BLG 252E - Object Oriented Programming
Assignment #1

Due: March, 26th 23:59

Introduction

For this assignment, you will develop an object-oriented program that simulates a university admission system. The admission program will use the perceptron algorithm, which is a classification algorithm that makes predictions based on a linear function ($wx + b$) combining a set of weights (w) with inputs (x) and bias (b), to decide whether a student might accepted to a certain university.

For any issues regarding the assignment, please contact Doğukan Arslan (**arslan.dogukan@itu.edu.tr**).

Implementation Notes

The implementation details given below will be considered for the grading:

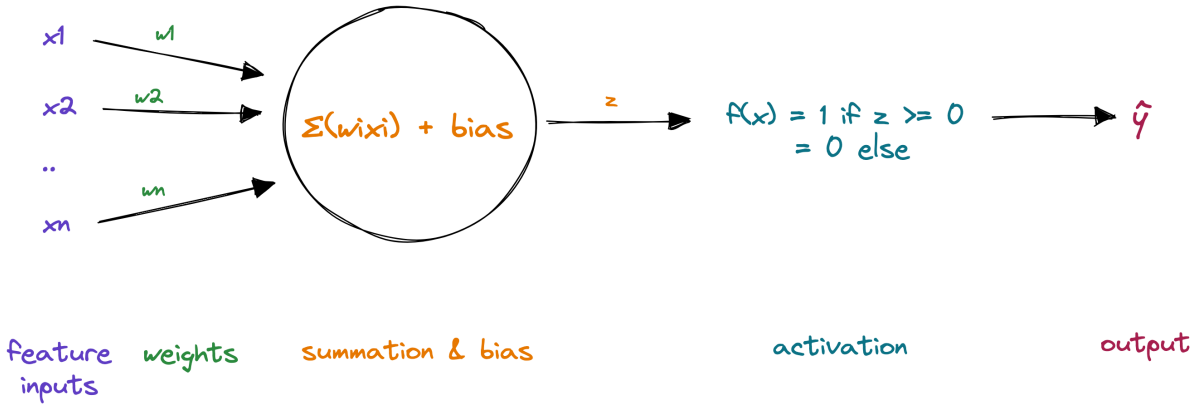
1. Follow a consistent coding style (indentation, variable names, etc.) with comments.
2. Follow suggested coding practices from the course documents.
3. Do not use any pre-compiled header files or STL commands.
4. This is an individual assignment and getting involved in any kind of cheating is subject to disciplinary actions.

Submission Notes

1. Your program should compile and run on the environment that is provided to you. Make sure that you have included all of your header files.
2. Make sure you tested your code using Valgrind and Calico. Include reports in the submission.
3. You should compress your files into an archive file (`.zip`) and only submit necessary files. Also, please write your name and ID on the top of each document that you will upload as in following format:
```
/* @Author
* Student Name:  <student_name>
* Student ID : <student_id>
*/
```
4. Submissions are made through **only** the Ninova system and have a strict deadline. Assignments submitted after the deadline will not be accepted. Do not risk leaving your submission to the last few minutes. Do not send your solutions by e-mail.

1 Perceptron

A perceptron is a type of artificial neural network that can be used for binary classification problems (like an e-mail is spam or not, if a patient has certain disease or not etc.). It is based on a single layer of neurons, each of which makes a decision based on the weighted sum of its inputs. The output of the perceptron is binary, either 0 or 1.



For instance, suppose we are predicting whether an email is spam or not based on the email's length in characters (x_1) and the number of exclamation marks (x_2) in the email. Also, our perceptron's weights are given as $[-0.1, 0.2]$ (so $w_1 = -0.1$ and $w_2 = 0.2$) and bias is 0.3. If an incoming email has 10 characters long ($x_1 = 10$) and it has 4 exclamation marks ($x_2 = 4$) in it, we can decide whether it is spam or not calculating;

$$z = x_1 w_1 + x_2 w_2 + b \tag{1}$$

$$z = 10 \cdot (-0.1) + 4 \cdot 0.2 + 0.3 \tag{2}$$

$$z = 0.1 \tag{3}$$

So, if $z = 0.1$ then, $f(x) = \tilde{y} = 1$ which means incoming email is spam.

2 Implementation Details

For this assignment, we will use the perceptron algorithm to whether a student can be accepted to a particular university, taking into account the student's GPA (Grade Points Average), GRE (Graduate Record Examinations) score, and TOEFL (Test of English as a Foreign Language) score.

2.1 Student Class

- `Student` *has a* **name** (eg., Doğukan), **GPA** (eg., 2.76), **GRE** score (eg., 163), **TOEFL** score (eg., 101), and **number of applications** (eg., 2).
- Initially, the name value is empty and GPA, GRE, TOEFL, and the number of application values are zero.
- Every time a student enters or leaves the system a message should appear.

2.2 University Class

- `University` *has a* **name** (eg., Istanbul Technical University), three **weight values** for GPA, GRE, and TOEFL (eg., [3.1, -0.1, 1.0]), corresponding to how much they prioritize these values, a **bias** value (eg., 1.6), and a **country** (eg., Türkiye).
- University *can* **evaluate a student**, based on the perceptron algorithm, using their GPA, GRE score, and TOEFL score.

- Initially, name and country values are empty and other values are random.
- Every time a student admitted or rejected a message should appear.

Warning: Make sure you use `const` modifier whenever necessary. Do not implement any function that is not required.

2.3 Main Function

`main.cpp` is provided along with this document. Please do not change anything. The output should be like:

Command Line

```
Welcome to the University Admission System!
Micheal logged in to the system.
Ross logged in to the system.
Amy logged in to the system.
Micheal logged in to the system.
Micheal is admitted to Scranton State University.
Ross is admitted to Central Perk College.
Ross is rejected from Brooklyn Institute of Technology.
Ross is rejected from McLaren's University.
Amy is admitted to Scranton State University.
Amy is rejected from McLaren's University.
Lily logged out of the system with 0 application(s).
Amy logged out of the system with 2 application(s).
Ross logged out of the system with 3 application(s).
Micheal logged out of the system with 1 application(s).
```

3 Report

Starting with the statement "university_array[3].evaluate_student(student3);", list the methods invoked by which objects and the arguments sent to those methods until the end of the program.

4 Test

4.1 Valgrind

Valgrind is a tool for memory debugging and memory leak detection. It is pre-installed your given Ubuntu environment. For more information, click **here**. Make sure that all heap blocks were freed in your code and no leaks are possible.

Run your executable with Valgrind:

```
valgrind --tool=memcheck --leak-check=full --show-leak-kinds=all ./bin/main |& tee valgrind_log.txt
```

Warning: Include your created log file in your submission.

4.2 Calico

Calico is a testing tool to check whether the program gives correct output for a given input. It is pre-installed your given Ubuntu environment. For more information, click **here**. Make sure that your code passes all the given test cases.

Run below command to test your code with given Calico file:

```
calico admission.yaml |& tee calico_log.txt
```

Warning: Include your created log file in your submission.