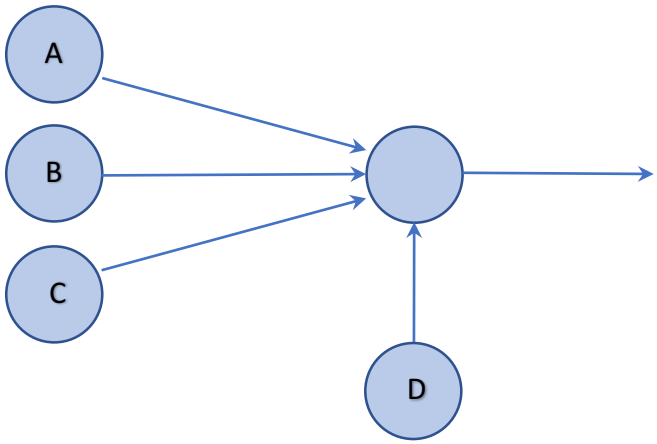
CSCI 3202 Problem Set 4. Due in class, Wednesday Dec. 13 (hard copy)

Problem 4.1 (50 points)

In this problem, you will write a program (original code, please – no libraries) to create and train a perceptron. Your program should implement a perceptron with three inputs, as follows:



The input values on A, B, and C are either 1 or 0. The perceptron should output a 1 if the input values of A and C are both 1, and 0 otherwise. (The value of B does not matter.) Thus, the perceptron should output a 1 if the input values are “1 1 1” (for A, B, and C); it should output a 0 if the input values are “0 1 1”. The value of the D input value is fixed at 1, and the edge weight from the D input value is fixed at -1. The output neuron should implement a continuous sigmoidal function as shown in class.

Train the perceptron by initializing the three edge weights to random values in the range -1 to 1; then use the perceptron training rule shown in class to update the edge weights after presenting repeated random three-bit patterns to the perceptron (updating occurs when the perceptron output is in error; if the output is correct, no updating takes place). Note that only the weights from A, B, and C are trained; the value of -1 from the D input remains fixed. Show the edge weight values after every 250 training samples, up to about 8000 samples. (By the way: you may have to experiment with several “alpha” values in the training rule to effectively train your perceptron.)

Problem 4.2 (50 points)

In this problem, you will implement a one-dimensional version of the Schelling "neighborhood model" shown in class. Your "city" should consist of 60 houses in a circle; thus, each house has immediate neighbors to its left and right. To begin with, 27 houses should be "occupied" by a family of type 1; 27 houses should be occupied by a family of type 2; and 6 houses should be empty ("type 0"). The initial positions of the occupants should be chosen at random (i.e., there is no deliberate pattern to the initial positions of occupants in the city).

A state of the city can be represented by a 60-digit string of 0's, 1's, and 2's where we can begin the string with the house at due north ("12 o'clock", if you want to think of it that way):

1 1 0 2 1 2 2 1 2 1 1 0 1 1 2 2 2 2 1 1 2 1 2 2 0… 2 2 2 [sixty values]

Note, in this example, that the final "2" is actually a neighbor of the initial "1".

Now, you should implement the following rule: an occupant is *dissatisfied* if it does not have at least two of its own kind among the four neighbors within two spaces of it. For instance, the initial "1" is dissatisfied (it has 1 "1" neighbor, 1 "0" neighbor", and 2 "2" neighbors). The second "2" in the string (at position number 5, counting from 0) is not dissatisfied: it has 2 "2" neighbors (and 2 "1" neighbors).

Repeat the following action four hundred times, or until there are no more dissatisfied occupants:

a. Find a random dissatisfied occupant.
b. Move that occupant to a randomly chosen empty location (leaving a "0" in the place where the occupant formerly lived; thus, the city should always have 27 "1" and 27 "2" occupants in total).

Show the sixty-digit number representing the city after every 20 iterations, up to four hundred iterations. Does the "ring city" move toward a "totally satisfied" state? What does that state look like?