Luke Jaffe
CS 6140
9/22/15

# HW1 Decision Tree, Linear Regression

**Problem 1.**
*Implemented in tree.py*

a. spam data
Usage: python tree.py -d 'spam'

The k-folds implementation is in kfolder.py. The KFolder class includes an optional normalization method.

Results of a sample trial with k = 10:
(tree depth 12, stop splitting at #elem < 100 or entropy < 0.3)
Training accuracy: 0.913547 correct
Testing accuracy: 0.900000 correct

b. housing data
Usage: python tree.py -d 'housing'

Results of sample trial:
(tree depth 4)
Training MSE: [ 15.80760926]
Testing MSE: [ 25.31984891]

**Problem 2.**
*Implemented in regression.py*

a. spam data
Usage: python regression.py -d 'spam'

The KFolder class in kfolder is also used here. There is also an evaluation class used to calculate error in evaluator.py.

Results of a sample trial with k = 10:
Training MSE: [ 0.10209151]
Training accuracy: 0.878024631731
Testing MSE: [ 0.1153544]
Testing accuracy: 0.874347826087

b. housing data
Usage: python regression.py -d 'housing'

Results of sample trial:
Training MSE: [ 24.47588278]
Testing MSE: [ 24.29223818]

**Problem 3.**

i. Given an arbitrary decision tree with repeated splits on feature f with threshold t, there exists an equivalent decision tree with only distinct splits on each path. Since the question is posed as having threshold t on feature f, we will assume a binary tree is implied. Let us suppose we have a node Q in our decision tree which contains the set of elements A. In order to split our node Q, we must partition A into two unique subsets B,C such that intersection(B, C) is null. For a feature f and threshold t, let us suppose that all elements in A with f < t are put in B, and all elements with f >= t are put in C. Now, we will make the same query (same f and t) on B. Since all elements with f < t are put in B, we can see that the two resulting subsets are B and the null set. Likewise, by performing the same query on C, we can see that the two resulting subsets are C and the null set. Further, we know that there can be no change in entropy before and after this repeated query, meaning that the split has no effect on the classification. No matter how many additional queries we insert between the original query and its repeat, the result is that same: the child nodes following the query will be the parent set and the null set. Thus, such a tree is equivalent to the same tree with no repeated queries.

ii.
a. Suppose we have a an arbitrary tree with unequal branching ratios throughout. Suppose the root of a tree R is connected to B child leaves (q1,q2,...,qB). If we can show that this tree can be represented as a binary tree for any value of B, we have proved by induction that a tree with arbitrary branching and depth can also be represented as a binary tree. To do this, we can take all the child leaves of R but one, and place them under a new root R1. Thus, by alternating through each child leaf (q1,q2,...,qB), we can build a binary tree with that leaf under R, and the rest under R1 (which is a child of R). If B > 3, we will perform this operation again, and so on, until the tree is binary. Thus, any non-binary tree can be transformed into a binary tree that implements the same classification.

b. We can split our our tree in two ways: in the first way, we take all the elements but one at a node Q, and put them under a new node Q1, such that the two children of Q are one element and Q1. By splitting at every node, we can perform exactly B-1 splits, meaning each element gets its own level for B levels. Alternatively, we can partition our elements evenly at each node, for a node Q with k elements, k/2 go left, and k/2 go right. We can see that the upper bound on this as a function of B will be $\log_2 B$. Therefore, our lower bound is $\log_2 B$, and our upper bound B.

c. By splitting with the first method, we can see that the number of nodes will be 2B-1, since each leaf element has its own parent node. Splitting with the efficient method also yields 2B-1 nodes (for a full tree). Therefore, our upper and lower bounds on the number of nodes as a function of B is 2B-1.

**Problem 4.**
Located in same folder as hw1_p4.pdf

**Problem 5.**
Located in same folder as hw1_p5.pdf

**Problem 7.**
Given two sets of vectors A and B, we will prove that they cannot be both linearly separable and have their convex hulls intersect. Let us suppose that both statements are true. In this case, there will be a d-dimensional hyperplane separating the two sets. Since we have supposed that statement two is true, that the convex hulls intersect, we know that this intersection must not be null. Let us consider the classification of a point in this intersection. This theoretical point lies in both A and B; but this is contradictory: If there is a d-dimensional hyperplane separating A and B, then this point must exist on *both* sides of the hyperplane. Since the point cannot exist in two locations simultaneously, we have proved our original claim by contradiction.