Introduction to Machine Learning *Sowmya Ramapatruni, BG80177*

HW1: Decision trees and KNN

1. 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 optimal value of τ would be 12. If the Time is less than 12, we consider it as Morning and If it is greater than 12 , we consider it as not Morning.

   (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 case of continuous values, we cannot split the data into subtrees as there will be no condition to consider. For example ,If we have a training set with attribute 'Marks'
   Marks: 60,70,80,90,100
   we end up creating 5 different paths to the leaf without splitting them.

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

   Memorizing the data and doing lookups is a bad strategy because doing so will not generalize the model. It leads to over fitting , as in performance on training data will be more than that of test data.
   Another drawback would be the complexity , the more data is stored, the more complex the model will be. For instance in case of decision tree , each training set will have its own path to the leaf node and the size of the tree will increase monotonously with training data.
   In decision trees , this can be prevented by choosing the best attribute which splits the whole data into two positive and negative examples. The correct classification will give the results with few tests. The best attribute can be calculated with the help of entropy and information gain concepts.
   The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).
   1. Calculate entropy of the target.
   2. The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.
   3. Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.
   4. A branch with entropy of 0 is a leaf node.
   5. A branch with entropy more than 0 needs further splitting.

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

> The decision boundary of 1-nearest neighbour classifier for 2 points will be the line passing through the midpoint of both the points. In geometrical terms , it should be a perpendicular bisector which is at equal distance from both the points.

4. 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].

> In case of Euclidean distance,
> a) when the data is transalted, the shortest distance point will always remain same.There will be a change in the magnitude of the distance, but shortest point will be the same.
> for example,
> A(1,2) B(2,3) are two data points After translating the points by 3 A‘(4,5) B‘(5,6)
> Distance from the origins of each of the points are:
> A $=\sqrt{5}$ B $=\sqrt{15}$ A‘$=\sqrt{41}$ B‘$=\sqrt{51}$
> Even after the translation , distance of A is shorter than B from the origin.
> b) Similarly when we scale the data by 3 A‘(3,6) B‘(6,9)
> A‘ $=\sqrt{45}$ B‘ $=\sqrt{117}$
> Distnace of A to origin is still shorter than B.
> c) Even when we rotate all the data points by similar angle, the shortest distance point will still be same .
> The same will be applicable for Manhattan distance, because formula to calculate Manhattan for two points A(a,b) B(c,d) is | a-c | + |b-d |.
> So the shortest distance point will always be the same , even though there will be a change in the magnitude

5. 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.

   (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.

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

(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].