Artificial Intelligence Tutorial 5 - Answers

1. Use the table of examples below:

Exam	Use Calculator	Duration (hrs)	Lecturer	Term	Difficulty
1	yes	3	Jones	summer	easy
2	yes	3	Jones	spring	difficult
3	no	3	Smith	spring	difficult
4	no	2	Armstrong	summer	easy
5	yes	2	Jones	summer	easy

1a) Calculate the Entropy of the set of five examples with respect to the binary categorisation into difficult and easy problems. Use the formula:

$$\text{Entropy}(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

Where p_+ is the proportion of exams which are in the positive category (which we'll take to be *difficult*), and p_- is the proportion of exams in the negative category.

There are 2 exams in the positive category and 3 in the negative category. Hence, p_+ is 2/5 and p_- is 3/5. We can simply put these into the formula as follows:

Entropy(S) = $-2/5 \log_2(2/5) - 3/5 \log_2(3/5)$

To calculate $\log_2(2/5)$ using our calculators, we need the well known result that $\log_2(y)=\ln(y)/\ln(2)$, where ln(y) is the natural logarithm of y, and is found on most calculators. The result of the calculation is:

Entropy(S) = $(-2/5)(-1.322) - (3/5)(-0.737) = 0.529 + 0.4422 = 0.9712$

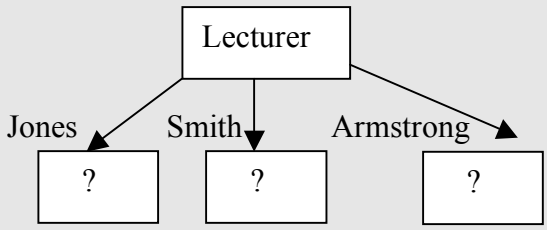
1b) Suppose an agent using the ID3 algorithm has chosen to use the lecturer attribute of the problem as the top node in the decision tree it is learning. In English, why would the algorithm have chosen that attribute? What would the partially learned decision tree look like after that choice?

1c) What attribute would be chosen as the next to put in the tree under the Jones arrow?

1b) The ID3 algorithm begins by deciding which attribute scores most for the information gain calculation, which is itself based on the entropy calculation. Information gain for attribute A is calculated as:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where $|S_v|$ is the number of examples in the set which take value V for attribute A. Information gain can be seen as the expected reduction in entropy which would be caused by knowing the value of attribute A. Hence, ID3 chooses the attribute which has the most expected reduction. The decision tree will therefore have the lecturer attribute as the top node of the tree, with three branches coming from it as follows:



1c) The lecturer attribute cannot be used, so the answer must be either the use-calculator, the term or the duration attribute. To determine which one would be used, we must calculate the information gain for each one with respect to the exams where Dr. Jones was the lecturer. So, we will only be using three examples in the following calculations, namely exams 1, 2 and 5: the ones which Dr. Jones lectured. Hence |S| will be taken as 3 from now on. Moreover, we need to calculate the entropy of this set of examples. Noting that only one example is difficult (positive) and two are easy (negative), we can calculate the entropy as:

Entropy(S) = $-(1/3)\log_2(1/3) - (2/3)\log_2(2/3) = 0.528 + 0.390 = 0.918$

We will start with the use-calculator attribute. This takes values yes and no, and the example set can be partitioned like this:

S_{ves} = {exam1, exam2, exam5} and S_{no} = {}

This means that the entropy of the 'yes' examples is the entropy of all the examples, so this is 0.918 as before. Also, the entropy of the 'no' examples is 0. Putting all this into the information gain calculation for the use-calculator attribute gives zero: because all the examples (of Jones' exams) require a calculator, knowing this fact does not reduce the entropy of the system.

We now move on to the duration attribute. The values for this are 2 and 3, and S_2 is {exam5}, whereas S_3 is {exam1, exam2}. We need to calculate the value of Entropy(S_2) and Entropy(S_3). For S_2, as there is only one example, the entropy will be zero, because we take $0*\log_2(0)$ to be zero, as discussed in the notes. For S_3, there is one difficult exam and one easy exam. Therefore, the value of Entropy(S_3) will be $-(1/2)\log_2(1/2) - (1/2)\log_2(1/2)$, which comes to 1. It had to be 1, because, with both examples having a different categorisation, the entropy must be as high as it possibly can be. The information gain for the duration attribute will therefore be:

Gain(S, duration) = Entropy(S)–(|S2|*Entropy(S2))/|S|-(|S3|*Entropy(S3))/|S|=0.918–0/3–2/3=0.251

If we finally look at the term attribute, we see that there are two possible values, spring and summer, and that S_{spring} = {exam2}, while S_{summer} = {exam1, exam5}. Hence Entropy(S_{spring}) will be 0, because there is only one value. Moreover, as exam1 and exam5 are both in the easy category, Entropy(S_{summer}) will also be zero. Therefore: Gain(S,term) = 0.918 – 0 – 0 = 0.918.

Hence, as the term attribute has the highest information gain, it is this attribute which will be chosen as the next node below the Jones arrow in the decision tree.

2. Suppose we have a function, f, which takes in two inputs and outputs an integer which takes only the value 10 or the value 17, such that

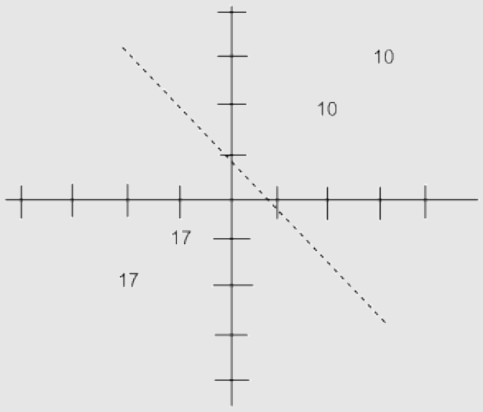
$$f(2,2) = f(3,3) = 10 \text{ and } f(-1,-1) = f(-2,-2) = 17.$$

2a) Plot f on a graph.

2b) Can f be represented as a perceptron? If so, explain your answer.

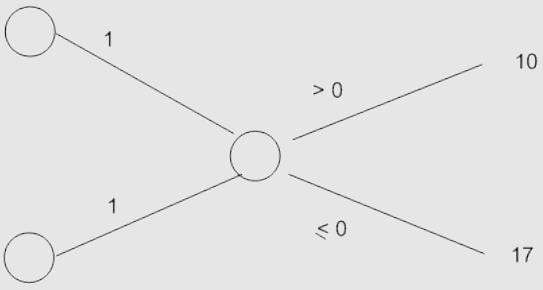
2c) Draw a suitable perceptron for the function.

2a) The plot is as follows:

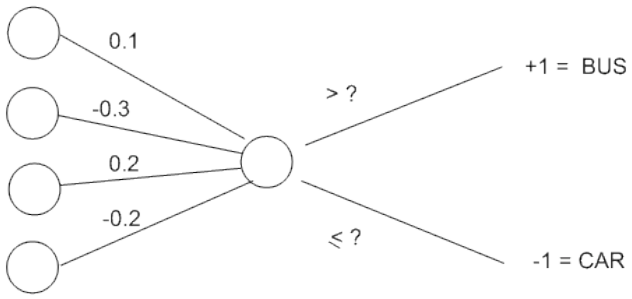


This is a linearly seperable graph, because we can draw a line through the graph which separates the 10s from the 17s, as indicated by the dotted line on the graph. Hence the function is learnable as a perceptron, because weights and a threshold can be found to distinguish the categories.

2b) To determine a suitable perceptron, we must realise that we only want the perceptron to "fire" when the two inputs are in the top right quadrant of the graph (i.e., both positive). Therefore, if we set the threshold to be zero, and the weights to be both 1, then two positive values input will produce a positive sum, and hence the perceptron will fire. On the other hand, two negative inputs will produce a negative sum. This will be less than the threshold (0), so the perceptron will not fire. Hence, the following simple perceptron is an acceptable representation of this function:



3. Suppose we are training a neural network to learn a function which takes three integers as input, representing weight, fuel capacity and passenger numbers and outputs either "car" or "bus". Suppose the ANN currently looks like this:



3a) Why are there four input nodes? Fill in the two ?s.

3b) Does the network predict a bus or a car for this triple of inputs: (10,12,13)?

3a) The top node is a special input node which always outputs 1. That's why there is seemingly one more node than the input requires. The advantage to having this is so that the threshold can be set at zero and doesn't have to be learned. So that means the two ?s are both zero.

3b) We have to remember that the output from the first input node is 1 and the output from the other three nodes is the same as the value input to them. Hence the weighted sum into the output node will be:

0.1*1 + -0.3*10 + 0.2*12 + -0.2*13 = 0.1 – 3 + 2.4 – 2.6 = -3.1

This is less than zero, so the network would predict that the triple of values came from a car.

3c) Suppose that the triple (10,12,13) has been mis-categorised by the perceptron. Using the perceptron learning rule, calculate the weight change for the weights in the network in light of this training network, if we use a learning rate of 0.1.

3d) What does the re-trained network look like?

3e) Does the re-trained network correctly categorise the example (10,12,13)?

3c) The actual output from the network for the triple (10,12,13) was -1 (a bus), but the target output was +1 (a car).

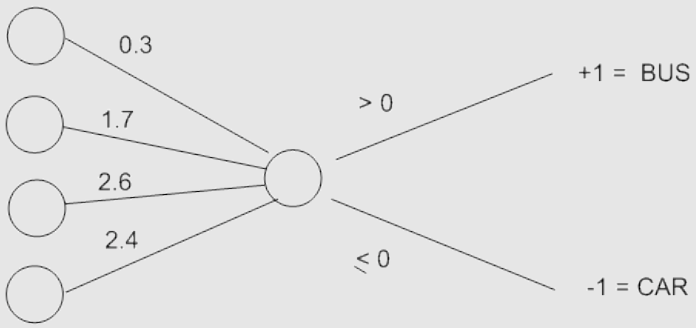
The weight change for a weight between an input node and the output node is calculated by multiplying the output from the input node by the difference between the target and observed output values. These values are then scaled by multiplying by the learning rate. The difference between the target and observed values for the triple is 1 – (-1) = 2. The weight changes are therefore calculated as:

change for weight 1 = 0.1 * 2 * 1 = 0.2
change for weight 2 = 0.1 * 2 * 10 = 2
change for weight 3 = 0.1 * 2 * 12 = 2.4
change for weight 4 = 0.1 * 2 * 13 = 2.6

3d) The weight changes get added on to the existing weights. Hence the new weights are:

weight 1 = 0.1 + 0.2 = 0.3
weight 2 = -0.3 + 2 = 1.7
weight 3 = 0.2 + 2.4 = 2.6
weight 4 = -0.2 + 2.6 = 2.4

and the altered network looks like this:



3e) If we now feed the input value forward through the network, we get the following value for the weighted sum:

1*0.3 + 10*1.7 + 12*2.6 + 13*2.4

This is clearly greater than zero, so the network now correctly predicts that the values came from a bus, rather than a car as it previously did.

3f) Has the network over-corrected? This triple: (5,7,7) used to be (correctly) categorised as a car. Is it still correctly categorised?

3f) The weighted sum for input triple (5,7,7) is 1*0.3 + 1.7*5 + 2.6*7 + 2.4*7 > 0. Hence this triple is now predicted to come from a bus, so this example is no longer correctly categorised by the network. It's likely that we should have used a smaller learning rate.