SysEng 5211/ElecEng 5810/Comp Eng 5310

Name: \_\_\_\_\_\_Ryan Patton\_\_\_\_

Homework 1

This is an individual assignment

1. Design and describe (what would you use for weights and bias) a single-layer perceptron can classify the following patterns:

Table 1, Pattern of Classification

|  |  |  |
| --- | --- | --- |
| Input X1 | Input X2 | Result |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

How is this related to the EXCLUSIVE OR function? Explain if this single-layer perceptron can implement that function.

The weights and bias for the logic gate function seen above would look like the following:

W1 \* X1 + W2 \* X2 + b = y, where W1 and W2 are weights for each of the input X’s and b is the bias added to the weighted inputs to give the result y.

The perceptron algorithm states the following: Prediction (y’) = 1 if WX + b > 0 and 0 if WX + b <= 0.

If the first row is initialized with W1 = 1, W2 = 1, and bias = -1, the result is: 0 + 0 – 1 = -1. Referring to the perceptron algorithm, if WX + b <= 0, then y’ = 0. Therefore, this row is correct.

For the second row with X1 = 0 and X2 = 1, the result is: 0 + 1 – 1 = 0. If WX + b <= 0, then y’ = 0 so this is correct too and the output is 0.

The same would follow for row 3 as X1 = 1 and X2 = 0, the result is 0 again: 1 + 0 – 1 = 0.

For the fourth row with X1 = 1 and X2 = 1, the result is: 1 + 1 -1 = 1. The perceptron algorithm is still valid for this.

Using the AND gate seen above and using the perceptron algorithm, W1 = 1, W2 = 1, and bias = -1 is valid: X1 \* (W1 = 1) + X2 \* (W2 = 1) – 1 = y

The truth table shown is a 2-input AND gate. The resultant output is truth if both of the inputs X1 and X2 are true (1), otherwise it’s false (0).

The EXCLUSIVE OR function would give results 0, 1, 1, 0. The truth table for the EXCLUSIVE OR function only goes to 1 when both of its inputs are logically different.

The Boolean representation of an XOR gate is:

X1\* X’2 + X’1 \* X2, which simplifies to (X1 + X2) \* (X1 \* X2)’

The XOR gate consists of an OR gate (X1 + X2), a NAND gate (-X1 – X2 + 1) and an AND gate (X1 + X2 – 1.5) so 2 perceptrons will need to be combined to make it work:

OR(2X1 + 2X2 – 1)

NAND(-X1 – X2 + 2)

AND(X1 + X2 – 1)

1. What are the two principles that make up fuzzy set theory goals described in class? How are these principles upheld in fuzzy systems?

The two principles that make up fuzzy set theory goals described in class are the principle of least commitment (PLC) and the principle of graceful degradation (PGD). PLC is upheld by avoiding making deterministic decisions for as long as possible, since it is difficult to recover from a wrong crisp decision early on. Marr described it as “Don’t do something that later must be undone.” The goal is to keep options open until the situation demands a final answer. PGD is upheld by delivering a partial answer as input degrades. By doing so, the potential to degrade is much more graceful than crisp counterparts, encompassing a degree of robustness and continuity.

1. Consider the function:

What are three different representations that could be used to solve this problem, including a brief description of how you would encode the representation? Which do you think would have superior performance and why?

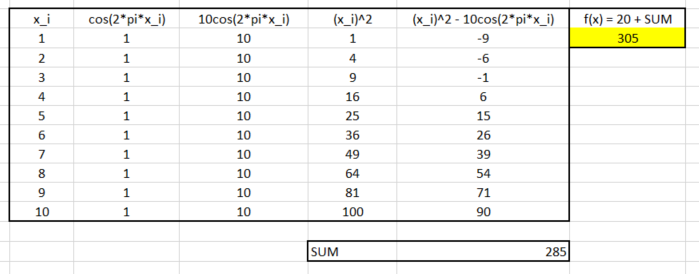
Three different representations that could be used to solve this problem are Neural Networks, Fuzzy Systems, and Evolutionary Computation.

Neural Networks – at a high-level, there would be 10 inputs going into the equation for x\_i = 1,2,3…10, and different bias/weights would be given to impact the predicted outcome, f(x). Different transfer functions may be picked that yield better results converging on 1 solution.

Fuzzy Systems – May use limits as x\_i approaches 10 with an intersection or union, working with the cosine function in the equation to come up with multicriteria decision-making, when multiple results are possible. The decision functions may sometimes be described in linguistic terms for fuzzy systems but that would not apply for this particular problem. Fuzzy systems characterize situations by attributing a degree to which a certain object belongs to a set. In this problem, fuzzy set theory would not be ideal because I believe you just end up with one definitive answer for every point along x\_i, not a spectrum of answers where cos(2\*pi\*x\_i) could be positive or negative. The only exception would be if that were the case where cos(2\*pi\*x\_i) yielded a positive and negative answer at every point because then there could be a diverge from a set outcome into multiple outcomes depending on which numbers were paired.

Evolutionary Computation – optimization algorithms seen in nature (reproduction, mutation, recombination, and selection) that help approximate solutions. Evolutionary computation popularly uses fitness approximation to help evaluate fitness functionality and give more accurate results to determine how well it’s fitting within the context of the problem. It’s built on the idea that the least-fit choices can be replaced with new, more accurate/better choices at each iteration.

I went ahead and solved the problem in Excel (seen below) so I could give better input on what the best performance method would be.



I think neural networks would have superior performance based on my findings because you’re looking for a definitive predictable answer, and if the right function is chosen with even sub-optimal weights/biases it would quickly converge on the solution. Neural nets should be able to predict the output fairly accurately and early on as your input is only one changing variable. You may have to divide the problem into training and testing. A fuzzy problem might be used for determining “which boy is tall” in a dataset, or a problem along those lines, because tall would be subjective and fuzzy set theory would help determine different groupings for what would be considered tall in the dataset. Evolutionary computation would help update the formula at each round to come up with the optimal solution. For example, if you had to get all the values within a certain threshold to pass a test, the solution could update at each iteration replacing values that do not work for passing thresholds until passing values are reached, if it’s possible for that to happen at all for the problem. This problem is really simple for a neural net but I believe it’s best equipped for this problem because you’re not searching for an optimization to the problem, the function doesn’t give multiple possible solutions at each iteration, and you’re trying to predict the pattern seen at each epoch to come up with a solution. If there were 10,000 epoch “i’s” for this problem, you could split it up easier into training, testing, and validation datasets, which could be a limiting factor for neural nets for this problem given that there are only 10.