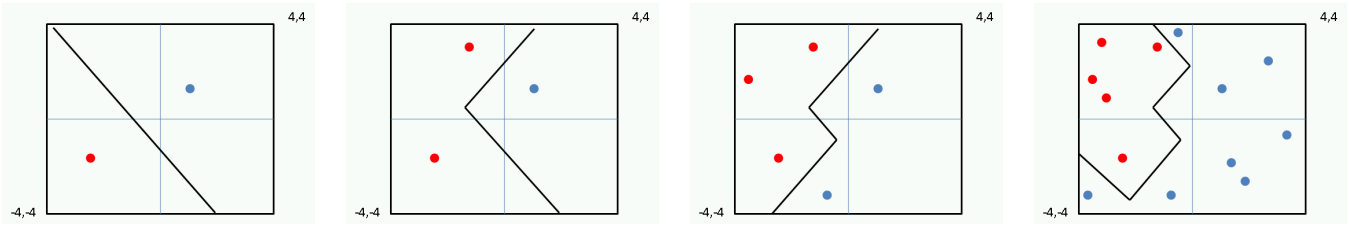
10-701/15-781: Homework 1 Solutions

Eric Xing, Tom Mitchell, Aarti Singh
Carnegie Mellon University
Updated on January 12, 2010

1 KNN and Decision Trees [35 pt, Amr]

1. **(10 points)** For each of the following figures, we are given a few data points in 2-d space,each of which is labeled as either positive (blue) or negative (red). Assuming that we are using L2 distance as a distance metric, draw the decision boundary for 1-NN for each case. For example, the decision boundary for the dataset in 1.b is given in figure 2.a.



(a) (b) (c) (d)

2. **(3 points)** In class we have mentioned that NN is a *lazy* classifier that needs to store all training instances until test time. However, in this problem we were able to draw a decision boundary for the 1-NN classifier. If we decided to store this decision boundary instead of storing all training data, would that *always* result in an improvement in terms of storage (memory) requirement for this classifier? [Please answer in no longer than 2-5 sentences]

   solutions: In general, the answer is NO. Each data point, say a, will contribute, in the worst case, a line that bisects the line segment between a and the nearest point to a from the opposite class. Thus in the worst case, we get a complete Voronoi diagram which won't resent in any storage advantage.

3. **(2 point)** Decision trees are known as batch learners that require the availability of all training data to build the tree. Thus the arrival of additional training data needs to be handled carefully. Does KNN suffer from this problem and why?

   solutions: NO since there is no training in KNN, thus we just add the new data points to old ones. All computations are done at test time. Some of you argued that the decision boundary will change with the arrival of a new point, while this is true, it should be noted that we do NOT store or compute that decision boundary thus no processing is needed here.

4. **(7 points)** Now assume that given a dataset D you built a decision tree T. Later on, someone handed to you an extra training data D'. Describe briefly how could you augment T to get T_2, a decision tree for both $D + D'$. Clearly T_2 might not be as good as a decision tree built form scratch over $D + D'$, however, an answer that re-builds the decision tree from scratch would get **no credit**. [Please answer in 2 to 5 sentences]

   solutions: Classify D' using T which will result in storing each data point in D' into a leaf node of T. Now just apply your decision tree learner to the current state of the tree to further split leave nodes until the stopping criteria is reached or until you exhaust all possible

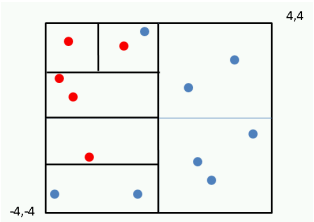
attributes. Note: you should keep track of the number of positive and negative data points at each lafe of T from D to be able to augment the tree in this way, however, I did not deduce points if you don't state that explicitly.

5. We would like to build a decision tree over the dataset in Figure 1.d. Each data item is described using two continuous attributes (x, y). One way of handing continuous attributes is discretization, here we will use binary-discretization and test if a given attribute is $\geq$ a given threshold which we will take to be the midpoint. For instance, in Figure 1.d, the range of both X and Y is [-4,4], thus the test at the first level of the tree is either $x \geq 0$ or $y \geq 0$. In the next level, we might test for $x \geq 2$,$x \geq -2$, $y \geq 2$, etc. In other words, we always consider the current range of both attributes in the training data at a given leaf node, and formulate a test that would divide this range into tow equal parts. For instance, if we apply this scheme to the data in Figure 1.b, we get the decision boundary in Figure 2.b (other solutions are possible as well).

   (a) **(2 points)** What would you expect the training error to be if we apply the above scheme to the data in Figure 1.d? and why?

   solutions: The training error is 0%, since we can always keep splitting each dimension until we surround each data point in the training set with its own rectangle and thus resulting in leaves with 100% purity in terms of class labels.

   (b) **(7 points)** To limit the size of the resulting decision tree, we will stipulate that any single attribute is tested at most twice on any path from the root of the tree to a a leaf node. With this restriction, draw the decision boundary of a possible decision tree over the data in Figure 1.d. you don't need to do any calculations, just draw the decision boundary that corresponds to a suitable decision tree.



   As the questions asked we only split attributes at mid-point. Some of you splitted an attribute at an arbitrary position. Indeed this might result in a simpler decision tree, however, it is more expensive to find such an optimal split as we have to test and compare multiple splitting point.

6. **(4 points)** Using the intuition you gained in this problem, state one advantage of Decision Trees over KNN and onde advantage of KNN over Decision Trees.

solutions: There are many. Here is one: it easier to apply KNN in situations where the data changes over time, while in DT we can trade off hypothesis complexity, and thus storage requirement, for accuracy, which in some cases, might result in better generalization.