

Solutions to Assignment 5

1. In our formula, the information gain formula is

$$\text{Gain}(A) = I(p/[p+n], n/[p+n]) - \text{Remainder}(A)$$

In our problem, $p=6$ and $n=6$, so

$$I(6/12, 6/12) = (6/12) \log_2 (6/12) - (6/12) \log_2 (6/12) = 1$$

For the attributes, we have the following:

		Will Wait				Will Wait				Will Wait	
		Yes	No			Yes	No			Yes	No
Bar	Yes	3	3	Hungry	Yes	5	2	Rain	Yes	3	2
	No	3	3		No	1	4		No	3	4

Therefore, the gain for each is:

$$\text{Gain}(\text{Bar}) = 1 - [(6/12) * I(3/6, 3/6) + (6/12) * I(3/6, 3/6)] = 1 - [(0.5 * 1) + (0.5 * 1)] = 0$$

$$\text{Gain}(\text{Hungry}) = 1 - [(7/12) * I(5/7, 2/7) + (5/12) * I(1/5, 4/5)] =$$

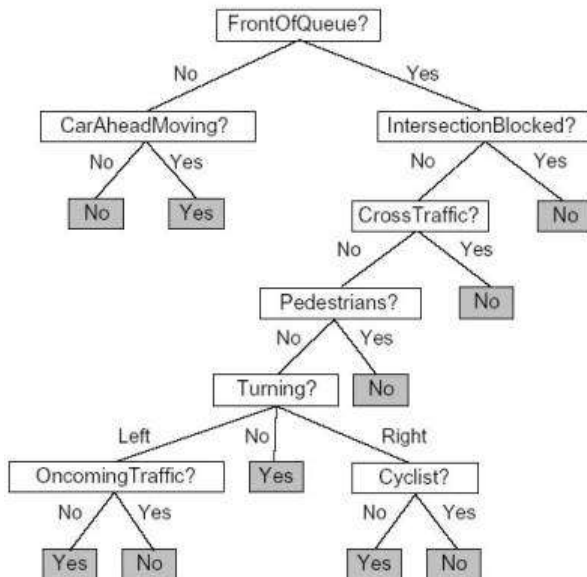
$$1 - [(0.5833 * 0.8631) + (0.4167 * 0.7219)] = 1 - [0.5034 + 0.3008] = 0.1958$$

$$\text{Gain}(\text{Rain}) = 1 - [(5/12) * I(3/5, 2/5) + (7/12) * I(3/7, 4/7)] =$$

$$1 - [(0.4167 * 0.971) + (0.5833 * 0.9852)] = 1 - [0.4046 + 0.5747] = 0.0207$$

Gain(Hungry) is clearly the best.

2. Here is a potential decision tree.



3. We will assume that Prolog is the logic programming language. It is certainly true that any solution returned by the call to *Resolve* will be a correct inverse resolvent. Unfortunately, it is quite possible that the call will fail to return because of Prolog's depth-first search. If the clauses in *Resolve* and *Unify* are infelicitously arranged, the proof tree might go down the branch corresponding to indefinitely nested function symbols in the solution and never return. This can be alleviated by redesigning the Prolog inference engine so that it works using breadth-first search or iterative deepening, although infinitely deep branches will still be a problem. Note that any cuts used in the Prolog program will also be a problem for the inverse resolution.

4. Some of the properties of perceptrons:

Outputs are independent of each other, as they are only a result of the inputs.

Can only answer questions which are linearly separable.

Simple.

Some properties of multi-layer networks:

Single layer networks can act as continuous functions, two layers can act as non-continuous functions.

Hard to characterize how many neurons are needed in hidden layer.

The obvious answer of a place where a multi-layer network would be better than a perceptron is in simulating the logical XOR gate, which was given as something a perceptron couldn't solve. A more complicated possibility would be approximating something like a tangent function, since it is non-continuous.