CS4700 Foundations of Artificial Intelligence
Fall 2016
Assignment 4 (60 pts total)
Due Tuesday, Dec. 6, 6pm on gradescope.com

You can discuss the homework with other students but you need to write up the solution in your own words. Also, you can consult the textbook and the lecture slides but you are not allowed to search for a solution on-line. Start each solution on a new sheet.

1. Neural Networks (30 pts.) A pure reflex agent is a function from the agent’s percepts to its actions. For each instantiation of the agent’s percepts, it specifies the action that the agent should take. Neural networks are a fairly common way to represent a reflex agent’s action function, one which allows the agent to learn a good action function from examples.

Consider a pure reflex agent with three actions — turn left, go forward, and shut off. The agent has three percepts at-home (H), gold-on-left (G), and bump-in-front (B). Our goal is to build a neural network control for the robot that will generate the same behavior shown in the decision tree in Figure 1.

(a) (7 pts.) Write down the Boolean function (using the results of the percepts and the logical propositions and combining them with and’s and or’s) describing when the agent shuts off. And show a perceptron (with labeled inputs, weights, and threshold function, 0/1 signals) which is activated when the agent should shut off.

(b) (8 pts.) Same as (a), but for the case where the agent should go forward.

(c) (5 pts.) Write down a Boolean function describing when the agent turns left. Explain why we cannot construct a perceptron which is activated when and only when the agent should turn left.

(d) (10 pts.) Construct a multi-layered feed-forward neural network that can tell the agent which action to take. The network will have three inputs, corresponding to the agent’s three percepts, and three outputs, corresponding to the agent’s three actions. For each possible combination of values of the three inputs (percepts), the output for precisely one action (the correct action) should be on. Your network should show the structure (which edges are present) as well as weights and thresholds. (Hint: think how you can incorporate your perceptrons from (a) and (b) into the final solution. Use step-function for activation function.)
2. **Neural Network Learning (30 pts.)** A central concept in (deep) neural net learning is the notion of *gradient descent*. During learning, the network adjusts its weights to get the desired output for a given input example by minimizing the squared error in the output value. The weights are adjusted in small steps using gradient descent (i.e., in each iteration take a small step in weight space in the opposite direction of the gradient wrt the weights). The backpropagation algorithm provides a smart way of calculating the changes in all the weights, given an input pattern and the desired output value. Because the general backpropagation equations are somewhat obscured by notational details (see lecture slides), here we consider a more direct calculation of the gradient for a small network.

See the two unit neural network in Figure 2. The two inputs are $x_1$ and $x_2$ (no extra threshold input), which can take on values 0 or 1. Each unit takes the weighted sum of its inputs and uses a sigmoid activation function to calculate its output value. Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$.

(a) (10 pts.) Write $o$, the output of unit #2, as a function of $x_1, x_2, w_1, w_2, w_3, w_4,$ and $f$. (Strictly speaking, we don’t need to use $f$.)

(b) (20 pts.) Now, given an input pattern $x_1$ and $x_2$, and a desired output $o^*$ (either 0 or 1), we want to adjust the weights in the network to minimize the error in the output, $o$. That is, we want to minimize the error given by $Err = (o - o^*)^2$. We do this by taking a small step in the direction opposite of the gradient in terms of the weights. (Aside: we consider $Err$ here a function of the weights, $w_1, w_2, w_3,$ and $w_4$, while $x_1, x_2,$ and $o^*$ are constants given by the training example under consideration.) Calculate the gradient of $Err$ wrt the weights. I.e., give the partial derivatives of $Err$ wrt $w_1, w_2, w_3,$ and $w_4$. Your expressions for the derivatives should only use $x_1, x_2, o^*, w_1, w_2, w_3, w_4,$ and $f$. (Again, strictly speaking, we don’t need to use $f$.) This means that from these expressions, given an input pattern, a desired output, and current weight values, we can directly calculate the actual numerical values of the partial derivatives to make our gradient descent step. Give intermediate steps in your calculations for partial credit.