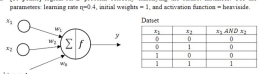
Bronco ID: 013001924
Last Name: Nalbandian
First Name: Andre

Assignment 4

Wednesday, October 27, 2021 12:43 PM

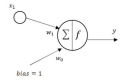
1. [20 points] Train the perceptron network below to solve the following logical problems:

   a. [10 points] logical AND problem correctly classifying the dataset instances. Use the parameters: learning rate η=0.4, initial weights = 1, and activation function = heaviside.



x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

   b. [10 points] logical NOT problem correctly classifying the dataset instances. Use the parameters: learning rate η=0.1, initial weights = 0, and activation function = heaviside.



x_1	NOT x_1
0	1
1	0

1a)

Dataset

x1	x2	x0	w1	w2	w0	t	z(net)	y	(t-y)	^w1	^w2	^w0
0	0	1	1	1	1	0	1	1	-1	0	0	-0.4
0	1	1	1	1	0.6	0	1.6	1	-1	0	-0.4	-0.4
1	0	1	1	0.6	0.2	0	1.2	1	-1	-0.4	0	-0.4
1	1	1	0.6	0.6	-0.2	1	1	1	0	0	0	0
0	0	1	0.6	0.6	-0.2	0	-0.2	0	0	0	0	0
0	1	1	0.6	0.6	-0.2	0	0.4	1	-1	0	-0.4	-0.4
1	0	1	0.6	0.2	-0.6	0	0	0	0	0	0	0
1	1	1	0.6	0.2	-0.6	1	0.2	1	0	0	0	0
0	0	1	0.6	0.2	-0.6	0	-0.6	0	0	0	0	0
0	1	1	0.6	0.2	-0.6	0	-0.4	0	0	0	0	0
1	0	1	0.6	0.2	-0.6	0	0	0	0	0	0	0
1	1	1	0.6	0.2	-0.6	1	0.2	1	0	0	0	0

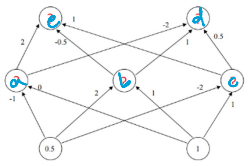
1b)

x1	x0	w1	w0	t	z(net)	y	(t-y)	^w1	^w0
0	1	0	0	1	0	0	1	0	0.1
1	1	0	0.1	0	0.1	1	-1	-0.1	-0.1
0	1	-0.1	0	1	0	0	1	0	0.1
1	1	-0.1	0.1	0	0	0	0	0	0
0	1	-0.1	0.1	1	0.1	1	0	0	0
1	1	-0.1	0.1	0	0	0	0	0	0

2. https://github.com/NootCode/CS4210-Assignment4/blob/master/Perceptron/perceptron.py

3. [20 points] The figure below is a network of linear neurons, that is the output of each neuron is identical to its input (linear activation function). The numbers at the connections indicate the weights of the links.

   a. [10 points] Find the value at the output nodes of the network for the given input (0.5,1).



   b. [5 points] Find the value at the output nodes of the network for the input (1,2). [5 points] Do you need to repeat the computations all over again? Why?

c) a) $0.5*1 + 1*0 = -.5$

b) $0.5*2 + 1*1 = 2$

c) $0.5*-2 + 1 = 0$

d) $-2(-.5) + 1(2) + 0.5(0) = 3$

e) $2(-.5) + -.5(2) + 1(0) = -2$

b) If the values were changed to (1,2) it would be equivalent to multiplying the original (.5,1) by 2. Therefore each of the outputs will follow the same ratio and be doubled. The computation between each of the neurons do not need to be repeated.

a) -1
b) 4
c) 0
d) 6
e) -4

4. https://github.com/NootCode/CS4210-Assignment4/blob/master/DeepLearning/deep_learning.py

5. [15 points] Consider the dataset below.

Outlook	Temperature	PlayTennis
Sunny	Hot	No
Overcast	Cool	Yes
Overcast	Hot	Yes
Rain	Cool	No
Overcast	Mild	Yes

We will apply *steady state* Genetic Algorithms ($r = 0.5$) to solve this classification problem, by using a similar approach of the Find-S algorithm. If the instance matches the rule pre-condition of the chromosome, you predict according to its post-condition, otherwise you predict the opposite class defined by the chromosome (there must be a prediction for each instance then). Follow the planning below to build your solution.

- Representation: single if-then rule by using bit strings (binary encoding).
  - Outlook <*Sunny, Overcast, Rain*>
  - Temperature <*Hot, Mild, Cool* >
  - Examples:
  - Outlook = *Overcast* → 010
  - Outlook = *Overcast* ∨ Rain → 011
  - Outlook = *Sunny* ∨ *Overcast* ∨ Rain → 111
  - Outlook = (*Overcast* ∨ *Rain*) ∧ (Temperature = *Hot*) → 011100
  - Outlook = *Sunny* ∧ Temperature = *Hot* then PlayTennis = *yes* → 1001001
- Initial population (Chromosomes) : (C_1=1001001, C_2=0100101, C_3=1011000, C_4=1101100).

Solution format:

Fitness(C_1) = ? $C_1 = 1001001 = 1/5 = .2$
Fitness(C_2) = ? $C_2 = 0100101 = 3/5 = .6$
Fitness(C_3) = ? $C_3 = 1011000 = 4/5 = .8$
Fitness(C_4) = ? $C_4 = 1101100 = 2/5 = .4$

1st generation (C_1, C_2, C_3, C_4): C_2 C_3 C_4 C_6

Pr(C_1) = ? (4th) $C_1 = .1$
Pr(C_2) = ? (3rd) $C_4 = .2$
Pr(C_3) = ? (2nd) $C_3 = .3$
Pr(C_4) = ? (1st) $C_3 = .4$

C_5 = ??????? → C_5 = ??????? $C_5 = 1011100$
C_6 = ??????? → C_6 = ??????? $C_6 = 1101000$

Fitness(C_5) = ? $C_5 = .6$
Fitness(C_6) = ? $C_3 = .8$
Fitness(C_5) = ? $C_5 = 1011106 \rightarrow 4/5 = .8$
Fitness(C_6) = ? $C_6 = 1101000 \rightarrow 3/5 = .6$

2nd generation (?, ?, ?, ?) C_3 C_4 C_5 C_6

Pr(C_2) = ? (4th) = $C_4 = .214$
Pr(C_3) = ? (3rd) = $C_2 = .214$
Pr(C_4) = ? (2nd) = $C_5 = .286$

$C_1 = .2/2 = .1$
$C_2 = .6/2 = .3$
$C_3 = .8/2 = .4$
$C_4 = .4/2 = .2$

Crossover

Mask = 1110000

1011000 → 1110000 → 1011100
1100100 → 1101000

P(

- Outlook = *Overcast* ∨ Rain → 011
- Outlook = *Sunny* ∨ *Overcast* ∨ Rain → 111
- Outlook = (*Overcast* ∨ *Rain*) ∧ (Temperature = *Hot*) → 011100
- Outlook = *Sunny* ∧ Temperature = *Hot* then PlayTennis = *yes* → 1001001

- Initial population (Chromosomes) : (C_1=1001001, C_2=0100101, C_3=1011000, C_4=1101100). Population size should remain the same (4 individuals) over time.
- Fitness function: accuracy
- Penalty criterion: no penalty.
- Selection method: The best two chromosomes are carried over to the next generation. The other two are selected for crossover by using the roulette wheel (simulation).
- Crossover strategy: single-point crossover with mask 1110000 in the 1st generation, two-point crossover with mask 0001100 in the 2nd generation. Use the following chromosomes to perform crossover (simulating the process of spinning the roulette wheel) according to the relative fitness (sectors of a roulette wheel), generating two offspring for each crossover:
  - (1st and 3rd) in the 1st generation
  - (1st and 2nd) in the 2nd generation
- Mutation: on the 6th bit of the chromosome(s) 1011000 generated during the 2nd generation.
- Termination criteria: accuracy = 1.0. **Return the corresponding chromosome(s) – your model.**

2nd generation (?, ?, ?, ?)

Pr(C?) = ? (4th) = C_4 = .214
Pr(C?) = ? (3rd) = C_7 = .214
Pr(C?) = ? (2nd) = C_8 = .286
Pr(C?) = ? (1st) = C_3 = .286

C? = ??????? → C? = ??????? C_7 = 1011010
C? = ??????? → C? = ??????? C_8 = 1011100

Applying mutation on 1011000 C_5 = 1011100

Fitness(C?) = ? C_8 = .8 C_3 = 1011010
Fitness(C?) = ? C_7 = 1
Fitness(C?) = ? C_5 = .8
Fitness(C?) = ? C_3 = 1

Final answer: C_3 = 1011010
C_7 = 1011010

1101100 → ... → 1101000

P(C)
$C_3 = C_5 = .2/2.8 = .286$
$C_2 = C_6 = .6/2.8 = .214$

Double Crossover

C_5 1011100 → 0001100 → 1011000 C_7
C_3 1011000 → 101 1100 C_8

6. [15 points] Consider the combinatorial optimization problem known as the "0-1 Knapsack problem", where we need to fill a knapsack with objects of different weights and values. The goal is to fill the Knapsack with the highest possible value, not exceeding its maximum capacity (weight supported) and carrying only one type of each object inside. Below is the list of available objects, with their respective weights and values.

Maximum weight capacity (C) = 15 kg.

Object	Tablet	Laptop	Projector
Weight (w)	5 kg	8 kg	10 kg
Value (v)	$ 570.00	$ 710.00	$ 640.00

Apply *generational* Genetic Algorithms (r = 1) to solve this optimization problem strictly following the planning below.

- Representation: binary, identifying whether the object should be included or not, with the 1st, 2nd, and 3rd positions of the chromosomes referring to the Tablet, Laptop, and Projector respectively.
- Initial population (Chromosomes) : (C_1=000, C_2=001, C_3=010, C_4=100). Population size should remain the same (4 individuals) over time.
- Fitness function: $\sum_{i=1}^{n} v_i x_i$, with n being the number of objects in the knapsack and x_i being the number of instances of object i to include in the knapsack (1, if the object is included and 0, otherwise).
- Constraint: $\sum_{i=1}^{n} w_i x_i \leq C$.
- Penalty criterion: If the maximum capacity of the knapsack (C) is exceeded, discard the obtained chromosome (offspring) and replicate the best among the two of its parents.
- Selection method: roulette wheel (simulation)
- Crossover strategy: single-point crossover with mask 110 in the 1st generation, single-point crossover with mask 100 in the 2nd generation. Use the following chromosomes to perform crossover (simulating the process of spinning the roulette wheel) according to the relative fitness (sectors of a roulette wheel), generating two offspring for each crossover:
  - (2nd and 3rd) and (2nd and 1st) in the 1st generation
  - (1st and 1st) and (1st and 2nd) in the 2nd generation
- Mutation: on the 3rd bit of the chromosome(s) 101 generated during the 2nd generation.
- Termination criteria: stop at the end of the 2nd generation. **Return the best chromosome found.**

Summary of the optimization function: $\max \sum_{i=1}^{n} v_i x_i$

subject to $\sum_{i=1}^{n} w_i x_i \leq C$ *and* $x_i \in \{0,1\}$

Solution format:

Fitness(C?) = ? C_1 = 000 = 0
Fitness(C?) = ? C_2 = 001 = 640
Fitness(C?) = ? C_3 = 010 = 710
Fitness(C?) = ? C_4 = 100 = 570

1st generation (C?, C?, C?, C?): C_1 C_2 C_3 C_4

Pr(C?) = ? (4th) C_1 = 0
Pr(C?) = ? (3rd) C_4 = .297
Pr(C?) = ? (2nd) C_2 = .323
Pr(C?) = ? (1st) C_3 = .370

C? = ??????? → C? = ??????? C_5 = 000 C_7 = 010
C? = ??????? → C? = ??????? C_6 = 101 C_8 = 000

Fitness(C?) = ? C_8 = 000 → 0
Fitness(C?) = ? C_6 = 101 → 1210
Fitness(C?) = ? C_7 = 010 → 710
Fitness(C?) = ? C_5 = 000 → 0

2nd generation (?, ?, ?, ?) C_6 C_7 C_9 C_4

Pr(C?) = ? (4th) C_8 = 0
Pr(C?) = ? (3rd) C_5 = 0
Pr(C?) = ? (2nd) C_9 = .37
Pr(C?) = ? (1st) C_6 = .63

C? = ??????? → C? = ??????? C_9 = 100 C_{11} = 110
C? = ??????? → C? = ??????? C_{10} = 101 C_{12} = 001

Applying mutation on ~~100~~ 101

Fitness(C?) = ? C_9 = 570
Fitness(C?) = ? C_{10} = 570
Fitness(C?) = ? C_{11} = 1280
Fitness(C?) = ? C_{12} = 640

Final answer: C_{11} → 110

P(C)
C_1 = 0
$C_2 = \frac{640}{1920} = .333$
$C_3 = \frac{710}{1920} = .370$
$C_4 = \frac{570}{1920} = .297$

Crossover 1st Gen

2nd C_2 001 → mask 110 → 000 / 101
3rd C_4 100

2nd C_2 001 → 110 → 011 (with 010) / 000
1st C_3 010

Crossover 2nd Gen

1st 101 → 100 → 101 / 101
1st 101

1st 101 → 100 → 001 / 110
2nd 010