

## 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.

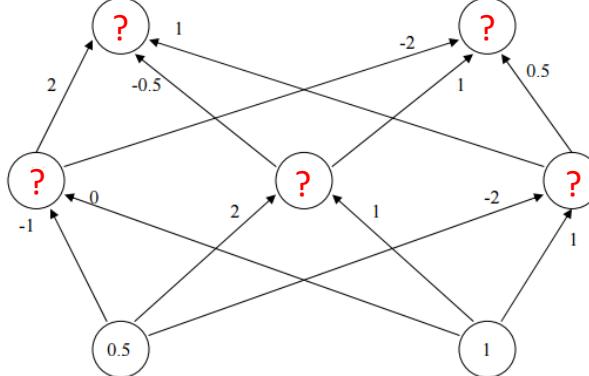
**Note 3:** Your deliverable should be a .pdf file submitted through Gradescope until the deadline. Do not forget to assign a page to each of your answers when making a submission. In addition, source code (.py files) should be added to an online repository (e.g., github) to be downloaded and executed later.

**Note 4:** All submitted materials must be legible. Figures/diagrams must have good quality.

**Note 5:** Please use and check the Canvas discussion for further instructions, questions, answers, and hints. The bold words/sentences provide information for a complete or accurate answer.

1. [30 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. [8 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  $(1, 2)$ . [2 points] Do you need to repeat the computations all over again? Why?
    - c. [8 points] Find the value at the output nodes of the network for the given input  $(0.5, 1)$  when the network uses the Sigmoid function as activation function.
    - d. [5 points] Find the value at the output nodes of the network for the input  $(1, 2)$  when the network uses the Sigmoid function as activation function. [2 points] Do you need to repeat the computations all over again? Why?

2. [15 points] Complete the Python program (perceptron.py) that will read the file optdigits.tra to build a Single Layer Perceptron and a Multi-Layer Perceptron classifiers. You will compare their performances and test which combination of two hyperparameters (learning rate and shuffle) leads you to the best prediction performance for each classifier. To test the accuracy of those distinct models, you will use the file optdigits.tes. You should update and print the accuracy of each classifier, together with the hyperparameters when it is getting higher.
3. [15 points] Deep Learning. Complete the Python program (deep\_learning.py) that will learn how to classify fashion items. You will use the dataset Fashion MNIST, which includes 70,000 grayscale images of  $28 \times 28$  pixels each, with 10 classes, each class representing a fashion item as illustrated below. You will use Keras to load the dataset which includes 60,000 images for training and 10,000 for test. Your goal is to train and test multiple deep neural networks and check their corresponding performances, always updating the highest accuracy found. This time you will use a separate function named `build_model()` to define the architectures of your neural networks. Finally, the weights of the best model will be printed, together with the architecture and the learning curves. To install TensorFlow use: `python -m pip install --upgrade tensorflow`.



4. [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. 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  $\langle$ Sunny, Overcast, Rain $\rangle$
  - Temperature  $\langle$ Hot, Mild, Cool $\rangle$

- 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<sub>1</sub>=1001001, C<sub>2</sub>=0100101, C<sub>3</sub>=1011000, C<sub>4</sub>=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 1<sup>st</sup> generation, two-point crossover with mask 0001100 in the 2<sup>nd</sup> 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:
  - (1<sup>st</sup> and 3<sup>rd</sup>) in the 1<sup>st</sup> generation
  - (1<sup>st</sup> and 2<sup>nd</sup>) in the 2<sup>nd</sup> generation
- Mutation: on the 6<sup>th</sup> bit of the chromosome(s) 1011000 selected/produced during the 2<sup>nd</sup> generation.
- Termination criteria: accuracy = 1.0. **Return the corresponding chromosome(s) – this will be your model.**

Solution format:

Fitness(C<sub>?</sub>) = ?

Fitness(C<sub>?</sub>) = ?

Fitness(C<sub>?</sub>) = ?

Fitness(C<sub>?</sub>) = ?

1<sup>st</sup> generation (C<sub>?</sub>, C<sub>?</sub>, C<sub>?</sub>, C<sub>?</sub>):

Pr(C<sub>?</sub>) = ? (4<sup>th</sup>)

Pr(C<sub>?</sub>) = ? (3<sup>rd</sup>)

Pr(C<sub>?</sub>) = ? (2<sup>nd</sup>)

Pr(C<sub>?</sub>) = ? (1<sup>st</sup>)

C<sub>?</sub> = ??????? → C<sub>?</sub> = ???????

C<sub>?</sub> = ??????? → C<sub>?</sub> = ???????

Fitness(C<sub>?</sub>) = ?

Fitness(C<sub>?</sub>) = ?

Fitness(C<sub>?</sub>) = ?

Fitness(C<sub>?</sub>) = ?

2<sup>nd</sup> generation (? , ?, ?, ?)

Pr(C<sub>?</sub>) = ? (4<sup>th</sup>)

Pr(C<sub>?</sub>) = ? (3<sup>rd</sup>)

Pr(C<sub>?</sub>) = ? (2<sup>nd</sup>)

$\Pr(C_?) = ?$  (1<sup>st</sup>)

$C_? = ??????? \rightarrow C_? = ???????$   
 $C_? = ??????? \rightarrow C_? = ???????$

Applying mutation on 1011000

Fitness( $C_?$ ) = ?

Final answer:

5. [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 1<sup>st</sup>, 2<sup>nd</sup>, and 3<sup>rd</sup> 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 1<sup>st</sup> generation, single-point crossover with mask 100 in the 2<sup>nd</sup> 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:
  - (2<sup>nd</sup> and 3<sup>rd</sup>) and (2<sup>nd</sup> and 1<sup>st</sup>) in the 1<sup>st</sup> generation
  - (1<sup>st</sup> and 1<sup>st</sup>) and (1<sup>st</sup> and 2<sup>nd</sup>) in the 2<sup>nd</sup> generation
- Mutation: on the 3<sup>rd</sup> bit of the chromosome(s) 101 selected/produced during the 2<sup>nd</sup> generation.
- Termination criteria: stop at the end of the 2<sup>nd</sup> 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: the same of question 4.

**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!**