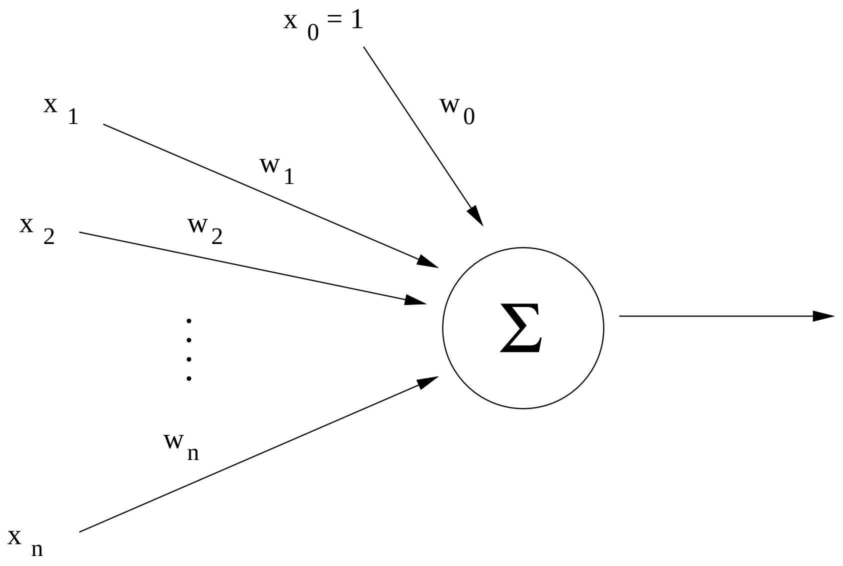
Artificial Intelligence - Assignment 5

Posted Friday, November 28, 2003, 11:30pm

This homework will be accepted without penalties until Monday, December 8, at noon.

1. [20 points]**Building a simple neural net**

   Consider the simple thresholded neuron, called perceptron, shown below. It accepts boolean inputs $x_1, x_2, \ldots, x_n \in \{0, 1\}$ and computes the weighted sum $S = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$. Unlike the sigmoid neurons, which we discussed at length in class, this kind of neuron returns a boolean value, not a real number. The output is 1 if $S \geq 0$ and 0 if $S < 0$.



   Now consider the following very hypothetical situation:

   The McGill cafeteria has asked the AI class to help them study the relationship between drinking orange juice and academic performance. If a student has an exam on a Friday, we want to predict the exam result based on whether the student had any orange juice for lunch on Monday, Tuesday, Wednesday and Thursday.

   A school week was arbitrarily selected, and 100 students were questioned. They each had to report on which days (Monday to Thursday) they did drink orange juice, and the outcome of their exam (pass or fail). The study revealed that all students who drank orange juice on exactly two of the days (and not on the other two days) passed their exams. All other students failed.

   Model the juice/exam function using a network composed of neurons such as the one given above. The requirements are very simple:

   - Your network must have 4 boolean inputs, one each for Monday, Tuesday, Wednesday and Thursday. The input value for each day is 1 if and only if orange juice was consumed on that particular day and 0 if no juice was consumed.
   - The network will have one hidden layer and exactly one output neuron. The output should be 1 if and only if the student is predicted to pass and 0 if the student is predicted to fail.

- Whenever juice was consumed on two of the days and not on the other two days, your network must output 1, otherwise it must output 0.

There are many ways to correctly build a network to these specifications. There is a solution with 6 neurons in the hidden layer, and one with only 2 neurons in the hidden layer.

(a) [15 points] Find both solutions and produce a drawing of each one, specifying exactly what all the weights are.

(b) [5 points] Which of the two network structures would you use if you actually wanted to perform learning based on real data? Give at least one reason for your choice.

2. [10 points] Russell and Norvig, pg. 762 Problem 20.19.

3. [20 points]**Gradient descent**

One way in which overfitting occurs in neural networks is is when the weights become too large. One way to avoid this phenomenon is to use an error function that penalizes large weights. For example, consider a single sigmoid neuron with weights $\theta = \langle \theta_0 \ldots \theta_m \rangle$ and let o_θ be the output of this neuron.

For a given instance, we define the error to be:

$$E(\theta) = \frac{1}{2}\left[(y - o_\theta)^2 + \lambda \sum_{j=1}^{m} \theta_i^2\right]$$

In other words, we add to the usual error a term that penalizes large weights.

(a) Derive an update rule for the weights of a sigmoid unit relative to this error function. Show the derivation and the final rule

(b) Now consider a similar error function for the case of a multi-layer neural network. Write down the rule for updating the weight w_{ij} of the network relative to this error function. Hint: you do not need to do another derivation in this case.).

4. [30 points] Russell and Norvig, pg. 646, Problem 14.4.

5. [10 points] Russell and Norvig, pg. 647, Problem 17.5, parts a and b.

6. [10 points] **Understanding Reward Structures**

Suppose you change the reward structure of the episodic task in the previous question by giving a reward of +1 when the robot enters the correct room, and a reward of -1 when it enters an incorrect room. You are wondering if the signs of the rewards are important, or only the intervals between them. Consider adding a constant C to all rewards. How would this affect the task?

Now consider the same question for a continuous (infinite-horizon) task. Prove that adding a constant C to all rewards simply adds a constant, K, to the values of all states. Compute K and explain how this would affect a learning task.