CMSC421 Fall 2005 Homework 5

December 19, 2005

1 *Problem 1: Decision Tree Learning (20 pts)*

(a) First we compute the information in the original data set. There are 6 positive ("Sail") and 8 negative ("Don't Sail") instances, so we have: $I(S) = -6/14 \log_2(6/14) - 8/14 \log_2(8/14) = 0.985$ bits.

If we use Outlook to subdivide S, the data set is divided into three sets. The sunny set has 2 positive and 3 negative instances; overcast has 4 positives; and rain has 5 negatives. The information of this split is $I(S, Outlook) = 5/14(-2/5 \log_2(2/5) - 3/5 \log_2(3/5)) + 4/14(-4/4 \log_2(4/4) - 0/4 \log_2(0/4)) + 5/14(-0/5 \log_2(0/5) - 5/5 \log_2(5/5)) = 0.347$ bits.

Applying the same technique to the EngineOK attribute, we get: $I(S, EngineOK) = 11/14(-6/11 \log_2(6/11) - 5/11 \log_2(5/11)) + 3/14(-0/3 \log_2(0/3) - 3/3 \log_2(3/3)) = 0.781$ bits.

The information gain associated with each attribute is the original data set's information minus the attribute's information: Gain(S,Outlook) = $0.985 - 0.347 = 0.638$ bits; Gain(S,EngineOK) = $0.985 - 0.781 = 0.204$ bits.

(b) To calculate the gain ratio, we have to compute the split information associated with each attribute. The outlook attribute produces three subsets containing 5,4 and 5 cases, so we have splitInfo(S,Outlook) = $-5/14 \log_2(5/14) - 4/14 \log_2(4/14) - 5/14 \log_2(5/14) = 1.577$ bits; splitInfo(S,EngineOK) = $-11/14 \log_2(11/14) - 3/14 \log_2(3/14) = 0.750$ bits.

The gain ratio is defined as GainRatio(S,Attr) = $\frac{gain(S,Attr)}{splitInfo(S,Attr}$. Therefore, we have GainRatio(S,Outlook) = $0.638/1.577 = 0.405$; GainRatio(S,EngineOK) = $0.204/0.750 = 0.272$.

(c) There are 3 children at the first level. The sunny branch requires a further split. For this branch, there's no single attribute that we can split on to complete the decision tree. The decision tree looks like Figure 1.

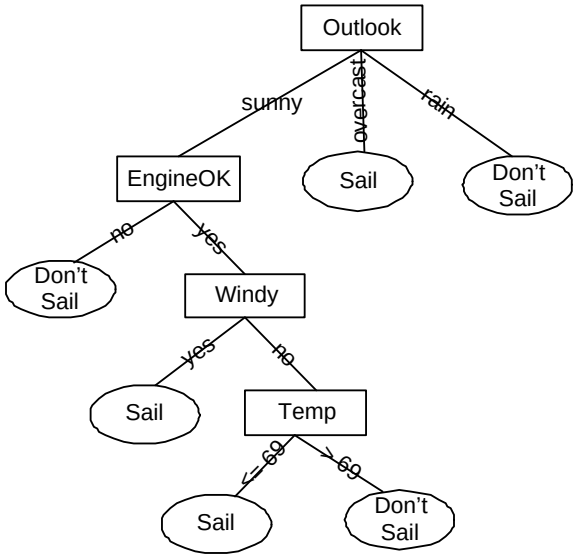

Figure 1: Decision Tree

2 *Problem 2: Version Spaces (20 pts)*

(a) G = {[any-type any-price any-size]}. The S set contains all of the conjunctive concepts that use the leaf-level attribute values. There are $5 \times 4 \times 4 = 80$ hypotheses in the initial S set. [chinese $$ big] is a representative member of the S set.

(b) After processing I1, a positive example with attribute values [chinese $ all-you-can-eat], the G set is unchanged, since {[any-type any-price any-size]} covers I1.

The only member of the initial S set that covers I1 is [chinese $ all-you-can-eat], so the other elements are deleted from the S set.

(c) G = { [asian any-price any-size] [any-type inexpensive any-size] [any-type any-price large]}

(d) 26. The easiest way to count this is that, given our positive example I1, are 3 possible types: any, asian and chinese, 3 possible prices: any, inexpensive and $ and 3 possible sizes: any, large and all-you-can-eat. This makes $3x3x3 = 27$ possible concepts that are consistent with the positive example. Of these, one of them is inconsistent with the negative example, [any, any, any].

(e)

- There is more than one possibility, and you should think of some of the others. The easiest example is being presented with [french,$$$$,petite] as a positive example.
- Again, there is more than one possibility. The easiest example is being presented with [chinese $ all-you-can-eat] as a negative example.
- The positive example [japanese, $$$$, tiny] would reduce the version space to the single concept [asian, any, any].
- The negative example [japanese, $, all-you-can-eat] would reduce the version space to the single concept [chinese, $, all-you-can-eat].

3 *Problem 3: Russell & Norvig exercise 20.3 (20 pts)*

This is a nontrivial sequential decision problem. It leads into general issues of statistical decision theory, stopping rules, etc. Here, we sketch the straightforward solution.

We can think of this problem as a simplified form of POMDP (Partially Observable Markov Decision Process). The "belief states" are defined by the numbers of cherry and lime candies observed so far in the sampling process. Let these be C and L, and let $U(C, L)$ be the utility of the corresponding belief state. In any given state, there are two possible decisions: *sell* and *sample*. There is a simple Bellman equation relating Q and U for the sampling case: $Q(C, L, sample) = P(cherry|C, L)U(C + 1, L) + P(lime|C, L)U(C, L + 1)$. Let the posterior probability of each h_i be $P(h_i|C, L)$, the size of the bag be N, and the fraction of cherries in a bag of type i be f_i. Then the value obtained by selling is given by the value of the sampled candies (which Ann gets to keep) plus the price paid by Bob (which equals the expected utility of the remaining candies for Bob): $Q(C, L, sell) = Cc_A + Ll_A + \sum_i P(h_i|C, L)[(f_iN - C)c_B + ((1 - f_i)N - L)l_B]$ and of course we have $U(C, L) = max\{Q(C, L, sell), Q(C, L, sample)\}$. Thus we can set up a dynamic program to compute Q given the obvious boundary conditions for the case where $C + L = N$. The solution of this dynamic program gives the optimal policy for Ann. It will have the property that if she could sell at (C, L), then she should also sell at $(C, L + k)$ for all positive k. Thus, the program is to determine, for each C, the threshold value of L at or above which she should sell. A minor complication is that the formula for $P(h_i|C, L)$ should take into account the non-replacement of candies and the finiteness of N, otherwise odd things will happen when $C + L$ is close to N.

4 *Problem 4: Russell & Norvig exercise 20.11 (20 pts)*

XOR (in fact any Boolean function) is easiest to construct using step-function units. Because XOR is not linearly separatable, we will need a hidden layer. It turns out that just one hidden node suffices. To design the network, we can think of the XOR function as OR with the AND case (both inputs on) ruled out. Thus the hidden layer computes AND, which the output layer computes

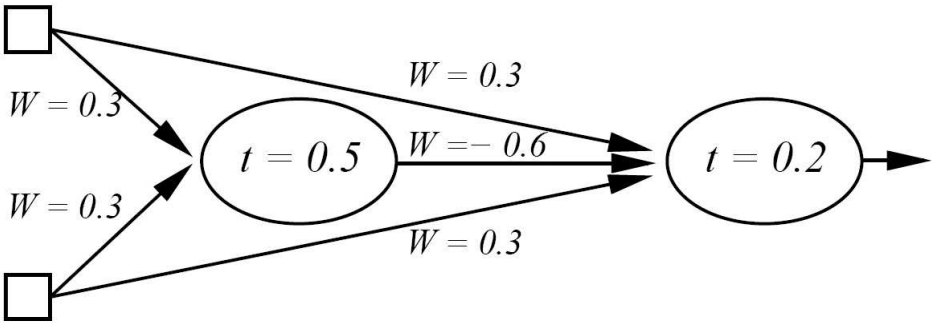

Figure 2: A network of step-function neurons that computes the XOR function

OR but weights the output of the hidden node negatively. The network shown in Figure 2 does the trick.