EECS 649: PROBLEM SET #11

Reading:

- R&N **19.6.2-4; 21.1; 19.7.1,4,5; 21.8; 19.9; 20.1-3**
  [needed: reading, Learning from Examples 3, Learning Probabilistic Models]

Total Points: 100

Notes:

- Submitted electronically (via Gradescope)
- **This problem set has a program; start early! :**
  - **Problems 11.3, 11.4, 11.5 and 11.6 need the Learning from Examples 3 lecture**
  - **Problems 11.1 and 11.2 need the Learning Probabilistic Models lecture**
  - **You can probably do 11.4 after doing just a little reading in the book**
  - **For 11.5 and 11.6, there is a linked example**

Problem 11.1 [10 points]

Compute the Conditional Probability Tables (CPTs) that would be "learned" given the 100 instances in Figure 20.1 on page 1 **(i.e., the robot arm data)** of this linked handout:

Problem 11.2 [25 points]

For the 14 data points in Table 3.2 on page 1 **(PlayTennis table at bottom)** of this linked handout

a. Draw the naive Bayes classifier.
b. Find all its CPTs.
c. Compute the predicted target attribute for the example appearing below the line under D14.
d. Compute the conditional probability this is correct given the observed data.

The answers to (c) and (d) are quite inter-related. Depending on how you approach this problem, you might have an answer to (d) before you answer (c). That is fine, of course.

Recall that we started this problem in lecture.

Problem 11.3 [10 points]

Construct by hand a feedforward neural network that computes the XOR of two inputs. Your network should have (1) a hidden layer consisting of linear threshold elements receiving inputs x_1 and x_2, and (2) a final linear threshold output unit receiving inputs from the outputs of the hidden layer (but not inputs from x_1 and x_2).

For this problem, you are to assume step function activation functions. You can place the threshold number inside your neuron. For example, an "AND" net can be drawn simply as follows:

```
          w1=1  +---------+
x1  ----------->|threshold|
                |   1.5   |------->
x2  ----------->|         |
          w2=1  +---------+
```

Problem 11.4 [10 points]
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 (product of x_1 and x_2) . Draw the four input points in this space, and the maximal margin separator. What is the margin? Now draw the separating line back in the original Euclidean input space.
See Figure 19.22 of R&N, 4e for an example.

Problem 11.5 [20 points]
The following training set is linearly separable:

inputs			
x1	x2	x3	output
1	0	0	1
0	1	1	0
1	1	0	1
1	1	1	0
0	0	1	0
1	0	1	1

Train (by hand) a 0-1 step threshold element on this training set. Counting input (x0) for the threshold, the unit has four inputs. Assume the initial values of all weights are zero.

a. (15 points) Train your unit with the *fixed increment error-correction procedure*: Equation (19.8) of R&N, 4e with **alpha=1**. Show the weights after each training example. Use the stopping criterion: the weights have not changed in one epoch (whole pass through the data).

b. (5 points) Draw a sketch of a 3-d cube with the inputs above as vertices (label positive--one-valued--examples with a filled circle and negative ones with an open circle), and sketch in the separating plane corresponding to the final weight set.
For the drawing in part (b), use the following linked figure as your axis.

See the linked Perceptron Learning Example.

The following problem requires a little programming.

Problem 11.6 [25 points]
Implement the Perceptron-Learning algorithm (see Equation (19.8) in R&N, 4e; also see the linked Perceptron Learning Example). Test it by re-doing Problem 11.5(a) above.

Turn in your **code** and your program's **output**. Comment on your results.

Notes: Preserve paper by printing out the weights after each training example in successive rows. That is, plot the transpose of the weight vector. As in Problem 11.5 above, remember that the perceptron learning rule requires that g'(in) be omitted from the update equation.

The notation concerning “transpose” simply means that you should write the weight vector as a row vector (even though you are probably more familiar with writing vectors of numbers as column vectors).