90, 180, 270] classes respectively.
    - For each hypothesis
        - The prediction is made based on the hypothesis.
        - Weight corresponding to the current hypothesis is updated to the initial prediction vector.
        - The final prediction is based on the maximum of the values in prediction vector corresponding to each class.

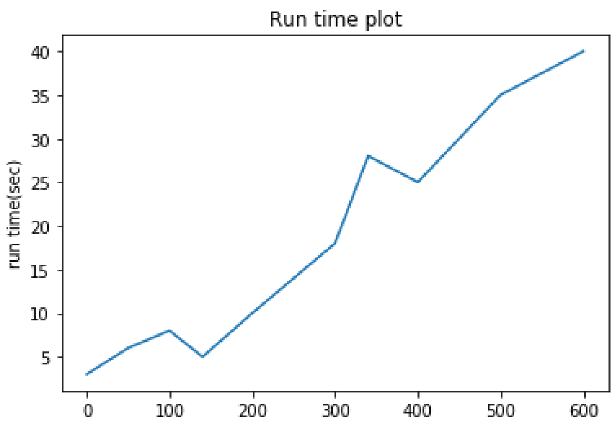
Reference :

- We have implemented multiple classification problem for adaboost using an algorithm proposed in one of the research papers. We made use of the weight, error calculation and the logic for handling multiclass.
- [Algorithm] (https://web.stanford.edu/~hastie/Papers/samme.pdf)

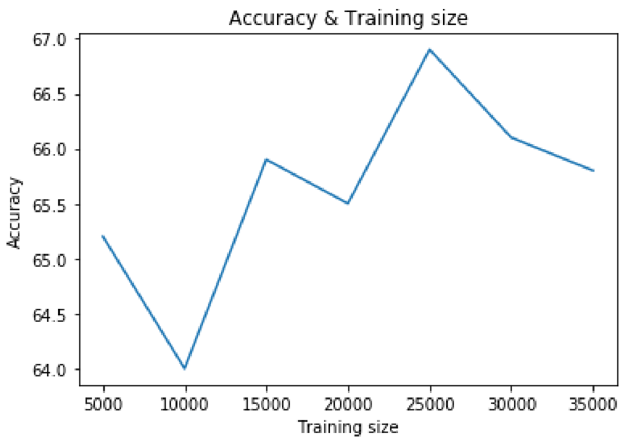
Graph 1: # of Hypothesis vs Accuracy



Graph 2: Runtime plot



Graph 3: Accuracy vs training size



3.Random Forest Algorithm

For decision tree, we have used an ensemble learning approach, where we have built random decision forests and using the max voting label out of all to predict the output.

Logic:

- By following the thumb rule for random forests i have considered square root number of features. So since we have 192 features in the form
- of R, G, B for each image, i have randomly generated 14 features and considered them to build each tree.
- I am calculating the entropies for each node at the median value of that feature, to pick up the best node to split at any point of time.
- Using gini entropy to calculate this. Splitting at median threshold so as to have maximum possible data left for further lower part of the tree.

Implementation:

- I have used a binary tree to build the random forests. With Tree as a main class and Node as sub class.
- Recursively built each random forest, by calling the bestsplit method, which returns the feature to split at, which gives max entropy.
- This helps us with runtime efficiency while both training and testing.

Parameter Tuning:

- When i tried to tune for depth cutoff, n=11 seemed to be stable, the peak point. > 11 it seems to overfitting and thus performing worse.
- For number of trees, more the trees better was the accuracy. But it seemes to stabilize after n= 60.

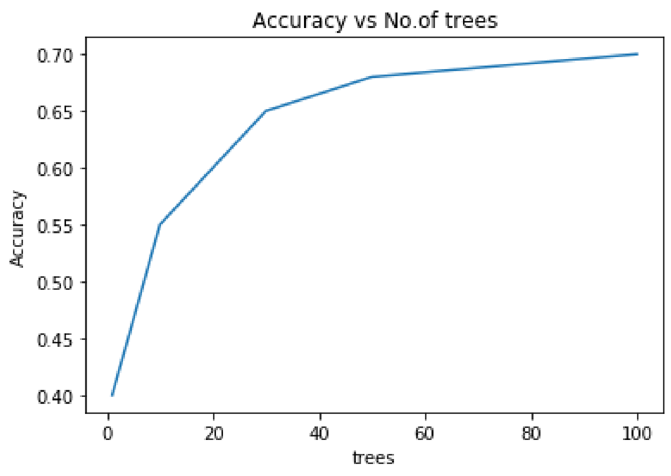
Dynamic depth cut-off:

- Later on i have followed a technique, where I am constantly checking if a node predicts a particular label with at least 90% confidence, then I am stopping the splitting there. Thus, it is a dynamic cutoff.

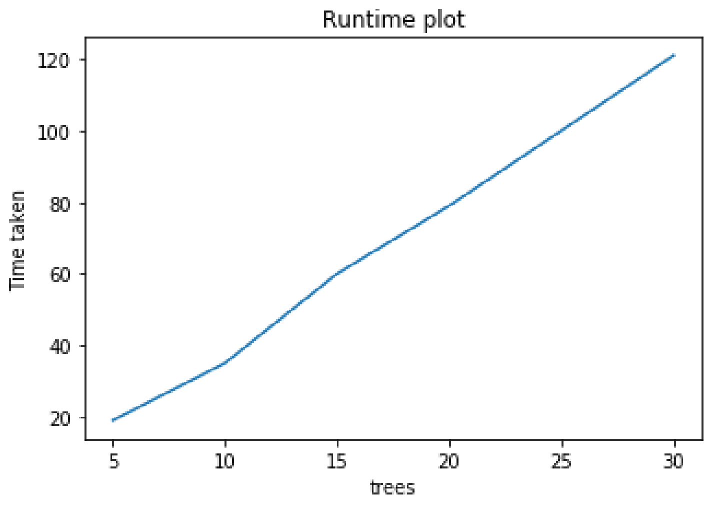
Result:

- We have got a final test accuracy of 69.74% for the given test data set. Following are the final parameters considered:
    - Depth: Dynamic (as mentioned above)
    - No of trees = 100
    - Split value = Median of feature value, in the data remaining at that node.

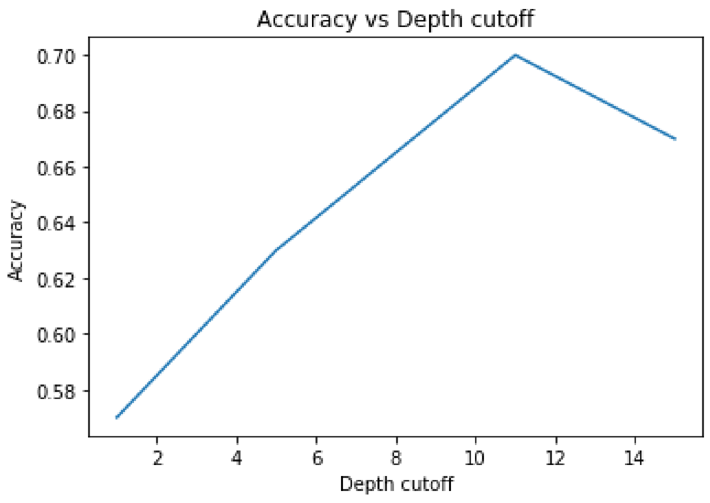
Graph 1: No.of trees vs Accuracy



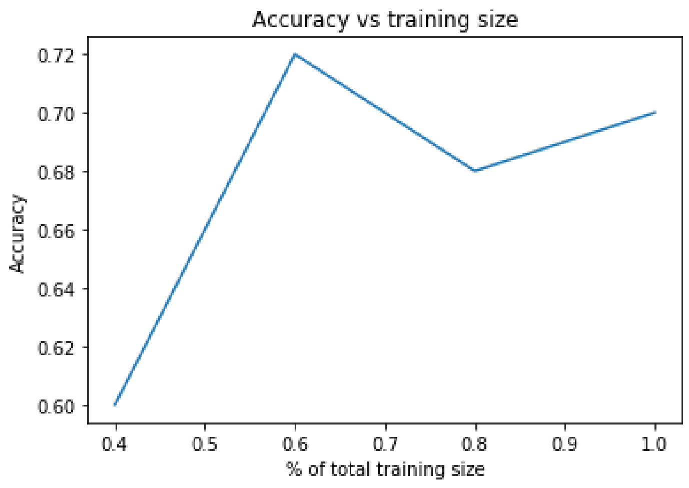
Graph 2: Runtime plot



Graph 3: Accuracy vs Depth cutoff



Graph 4: Accuracy vs Training size



4. Best Model

- Since kNN and Random Forests were better performing, out of the 3 models, we tried an ensemble of both but it did not perform as expected.
    - If we had the probabilities of predictions for 4 labels, ensemble approach would have worked in that case.

    - But considering we had only 4 labels, we needed at least 4 good performing models other than kNN and Random Forests, with more training data for this approach to have worked.
- For this particular problem here kNN performed better, with an accuracy of 72%
- But in general we believe that Random forests would be more accurate given more training data to learn and with better parameter tuning.
- Other than these algorithms, Convolutional Neural Networks would be a better choice for image classification any day. But it is difficult to implement it from scratch, unlike above models.

5. Sample Predictions:

Correctly classified examples:

- These are the examples of correctly predicted images. As per our interpretation, the images which had uniform background like above were most correctly predicted. Most the images with had common sceneries like blue sky towards the top of the image were easily predicted since they are the most common in the training examples.

Incorrectly Classified:

- Here as seen in the first image, our algorithm correctly predicts it as 0 degrees, but the actual test label was given as 180. So, there are few images with noise in the given data.
- The images taken in an angle were very difficult to predicted. The model was mostly getting confused between 180, 270 degrees images.

•