Introduction to Machine Learning *Phani Teja Kesha,KP38691*

HW1: Decision trees and KNN

1.

2.

3. In class, we looked at an example where all the attributes were binary (i.e., yes/no valued). Consider an example where instead of the attribute "Morning?", we had an attribute "Time" which specifies when the class begins.

   (a) We can pick a threshold τ and use (Time $< \tau$)? as a criteria to split the data in two. Explain how you might pick the optimal value of τ.

   The Threshold value must be picked in such a way that the data is equally distributed. Consider the time to be a value between 0 to 24.For example we can consider the threshold value τ as 12,where the data is split based on the equations τ >12 or τ <12. By splitting like this the data is distributed into two half's and all data is treated equally.If we choose a threshold value that does not equally divide the data , then some of the data might be intrepreted in the wrong manner and the accuracy of the predictions will be decreased.

   (b) In the decision tree learning algorithm discussed in class, once a binary attribute is used, the subtrees do not need to consider it. Explain why when there are continuous attributes this may not be the case.

   In the case of continuos attributes for decision tree all the subtrees are needed to be considered. This is because in binary attribute the data is aldready discrete with just two values where as in continuous we use a threshold value to convert the continuous data into two parts which gives the output in binary attributes . Due to this conversion between continuous to discrete the data might lose some feautures, so we need to consider all the subtrees to be considered to predict the value of the test data .

4. Why memorizing the training data and doing table lookups is a bad strategy for learning? How do we prevent that in decision trees?

   If we memorize the training data , then the machine will not learn anything and will just look like a lookup table. For example when ever we use the testing data, the machine will automatically try to find the output from the training data which is exactly same as the test data. This is called as overfitting. Due to this the machine will never try to learn anything from the training data so this is not a good strategy for learning. To prevent this from decision trees we create the tree in such a way that it is not overfitting and some training data is used as a test which is called as the validation stage,In this stage the tree will not learn but will check the liability of the tree. After the learning is completed then the test data is used.

5. What does the decision boundary of 1-nearest neighbor classifier for 2 points (one positive, one negative) look like?

> The decision boundary may depend on the distribution of the positive and negative data . If all the points of same classifier can be divided into halfs using a line ,then the decision boundary is in the form of a straight line .The straight line can be drawn even if some of the classifiers fall under the wrong category. The boundary will not try to overfit . If the data cannot be divide into two parts then the space is divided into many boundaries . These boundaries act as the decision block and the decision is given by considering all the boundaries of the space.

6. Does the accuracy of a kNN classifier using the Euclidean distance change if you (a) translate the data (b) scale the data (i.e., multiply the all the points by a constant), or (c) rotate the data? Explain. Answer the same for a kNN classifier using Manhattan distance[1].

> a)If we translate the data the accuracy of the kNN classifier using the Euclidean distance will not change because if we translate all the data, the distance between the test points and training points will increase in the same ratio. That means the smallest distance will be from the same test point.
> Example :Training data is(1,1,1) and (2,2,2) .Testing point is (0,0,0) .If training data is translated by 2,training data will become (3,3,3) and (4,4,4). The distance between testing and training data will become 3,8 to 27,64 which makes the data (1,1,1) the nearest data point.
> b)If we Scale the data the accuracy of the kNN classifier using the Euclidean distance will not change because if we Scale all the data, the distance between the test points and training points will increase in the same ratio. That means the smallest distance will be from the same test point.
> Example : Training data is(1,1,1) and (2,2,2) .Testing point is (0,0,0) .If training data is scaled by 2,training data will become (2,2,2) and (4,4,4). The distance between testing and training data will become 3,8 to 8,64 which makes the data (1,1,1) the nearest data point.
> c)If we Rotate the data the accuracy of the kNN classifier using the Euclidean distance will not change because if we Scale all the data, the distance between the test points and training points will increase in the same ratio. That means the smallest distance will be from the same test point.
> Example : Training data is (3,4) test data is (0,0) . Distance is 5,if we rotate the test data by 30 degree, the point is (4.6,1.96) . The distance is again 5. That means the distance is not effected by the rotation of data points.

> In the case of Manhattan DIstance the accuracy will not change if we translate or change the data but will change if we rotate the data .

7. Implement kNN in Matlab or Python for handwritten digit classification and submit all codes and plots:

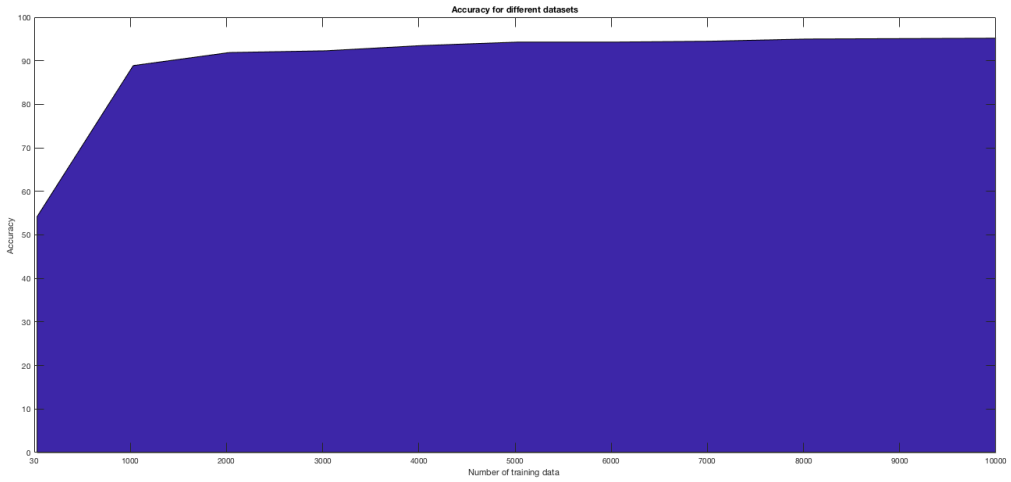
   (a) Download MNIST digit dataset (60,000 training and 10,000 testing data points) and the starter code from the course page. Each row in the matrix represents a handwritten digit image. The starter code shows how to visualize an example data point in matlab. The task is to predict the class (0 to 9) for a given test image, so it is a 10-way classification problem.

   (b) Write a Matlab or Python function that implements kNN for this task and reports the accuracy for each class (10 numbers) as well as the average accuracy (one number).
   [acc acc_av] = kNN(images_train, labels_train, images_test, labels_test, k)
   where *acc* is a vector of length 10 and *acc_av* is a scalar. Look at a few correct and wrong predictions to see if it makes sense. To speed it up, in all experiments, you may use only the first 1000 testing images.

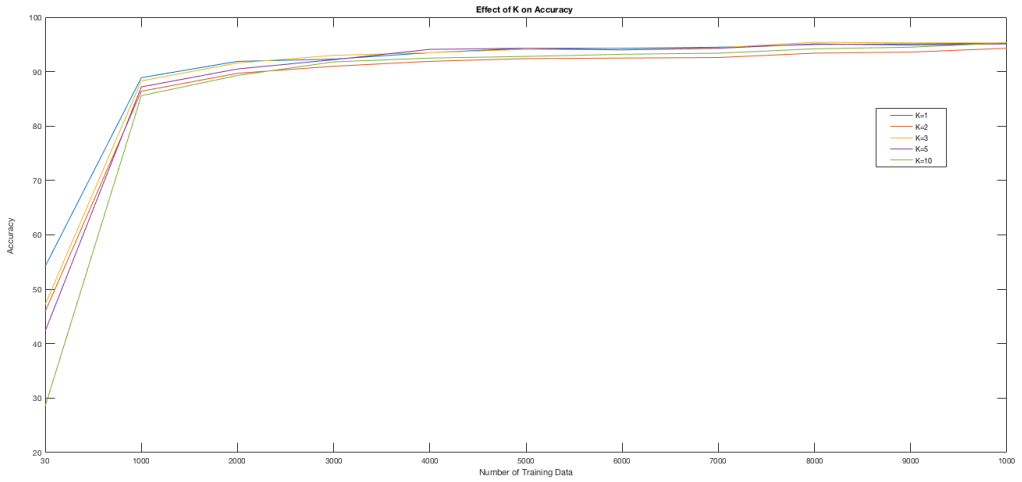
[1] `http://en.wikipedia.org/wiki/Taxicab_geometry`

For K=5 and for 1000 Testing data
Accuracy is 97.30

(c) For $k = 1$, change the number of training data points (30 to 10,000) to see the change in performance. Plot the average accuracy for 10 different dataset sizes. You may use command *logspace* in matlab. In the plot, x-axis is for the number of training data and y-axis is for the accuracy.



(d) Show the effect of k on the accuracy. Make a plot similar to the above one with multiple colored curves on the top of each other (each for a particular k in [1 2 3 5 10].) You may use command *legend* in matlab to name different colors.



(e) Choose the best k for 2,000 total training data by splitting the training data into two halves (the first for training and the second for validation). You may plot the average accuracy wrt k for this. Note that in this part, you should not use the test data. You may search for k in this list: [1 2 3 5 10].

K=5
K and Accuracy
K=1 Acc=0.8630
K=2 Acc=0.8510
K=3 Acc=0.8640
K=5 Acc=0.8650
K=10 Acc=0.8350