CS 495 – Assignment 10: ID3

This purpose of this assignment is to build familiarity with ID3, which can be thought of as the "generic" decision tree algorithm. Throughout this exercise, by "set" we really mean "multiset" (in a multiset an element can appear more than once). The target feature will always be categorical.

Entropy

Entropy is a measure of disorder (lack of uniformity) in a set. The entropy of a set *S* is defined as:

$$Entropy(S) = -\sum_{i=1}^{n} \Pr[i] \cdot \lg_2(\Pr[i])$$

…where 1 through *n* represent the different types of elements found in *S*, and $\Pr[i]$ is the probability of selecting an element of type *i* when picking an element uniformly at random from *S* (equivalently, the proportion of elements that have that type). For example, suppose we have a bag B with ½ oranges and ½ apples. Then the entropy of B is:

$$Entropy(B) = -\sum_{i=1}^{2} \Pr[i] \cdot \lg_2(\Pr[i]) = -(.5 \cdot \lg_2(.5) + .5 \cdot \lg_2(.5)) = -(-.5 - .5) = 1$$

Intuitively, if one selects an element uniformly at random, the entropy is the expected number of bits it would take to describe what type of element was selected (with an optimal encoding scheme). With apples and oranges in equal numbers, exactly one bit would be needed to identify a random fruit.

Try calculating the entropy of a bag containing 1/4 apples and 3/4 oranges. It should come out greater than 0 but less than 1. If you don't have a calculator with log base 2, you can still get it using the formula: $\lg_2(a) = \frac{\ln(a)}{\ln(2)}$, or $\lg_2(a) = \frac{\log(a)}{\log(2)}$.

Now try the extreme where the bag has only apples. Does the answer match the intuitive notion of entropy (the number of bits needed to describe which type of element got picked)?

One more: calculate the entropy of a set which has ½ pears, ¼ oranges, and ¼ apples. Note that the answer corresponds to the average word length of a binary string code where "0" is a pear, "10" is an orange", and "11" is an apple.

Information gain

The next step to understanding ID3 is the notion of information gain, which is a measure of how many bits of "disorder" are removed by splitting set S into two or more parts. Information gain is the entropy of S minus the entropy of each partition of S (weighted by its relative size). If S is partitioned into *m* disjoint sets $S_1, S_2, \ldots, S_m$ by feature *f*, then we define *InfoGain(S,f)* as:

$$InfoGain(S, f) = Entropy(S) - \sum_{i=1}^{m} \frac{|S_i|}{|S|} \cdot Entropy(S_i)$$

…where *f* is the feature we are splitting on, $|S|$ is the number of elements in S and $|S_i|$ is the number of elements in partition S_i., when splitting on feature *f*.

Suppose S is the set which had ½ pears, ¼ oranges, and ¼ apples. How much information gain would we get if partitioned it into two sets: one with all pears and one with ½ oranges and ½ apples?

In the context of ID3, pear / apple / orange would be target feature values. Actually, the algorithm only applies the notions of entropy and information to the target feature – not to descriptive features. ID3 builds each node of the decision tree by splitting the set on a descriptive feature. Whichever feature results in the best information gain is chosen for that node of the tree. The resulting fragments of the initial set are considered recursively one step further down the tree. Here is the pseudocode for ID3:

```
//Precondition: S is non-empty
ID3(TrainingInstances S, FeatureList descriptiveFeatures, Feature targetFeature) {
	if (the target value is the same for all instances in S) {
		return a leaf node labeled with that target value
	}
	if (descriptiveFeatures is empty) {
		return a node labeled with S.mostCommonValueOf(targetFeature)
	}
	Let f be the feature that maximizes: InfoGain(S, f)
	Node n = new Node that splits on feature f
	For each possible value i of feature f {
		Node t;
		Let S_i be the subset of S which has the value i for f
		if (S_i is empty) {
			t = new Node()
			t.label = S.mostCommonValueOf(targetFeature)
		}
		else {
			t = ID3(S_i, descriptiveFeatures with f removed, targetFeature)
		}
		Add t as a child of n, corresponding to the test "Does feature f = i?"
	}
	return n
}
```

Trace through the steps of ID3 to build a decision tree for the following dataset:

safety	**condition**	**buy**
low	poor	no
low	good	no
low	good	no
low	ok	no
med	good	no
med	ok	no
high	ok	no
high	ok	no
high	ok	no
high	ok	no
high	ok	yes
high	good	yes
high	good	yes

Continous descriptive features

For continuous descriptive features, all values for the feature under consideration are put into sorted order. The algorithm then considers any split point where neighboring instances have different target values. For example, if the target values are + and –, then the following (sorted) set of instances would have 4 possible split points to consider: – – – – – + + + + + – – – + + + –

Trace through the steps of ID3 again – this time on the following dataset which has only one continuous descriptive feature and one categorical target feature. You should end up with two internal nodes and three leaf nodes in your tree.

Width	**QCPass**
5.16	No
5.26	No
5.27	No
5.34	No
5.36	No
5.38	No
5.45	No
5.47	No
5.58	Yes
5.65	Yes
5.67	Yes
5.71	Yes
5.76	No
5.79	No
5.92	No
5.99	No