CS 4210 – Assignment #4

Maximum Points: 100 pts.

Bronco ID: |__|__|__|__|__|__|__|__|__|

Last Name: ______________________

First Name: ______________________

Note 1: Your submission header must have the format as shown in the above-enclosed rounded rectangle.
Note 2: Homework is to be done individually. You may discuss the homework problems with your fellow students, but you are NOT allowed to copy – either in part or in whole – anyone else's answers.
Note 3: Your deliverable should be a .pdf file submitted through Gradescope until the deadline. In addition, source code (.py files) should be added to an online repository (e.g., github) to be downloaded and executed later.
Note 4: All submitted materials must be legible. Figures/diagrams must have good quality.
Note 5: Please use and check the Blackboard discussion for further instructions, questions, answers, and hints.

1. [20 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 (1º and 2nd) in the 2nd generation

- Mutation: on the 3rd bit of the **first** chromosome 101 generated for the 3rd generation.
- Termination criteria: stop in the 3ª generation. **Return the best chromosome found until then.**

Summary of the optimization function:

$$\max \sum_{i=1}^{n} v_i x_i$$

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

Solution format:

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

Fitness(C_1) = ?	Pr(C_1) = ?
Fitness(C_2) = ? and weigh ?	Pr(C_2) = ?
Fitness(C_3) = ? and weight ?	Pr(C_3) = ?
Fitness(C_4) = ? and weight ?	Pr(C_4) = ?

Suppose C_2 and C_4 and C_2 and C_3 are selected for crossover:

C_2 = 00|1 → C_5 = ? C_2 = 00|1 → C_7 = ?

C_4 = 10|0 → C_6 = ? C_3 = 01|0 → C_8 = ? Apply mutation now on C_5, C_6, C_7, C_8 if specified.

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

…

3rd generation (?, ?, ?, ?)

…

2. [20 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 (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:
    - $\text{Outlook} = \mathit{Overcast} \rightarrow\ 010$
    - $\text{Outlook} = \mathit{Overcast} \vee \text{Rain} \rightarrow\ 011$
    - $\text{Outlook} = \mathit{Sunny} \vee\ \mathit{Overcast} \vee \text{Rain} \rightarrow\ 111$
    - $\text{Outlook} = (\mathit{Overcast} \vee\ \mathit{Rain}) \wedge (\text{Temperature} = \mathit{Hot}) \rightarrow\ 011100$
    - $\text{Outlook} = \mathit{Sunny} \wedge \text{Temperature} = \mathit{Hot}$ then $\text{PlayTennis} = \mathit{yes} \rightarrow 1001001$

- 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 generated by using the roulette wheel (simulation). In case of a tie, the newest chromosomes should have a higher rank.
- 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 1011000 generated for the 3rd generation.
- Termination criteria: accuracy = 1.0. **Return the corresponding chromosome.**

Solution format:

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

Fitness(C_1) = ?	Pr(C_1) = ?
Fitness(C_2) = ?	Pr(C_2) = ?
Fitness(C_3) = ?	Pr(C_3) = ?
Fitness(C_4) = ?	Pr(C_4) = ?

Suppose C_1 and are C_2 are selected for crossover:

C_1 = 100|1001 → C_5 = ?
C_2 = 010|0101 → C_6 = ?   Apply mutation now on C_5, C_6 if specified.

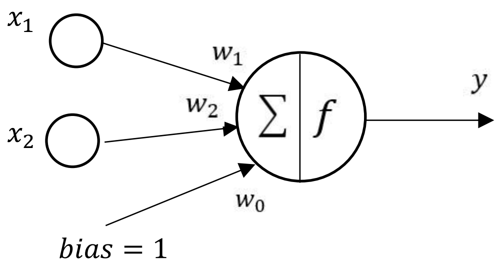
2nd generation (?, ?, ?, ?)

…

3rd generation (?, ?, ?, ?)

…

3. [20 points] Train the perceptron network below to solve the logical AND problem correctly classifying the dataset instances. Use the parameters: learning rate η=0.4, initial weights = 1, and activation function = heaviside.

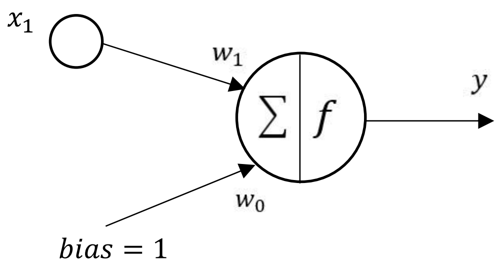


Datset

x_1	x_2	$x_1\ AND\ x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Solution format: Include the solution table (as illustrated in the lecture) with all variables and values calculated for each iteration. Hint: you can use an Excel spreadsheet to solve this problem and add your solution table here.

4. [20 points] Train the perceptron network below to solve the logical NOT problem correctly classifying the dataset instances. Use the parameters: learning rate η=0.1, initial weights = 0, and activation function = heaviside.



Datset

x_1	$NOT\ x_1$
0	1
1	0

Solution format: Include the solution table (as illustrated in the lecture) with all variables and values calculated for each iteration. Hint: you can use an Excel spreadsheet to solve this problem and add your solution table here.

5. [20 points] Complete the Python program (perceptron.py) that will read the file optdigits.tra to build a Perceptron classifier. You will simulate a grid search, trying to find which combination of two Perceptron hyperparameters (eta0, random_state) leads you to the best prediction performance. To test the accuracy of those distinct models, you will also use the file optdigits.tes. You should update and print the accuracy, together with the hyperparameters, when it is getting higher, printing at the end the highest accuracy and corresponding hyperparameters.

Important Note: Answers to all questions should be written clearly, concisely, and unmistakably delineated. You may resubmit multiple times until the deadline (the last submission will be considered).

NO LATE ASSIGNMENTS WILL BE ACCEPTED. ALWAYS SUBMIT WHATEVER YOU HAVE COMPLETED FOR PARTIAL CREDIT BEFORE THE DEADLINE!