CS303A Homework 3

Q1: Consider the following data set comprised of three binary input attributes (A_1 ,A_2 , and A_3) and one binary output:

Example	A_1	A_2	A_3	Output y
$\mathbf{x}_1$	1	0	0	0
$\mathbf{x}_2$	1	0	1	0
$\mathbf{x}_3$	0	1	0	0
$\mathbf{x}_4$	1	1	1	1
$\mathbf{x}_5$	1	1	0	1

Use the algorithm (as below) to learn a decision tree for these data. Show the computations made to determine the attribute to split at each node.

function DECISION-TREE-LEARNING($examples$, $attributes$, $parent\_examples$) **returns** a tree

  if $examples$ is empty **then return** PLURALITY-VALUE($parent\_examples$)
  else if all $examples$ have the same classification **then return** the classification
  else if $attributes$ is empty **then return** PLURALITY-VALUE($examples$)
  else
    $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, $examples$)
    $tree \leftarrow$ a new decision tree with root test A
    for each value v_k of A **do**
      $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
      $subtree \leftarrow$ DECISION-TREE-LEARNING(exs, $attributes - A$, $examples$)
      add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
    return $tree$

`PLURALITY-VALUE` selects the most common output value among a set of examples, breaking ties randomly.
`IMPORTANCE` is information gain, based on entropy.

Q2: Construct a support vector machine that computes the `XOR` function. Use values of $+1$ and -1 (instead of 1 and 0) for both inputs and outputs, so that an example looks like $([-1, 1], 1)$ or $([-1, -1], -1)$. Map the input $[x_1, x_2]$ into a space consisting of x_1 and $x_1 x_2$.

a. Draw the four input points in this space, and the maximal margin separator. What is the margin?
b. Now draw the separating line back in the original Euclidean input space.

Q3: Suppose you had a neural network with a linear activation function $g(x) = cx + b$ (c and b are constants)

a. Assume that the network has one hidden layer. For a given assignment to the weights $\mathbf{w}$, write down equations for the value of the units in the output layer as a function of $\mathbf{w}$ and the input layer $\mathbf{x}$, without any explicit mention of the output of the hidden layer. Show that there is a network with no hidden units that computes the same function.
b. Repeat the calculation in part (a), but this time do it for a network with any number of hidden layers.
c. Suppose a network with one hidden layer and linear activation functions has n input and output nodes and h hidden nodes. What effect does the transformation in part (a) to a network with no hidden layers have on the total number of weights? Discuss in particular the case $h \ll n$.