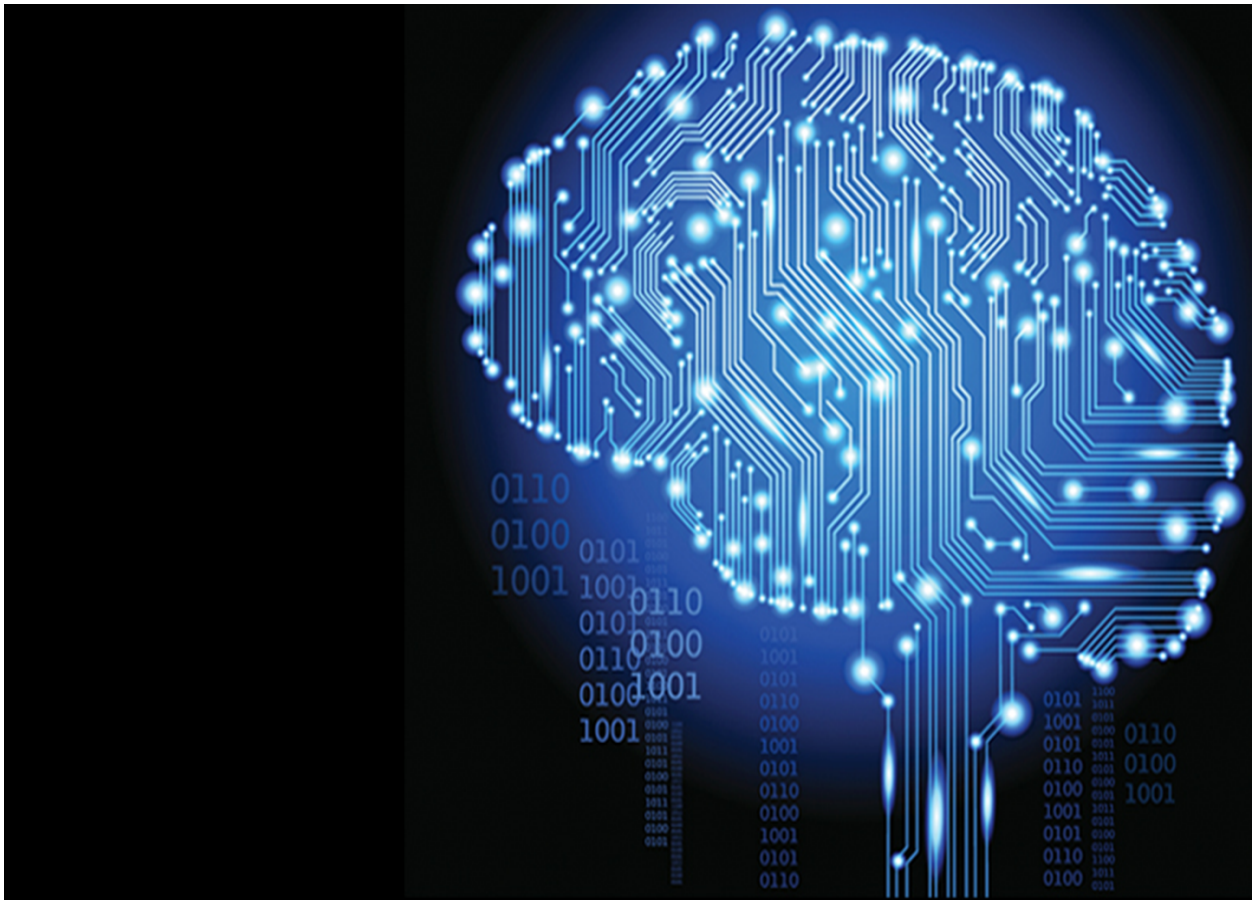
Computer Science 340: Artificial Intelligence

Project report

Instructor: Dr. Alexander Astaras

Written by: Enklid Hoxha

Dt: 04/12/2022



Introduction

This project is divided into two parts where part A asks for the implementation of a text interface menu with options based on the concepts of file reading and file writing; and part B asks for the implementation of a simple artificial neural network to solve a simple classification problem (further description about each part of the project follows below).

Part A

As above mentioned, part A asks for the implementation of a text interface menu with 6 options, where the first 5 options require some interaction with data inside a file. More specifically, the data for this project refers to the Grand Prix winners of the Formula1, 2018 season. The data is divided into 6 fields, namely: Country, Date, Name, Team, Laps, and Time. The requirements of each option go as follows:

1. Requires the program to display the data from the file to the screen
2. Requires the program to ask the user about a number, signifying the number of laps, based on which the program is to display on the screen in a sorted order only those records with a number of laps bigger or equal to the given number
3. Requires for the program to calculate an additional stat in the form of the average time per lap during each race, display this data on the screen as well as write this data in a file
4. Requires for the user to run option 3 first, and if the condition is satisfied then the program will ask the user for a sorting field and a sorting order, based on which the program is to display the data on the screen
5. Requires the plotting of a column chart which shows the seasonal average time per lap for every player
6. Option 6 serves as an Exit button

Part A Design

Part A is designed with the class “record” as its basis. Record as the name suggests serves as a record to hold all the information in one line of the file. A “reader” object would read the data from the file line by line and fill the records. Later these records are put in a table, and after various sortings or filterings as per the option’s requirements the table is displayed on the screen. Each time one of the options requires for the program to write something on a file, the “writer” will get the information from the reader and and after various changes it will write the data in the file as required

Program Requirements

For this program to run there is a need to download the following libraries:

- Numpy
- Matplotlib
- Prettytable

On top of the code there is a commented out block that if run will install the above-mentioned libraries.

```
"""import os
os.system('pip install numpy')
os.system('pip install prettytable')
os.system('pip install matplotlib')"""  # install commands for libraries used in this project
```

Text Interface

The first thing that the user will see upon running the program is the menu for the program which would appear as follows:

```
Hello and Welcome to my project for CS340, Part A
These are the options for the menu:



        1) Read data from the input file and display on a table

        2) Filter the previous data by the number of laps and display the sorted data

        3) Calculate a new stat. the average time per one lap, add the new information to the table and
           create an output file with the said information

        4) Using the output data file, sort the  information according to any of the seven fields in
           ascending or descending order

        5) Calculate the seasonal average time for lap for each player in the Grand Prix won by them and
           display the information in a graph

        6) Exit the Program


          Your Choice:
```

There is Error trapping applied throughout the program, and upon receiving an invalid input the program will notify the user through a message:

```
Hello and Welcome to my project for CS340, Part A
These are the options for the menu:



        1) Read data from the input file and display on a table

        2) Filter the previous data by the number of laps and display the sorted data

        3) Calculate a new stat. the average time per one lap, add the new information to the table and
           create an output file with the said information

        4) Using the output data file, sort the  information according to any of the seven fields in
           ascending or descending order

        5) Calculate the seasonal average time for lap for each player in the Grand Prix won by them and
           display the information in a graph

        6) Exit the Program


          Your Choice: f
 Not a Valid Input!! Please try again.
```

If the user would end up choosing option 1 in the menu then the table with the data from the file will be displayed as follows:

Country	Date	Name	Team	Laps	Time
Australia	25-Mar-18	Sebastian_Vettel	FERRARI	58	01:29:33.000
Bahrain	08-Apr-18	Sebastian_Vettel	FERRARI	57	01:32:01.940
China	15-Apr-18	Daniel_Ricciardo	RED_BULL	56	01:35:36.380
Azerbaijan	29-Apr-18	Lewis_Hamilton	MERCEDES	51	01:43:44.291
Spain	13-May-18	Lewis_Hamilton	MERCEDES	66	01:35:29.972
Monaco	27-May-18	Daniel_Ricciardo	RED_BULL	78	01:42:54.807
Canada	10-Jun-18	Sebastian_Vettel	FERRARI	68	01:28:31.377
France	24-Jun-18	Lewis_Hamilton	MERCEDES	53	01:30:11.385
Austria	01-Jul-18	Max_Verstappen	RED_BULL	71	01:21:56.024
Great_Britain	08-Jul-18	Sebastian_Vettel	FERRARI	52	01:27:29.784
Germany	22-Jul-18	Lewis_Hamilton	MERCEDES	67	01:32:29.845
Hungary	29-Jul-18	Lewis_Hamilton	MERCEDES	70	01:37:16.427
Belgium	26-Aug-18	Sebastian_Vettel	FERRARI	44	01:23:34.476
Italy	02-Sep-18	Lewis_Hamilton	MERCEDES	53	01:16:54.484
Singapore	16-Sep-18	Lewis_Hamilton	MERCEDES	61	01:51:11.611
Russia	30-Sep-18	Lewis_Hamilton	MERCEDES	53	01:27:25.181
Japan	07-Oct-18	Lewis_Hamilton	MERCEDES	53	01:27:17.062
United_States	21-Oct-18	Kimi_Raikkonen	FERRARI	56	01:34:18.643
Mexico	28-Oct-18	Max_Verstappen	RED_BULL	71	01:38:28.851
Brazil	11-Nov-18	Lewis_Hamilton	MERCEDES	71	01:27:09.066
Abu_Dhabi	25-Nov-18	Lewis_Hamilton	MERCEDES	55	01:39:40.382

If the user would instead choose the second option, then the program would ask for another value which will serve as the threshold for the Laps field. After the value is given then the program will display the records satisfying the condition, ordered in ascending order

```
          Your Choice: 2
you have chosen option 2....


 What is the number of laps you want to search by?  68


+---------+-----------+------------------+----------+------+--------------+
| Country |    Date   |       Name       |   Team   | Laps |     Time     |
+---------+-----------+------------------+----------+------+--------------+
|  Canada | 10-Jun-18 | Sebastian_Vettel | FERRARI  |  68  | 01:28:31.377 |
| Hungary | 29-Jul-18 |  Lewis_Hamilton  | MERCEDES |  70  | 01:37:16.427 |
| Austria | 01-Jul-18 |  Max_Verstappen  | RED_BULL |  71  | 01:21:56.024 |
|  Brazil | 11-Nov-18 |  Lewis_Hamilton  | MERCEDES |  71  | 01:27:09.066 |
|  Mexico | 28-Oct-18 |  Max_Verstappen  | RED_BULL |  71  | 01:38:28.851 |
|  Monaco | 27-May-18 | Daniel_Ricciardo | RED_BULL |  78  | 01:42:54.807 |
+---------+-----------+------------------+----------+------+--------------+
```

Option 3, similar to option 1 will display a table, with the only difference that the table will have an additional field in the form of “avg lap time”

```
+---------------+-----------+------------------+----------+------+--------------+--------------+
|    Country    |    Date   |       Name       |   Team   | Laps |     Time     | avg time lap |
+---------------+-----------+------------------+----------+------+--------------+--------------+
|   Australia   | 25-Mar-18 | Sebastian_Vettel | FERRARI  |  58  | 01:29:33.000 | 00:02:33.638 |
|    Bahrain    | 08-Apr-18 | Sebastian_Vettel | FERRARI  |  57  | 01:32:01.940 | 00:02:37.876 |
|     China     | 15-Apr-18 | Daniel_Ricciardo | RED_BULL |  56  | 01:35:36.380 | 00:02:42.435 |
|   Azerbaijan  | 29-Apr-18 |  Lewis_Hamilton  | MERCEDES |  51  | 01:43:44.291 | 00:02:02.450 |
|     Spain     | 13-May-18 |  Lewis_Hamilton  | MERCEDES |  66  | 01:35:29.972 | 00:01:27.818 |
|     Monaco    | 27-May-18 | Daniel_Ricciardo | RED_BULL |  78  | 01:42:54.807 | 00:01:19.164 |
|     Canada    | 10-Jun-18 | Sebastian_Vettel | FERRARI  |  68  | 01:28:31.377 | 00:01:18.108 |
|     France    | 24-Jun-18 |  Lewis_Hamilton  | MERCEDES |  53  | 01:30:11.385 | 00:02:42.102 |
|    Austria    | 01-Jul-18 |  Max_Verstappen  | RED_BULL |  71  | 01:21:56.024 | 00:01:09.240 |
| Great_Britain | 08-Jul-18 | Sebastian_Vettel | FERRARI  |  52  | 01:27:29.784 | 00:02:41.957 |
|    Germany    | 22-Jul-18 |  Lewis_Hamilton  | MERCEDES |  67  | 01:32:29.845 | 00:01:23.834 |
|    Hungary    | 29-Jul-18 |  Lewis_Hamilton  | MERCEDES |  70  | 01:37:16.427 | 00:01:23.378 |
|    Belgium    | 26-Aug-18 | Sebastian_Vettel | FERRARI  |  44  | 01:23:34.476 | 00:02:54.965 |
|     Italy     | 02-Sep-18 |  Lewis_Hamilton  | MERCEDES |  53  | 01:16:54.484 | 00:01:27.660 |
|   Singapore   | 16-Sep-18 |  Lewis_Hamilton  | MERCEDES |  61  | 01:51:11.611 | 00:02:49.371 |
|     Russia    | 30-Sep-18 |  Lewis_Hamilton  | MERCEDES |  53  | 01:27:25.181 | 00:02:39.966 |
|     Japan     | 07-Oct-18 |  Lewis_Hamilton  | MERCEDES |  53  | 01:27:17.062 | 00:02:39.812 |
| United_States | 21-Oct-18 |  Kimi_Raikkonen  | FERRARI  |  56  | 01:34:18.643 | 00:02:41.470 |
|     Mexico    | 28-Oct-18 |  Max_Verstappen  | RED_BULL |  71  | 01:38:28.851 | 00:01:23.223 |
|     Brazil    | 11-Nov-18 |  Lewis_Hamilton  | MERCEDES |  71  | 01:27:09.066 | 00:01:14.649 |
|   Abu_Dhabi   | 25-Nov-18 |  Lewis_Hamilton  | MERCEDES |  55  | 01:39:40.382 | 00:02:49.734 |
+---------------+-----------+------------------+----------+------+--------------+--------------+
```

For Option 4 there are 2 cases:

First case is when the user seeks to run option 4 before running option 3. In this case the user will be greeted by a message asking for option 3 to be run first.

```
Hello and Welcome to my project for CS340, Part A
These are the options for the menu:




        1) Read data from the input file and display on a table

        2) Filter the previous data by the number of laps and display the sorted data

        3) Calculate a new stat. the average time per one lap, add the new information to the table and
           create an output file with the said information

        4) Using the output data file, sort the  information according to any of the seven fields in
           ascending or descending order

        5) Calculate the seasonal average time for lap for each player in the Grand Prix won by them and
           display the information in a graph

        6) Exit the Program


         Your Choice: 4
To run this option you need to run option 3 first
```

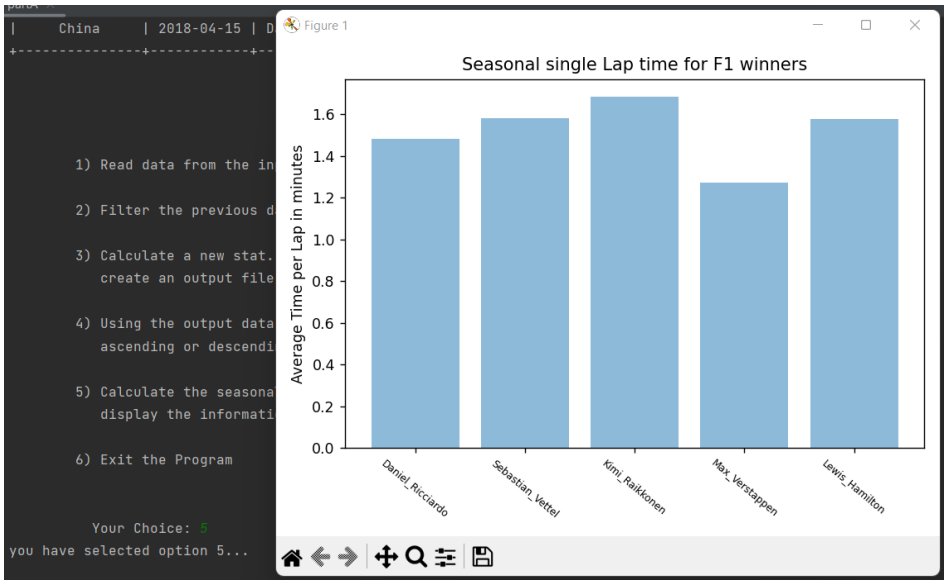
In case the user seeks to run option 4 after running option 3 then the program will display 2 additional menus asking for a sorting field and a sorting order:

```
         Your Choice: 4
you have selected option 4....


           Please select one of the following fields for sorting:
           1) Country
           2) Date
           3) Name
           4) Team
           5) Laps
           6) Time
           7) Average Lap Time
 Your choice: 3

       PLease select an sorting order:

       1) Ascending
       2) Descending
 Your choice: 2
```

If the inputs are valid (like in the case displayed in the picture above), then the program will display a table sorted based on the selected field in the selected order

Country	Date	Name	Team	Laps	Time	Avg Lap Time
Great_Britain	2018-07-08	Sebastian_Vettel	FERRARI	52	01:27:29.784	00:02:41.957
Canada	2018-06-10	Sebastian_Vettel	FERRARI	68	01:28:31.377	00:01:18.108
Belgium	2018-08-26	Sebastian_Vettel	FERRARI	44	01:23:34.476	00:02:54.965
Bahrain	2018-04-08	Sebastian_Vettel	FERRARI	57	01:32:01.940	00:02:37.876
Australia	2018-03-25	Sebastian_Vettel	FERRARI	58	01:29:33.000	00:02:33.638
Mexico	2018-10-28	Max_Verstappen	RED_BULL	71	01:38:28.851	00:01:23.223
Austria	2018-07-01	Max_Verstappen	RED_BULL	71	01:21:56.024	00:01:09.240
Spain	2018-05-13	Lewis_Hamilton	MERCEDES	66	01:35:29.972	00:01:27.818
Singapore	2018-09-16	Lewis_Hamilton	MERCEDES	61	01:51:11.611	00:02:49.371
Russia	2018-09-30	Lewis_Hamilton	MERCEDES	53	01:27:25.181	00:02:39.966
Japan	2018-10-07	Lewis_Hamilton	MERCEDES	53	01:27:17.062	00:02:39.812
Italy	2018-09-02	Lewis_Hamilton	MERCEDES	53	01:16:54.484	00:01:27.660
Hungary	2018-07-29	Lewis_Hamilton	MERCEDES	70	01:37:16.427	00:01:23.378
Germany	2018-07-22	Lewis_Hamilton	MERCEDES	67	01:32:29.845	00:01:23.834
France	2018-06-24	Lewis_Hamilton	MERCEDES	53	01:30:11.385	00:02:42.102
Brazil	2018-11-11	Lewis_Hamilton	MERCEDES	71	01:27:09.066	00:01:14.649
Azerbaijan	2018-04-29	Lewis_Hamilton	MERCEDES	51	01:43:44.291	00:02:02.450
Abu_Dhabi	2018-11-25	Lewis_Hamilton	MERCEDES	55	01:39:40.382	00:02:49.734
United_States	2018-10-21	Kimi_Raikkonen	FERRARI	56	01:34:18.643	00:02:41.470
Monaco	2018-05-27	Daniel_Ricciardo	RED_BULL	78	01:42:54.807	00:01:19.164
China	2018-04-15	Daniel_Ricciardo	RED_BULL	56	01:35:36.380	00:02:42.435

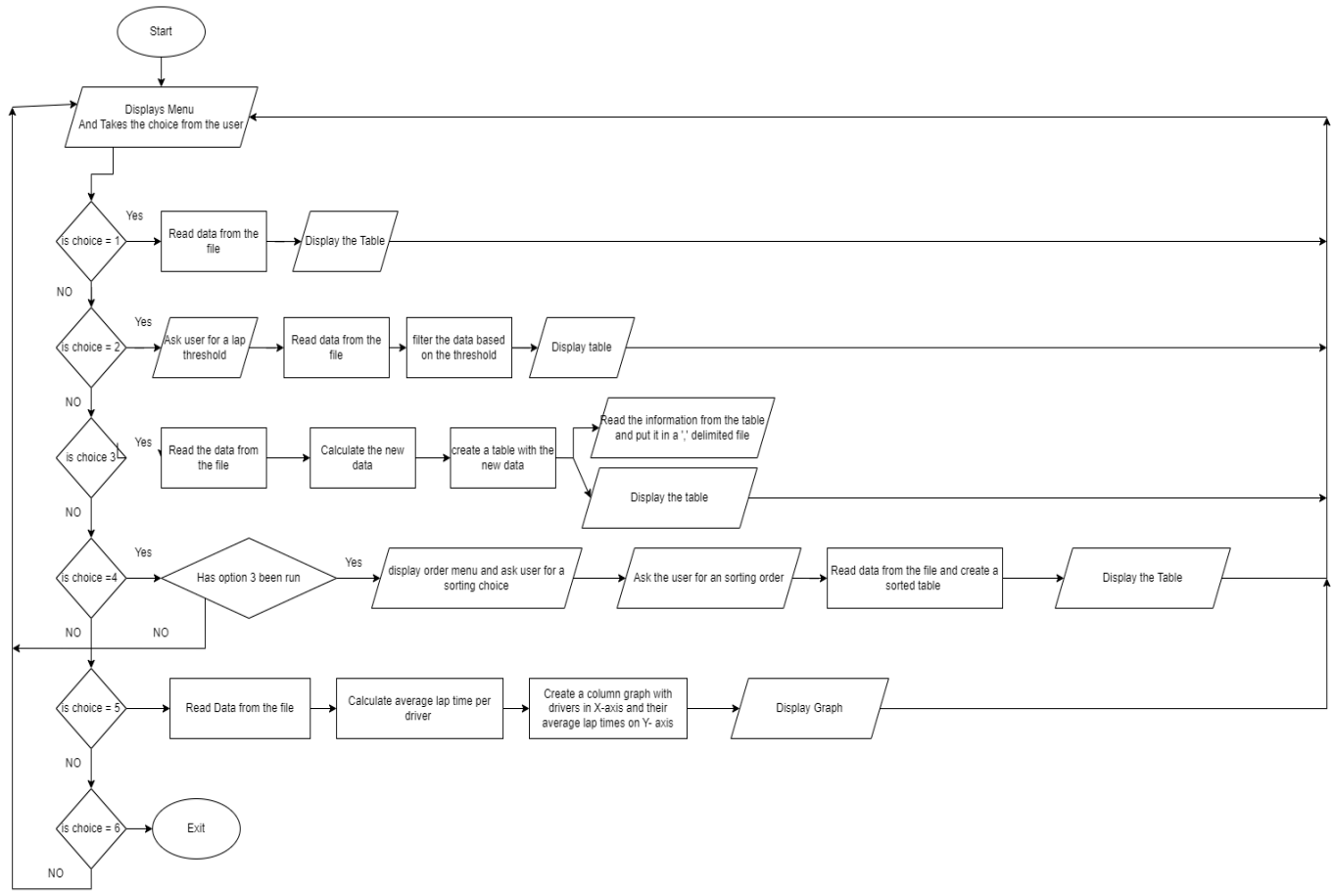
For Option % the user will be greeted with a pop up window showing a graph displaying the seasonal average time per lap for each racer



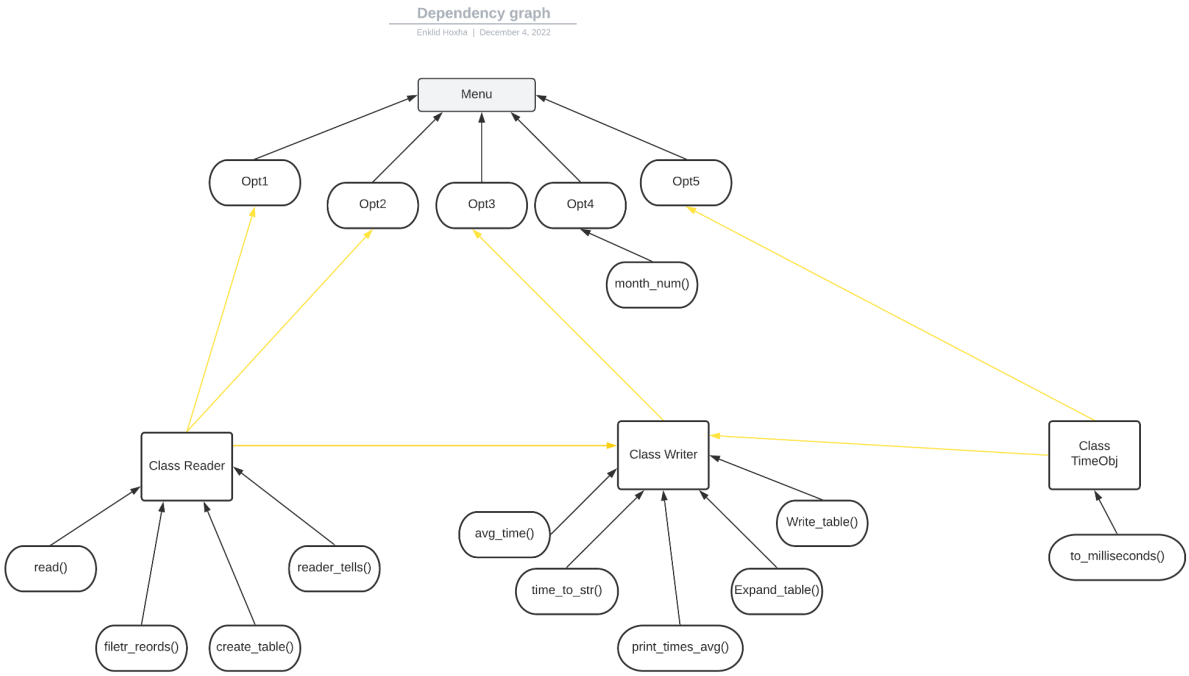
If the user selects option 6, then the program will display a ‘Goodbye’ message and exit

```
Your Choice: 6

  Thank you for running this program
  Hopefully we will meet again
  Good Bye!
```

To facilitate the better understanding of the code for part A, attached below will be a simple flowchart and the function dependency diagram



Flowchart



Explanation of the dependency diagram.

As mentioned in the Design section for part A there will be 5 main options (one for each menu). Each of these options is accessible from the menu. Supporting these options are the following classes (Reader, Writer, and TimeObj). Each of these classes have methods which are utilized in the main options. Additionally, there is some form of hierarchy between the classes as well. Case in being: the “writer” class utilizes the methods from the reader class and the TimeObj class as well

Part B

Part B describes a simple classification problem where through an artificial neural network with 3 layers (Input, Hiddel, Output) the program is to classify a series of unary numbers into bigger or smaller than 4, as well as even or odd. Towards this end a training database is fed to the ANN,the latter is then put in a training cycle of X amount of epochs given by the user. After the training is completed the ANN is expected to be able to provide the right classification with a high accuracy for imputed unary number up to 9 and possibly more.

All of this process is to be provided to the user divided into steps, each step implemented in one of the options in the menu below:

1. The user can give a topology (However because of the type of the training data and the expected output this includes only the size of the hidden layer)
2. Option 2 asks the user for a training file, a learning rate, and a number of epochs. Which upon being received start the training phase for the ANN
3. In option 3 the program reads a text file with randomly ordered unary numbers up to 9 and gives back a file with the unary numbers and the predicted classifications
4. In option 4 the program shows a graph which will display the loss descend during the training phase

As per the requirements, other than the number of epochs which is to be provided by the user in every case, the other variables are set to have default values as follows:

Topology: 10-14-2

Learn Rate: 4.5

Training file: Training_file.txt

Specifics about the ANN

Since the ANN model for this project does not incorporate biases, in order for it to reach a satisfactory level of accuracy it needs a bigger learning rate than what is considered the norm. And while for various combinations the best learning rate also changes, throughout

the testing phase it was noticed that a learning rate of 4.5 (as opposed to 0.1 or even 0.01 which is the norm) performs moderately well in most of the scenarios. There is ofcourse a concern that a large learning rate will skip the most optimal point in the training path, however this concern is partially lifted through introduction of a decaying factor of 1% for the learning rate which applies every 50 training epochs.

Another feature of the program is that everything is zeroed everytime the user selects a new topology for the ANN. Thus the user is given a choice, which can be used to take the same ANN through multiple training phases (A very interesting option is to train the ANN for smaller number of epochs in each training phase while alsointroducing a smaller learning rate in each training phase)

Interface

Upon running the program the user is greeted with the menu for this part of the project

```
1) Enter network topology
2) Initiate a training pass
3) Classify test data
4) Display training result graphics
5) Exit

Please choose an option:
```

Similar to Part A Part B also has error trapping implemented upon receiving user input

```
1) Enter network topology
2) Initiate a training pass
3) Classify test data
4) Display training result graphics
5) Exit

Please choose an option: f
Not a Valid Input!!
```

Upon selecting option 1 the program asks the user to give a size for the hidden layer

```
1) Enter network topology
2) Initiate a training pass
3) Classify test data
4) Display training result graphics
5) Exit

Please choose an option: 1
you have selected option 1, '
the default topology for this network is set to 10-14-2
Size of the Hidden Layer:
```

The user can choose to give a value or press enter and let the default value be applied

Upon selecting option 2 the program asks the user for the training file, learning rate and number of epochs

```
1) Enter network topology
2) Initiate a training pass
3) Classify test data
4) Display training result graphics
5) Exit

Please choose an option: 2
you have selected option 2
What is the training file for this run:
Learning Rate: 4.5
Number of Epochs: 1000
```

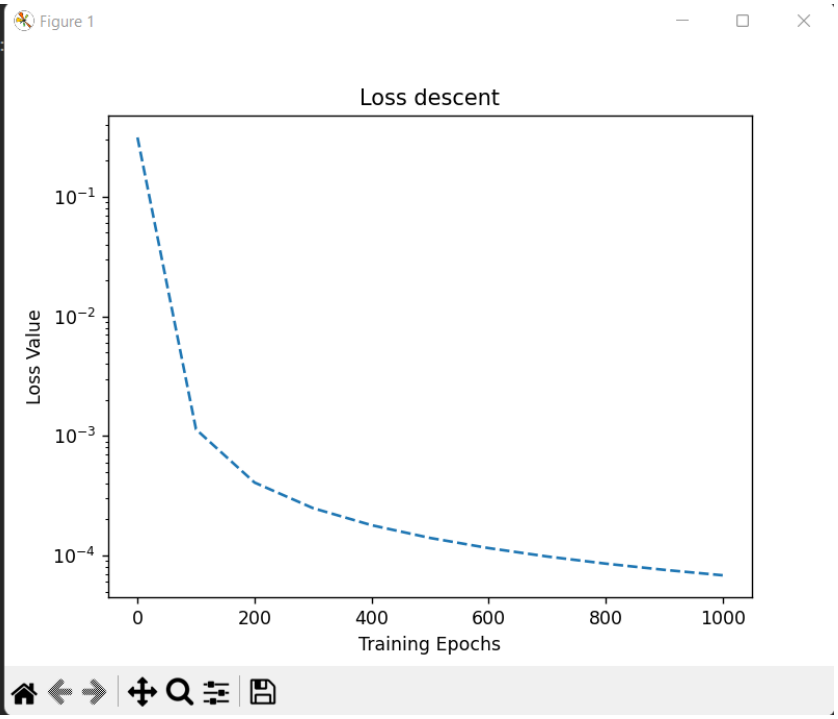
For the training file and learning rate the user can choose to leave the default values in place but a value for the training epochs has to be provided

Option 3 simply displays a confirmation message that the program has red the input file and has created an output file with the results. This file is available upon the termination of the program

```
1) Enter network topology
2) Initiate a training pass
3) Classify test data
4) Display training result graphics
5) Exit

Please choose an option: 3
the file: input_data.txt has been fed to the network
the results can be found in training_outputs.txt
```

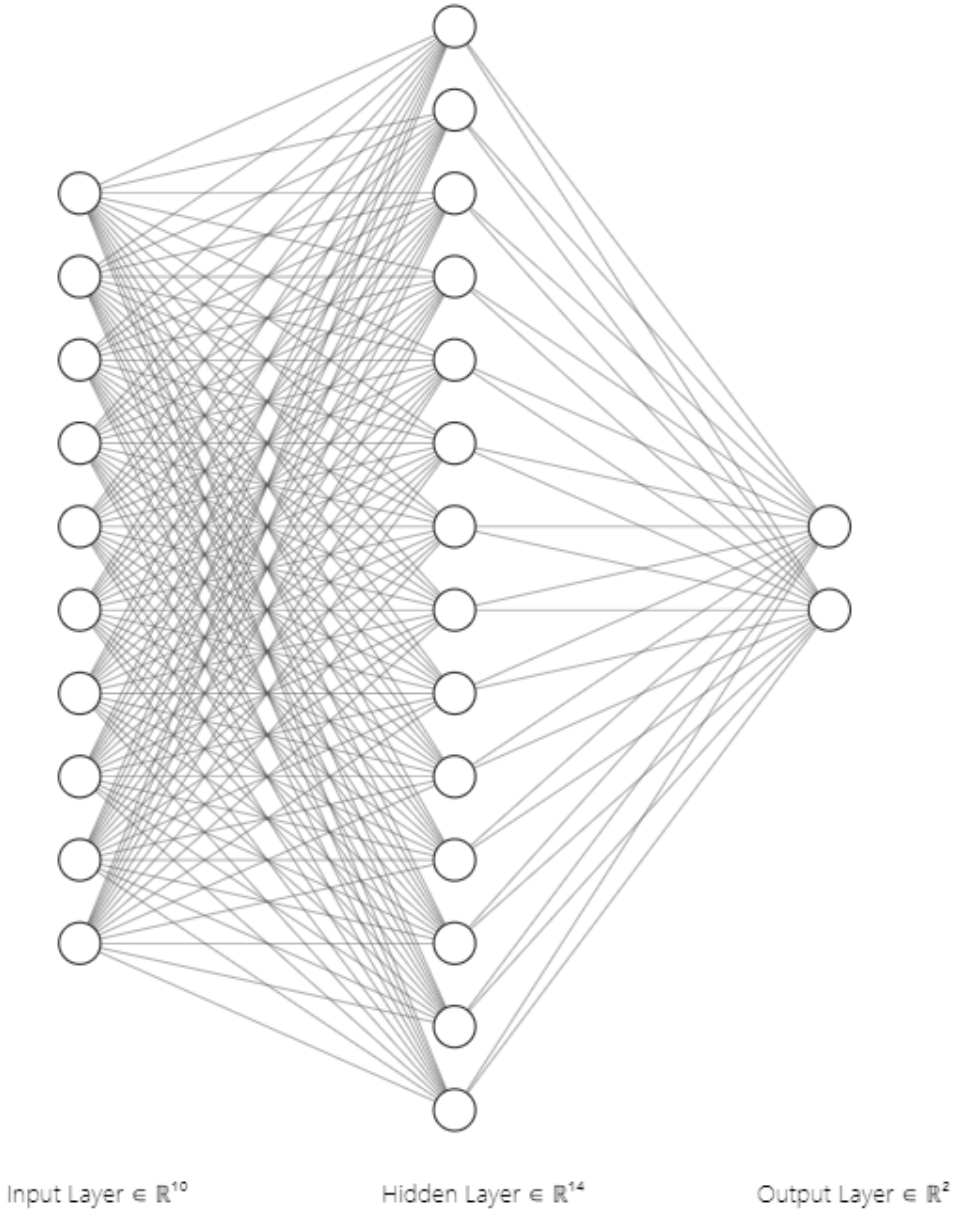
It is advised for option 4 to be run only once per topology as the graph produced after multiple training phases is rather unorthodox; however it will be explained in a sector further down the report

```
you have selected option 2
What is the training file for this run:
Learning Rate:
Number of Epochs: 1000

Loss: 0.31365829264869827
Loss: 0.001134346289566066
Loss: 0.00040746647765739204
Loss: 0.0002488180920768543
Loss: 0.00017919177670192276
Loss: 0.0001401888577899467
Loss: 0.00011530778420304134
Loss: 9.808546380790386e-05
Loss: 8.547458304248833e-05
Loss: 7.585203372999046e-05
Loss: 6.827474733556677e-05

1) Enter network topology
2) Initiate a training pass
3) Classify test data
4) Display training result graphics
5) Exit

Please choose an option: 4
you have selected option 4
```



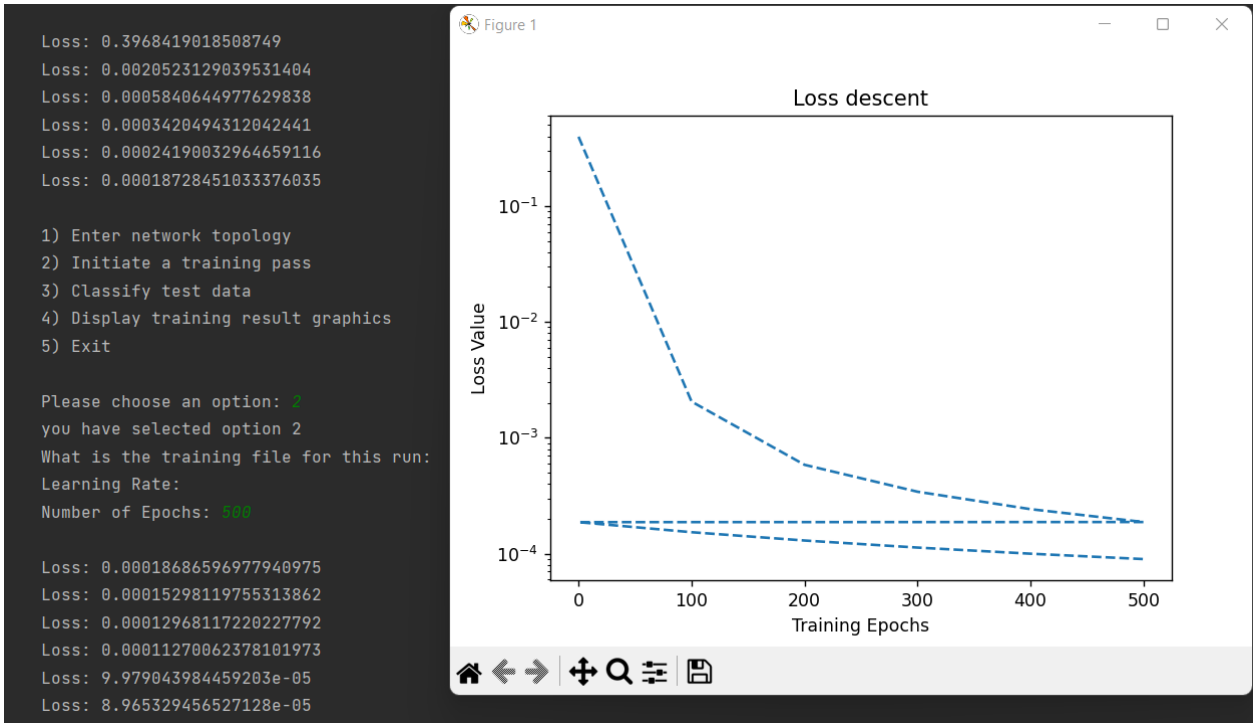
ANN diagram

Following is the ANN diagram showing neuron interconnectivity for the default topology 10-14-2



Experimenting with ANN

Earlier it was mentioned how producing of a loss descent diagram after multiple training phases will result in an unorthodox diagram (example down below)

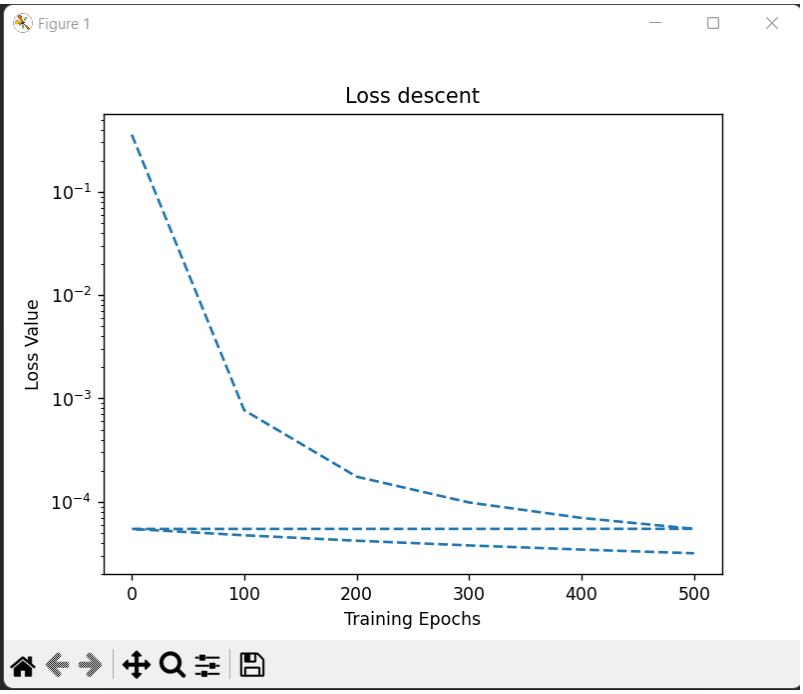


On itself this diagram is also quite interesting as it allows us to observe the curve of the loss descent in the first 500 training epochs and the later 500 training epochs (the two lines starting from X = 0 ignoring for this case the line connection between them)

Through experimenting with this diagram we can compare the results and find a function which would give us a function for approximating the decaying factor of the learning rate

For example:

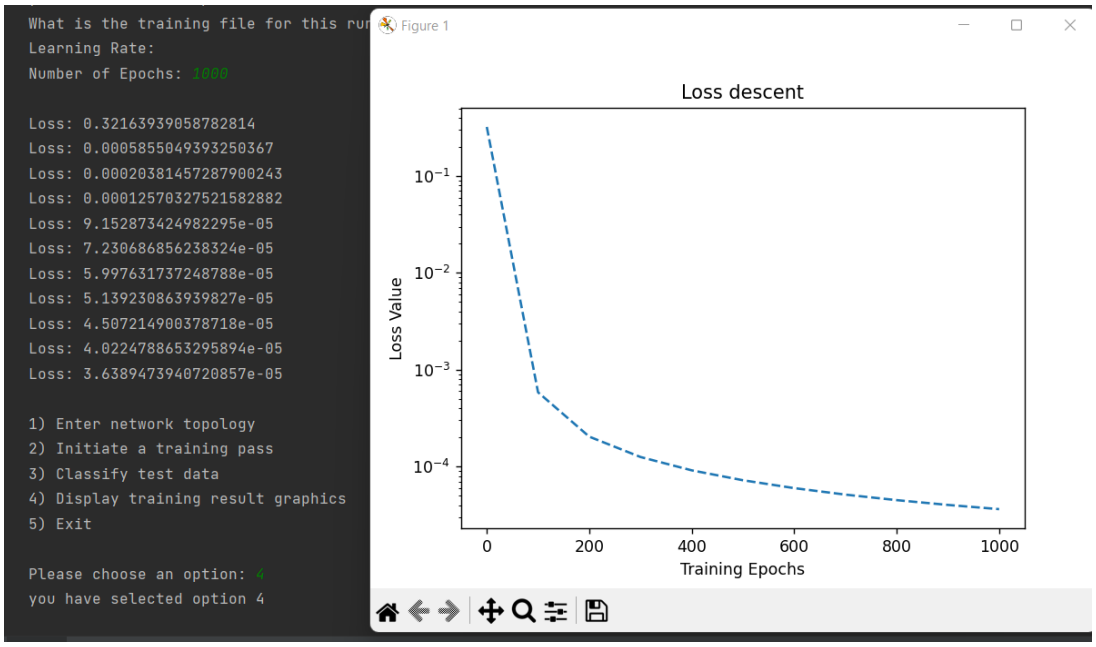
```
Loss: 0.0007688790658241776
Loss: 0.0001749505227697894
Loss: 9.848031377627511e-05
Loss: 6.983279657206343e-05
Loss: 5.4759078809027265e-05

1) Enter network topology
2) Initiate a training pass
3) Classify test data
4) Display training result graphics
5) Exit

Please choose an option: 2
you have selected option 2
What is the training file for this run:
Learning Rate: 3
Number of Epochs: 500

Loss: 5.46443687997711e-05
Loss: 4.7411949519521546e-05
Loss: 4.202018591050524e-05
Loss: 3.783614222490599e-05
Loss: 3.449014048996999e-05
Loss: 3.1750875455558496e-05
```



In this second trial the first 500 epochs were conducted with an initial learn rate of 4.5 while the last 500 epochs were conducted with a learn rate of 3. As compared to the previous case shown earlier we can see a very slight difference in the end result. However this slight difference is enough to confirm that introducing a new, slightly lower initial learning rate in the middle of the 2 training phases can improve the end result. The difference in the curve is not really visible but this is mostly because the graphs were generated with a small number of epochs meaning that it is harder to spot a difference on the curve.

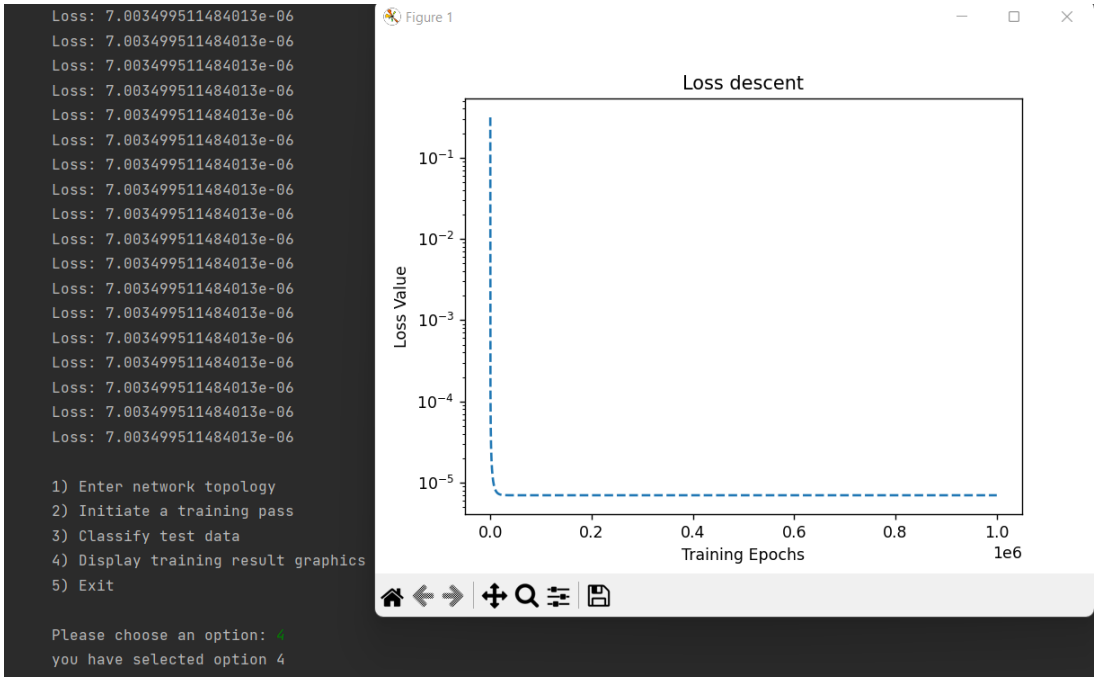
Other interesting Experiments include the training for a large amount of epochs or using a large hidden layer.

On the first experiment, using a large hidden layer while maintaining the same learning rate and number of epochs the ANN produced the following results



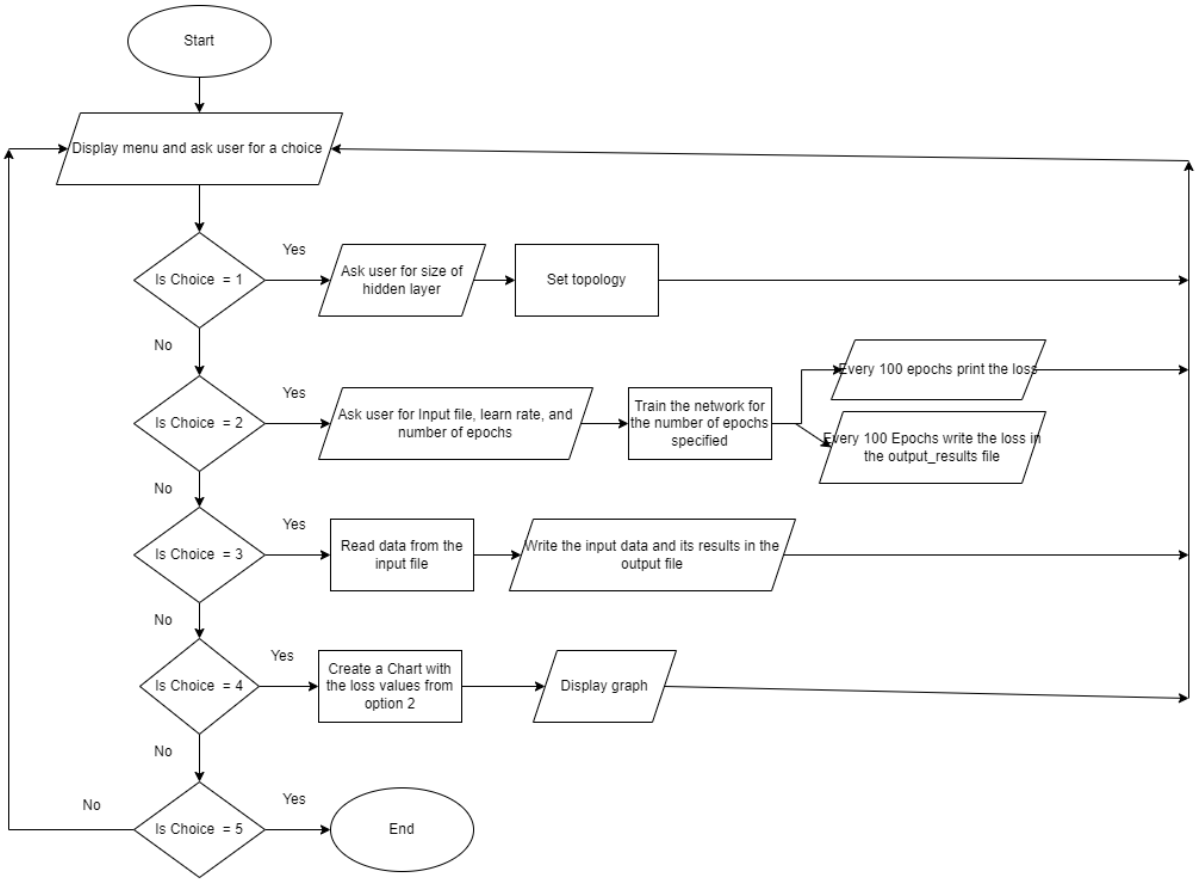
First thing to notice is that there is not a big difference between the default case trained for 1000 epochs and this case where the hidden layer is comprised of 100 neurons

However, the second experiment, training for a large number of epoch proved more interesting.

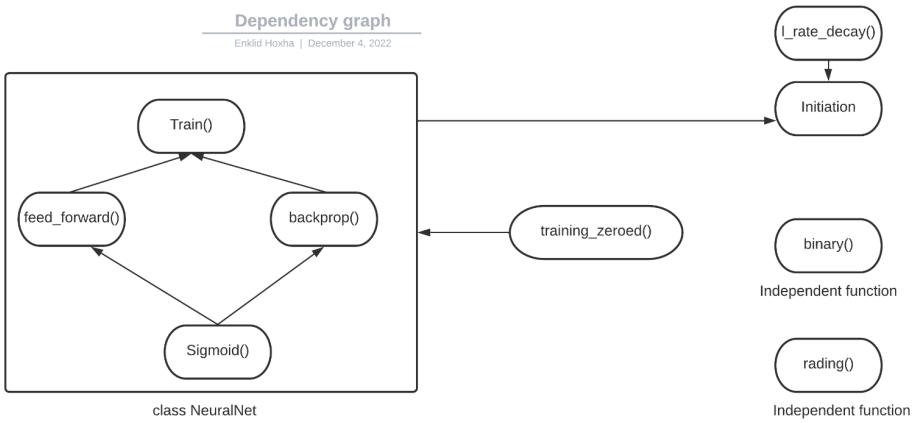


The end result is definitely better than the standard case, however more interesting is the curve for the loss descent. It can be seen that after a certain number of epochs there is no big improvement from the ANN (this result was produced by training for 1 million epochs)

Flowchart and Function dependency diagram for part B



flowchart



Function dependency diagram

Future Improvements

Trying to develop an ANN has been a challenge filled with many discoveries, and undoubtedly there are many things that can be improved upon.

1. Introduction of Biases - This would be a qualitative improvement giving the ANN another tool to sharpen its calculations, resulting in better predictions in fewer training epochs
2. Further studying the Decaying factor for the learning rate - If this is coupled with the first point mentioned in this list then one would end up with a perfectly equipped ANN
3. Introduction of more hidden layers for deep learning - Although the qualitative and quantitative improvements through this method are still unknown, this step would give the ANN more power to possibly handle more complicated problems

Reference list

activity.act.edu. (2022). *ACTivity Learning Management System: Log in to the site*. [online] Available at: https://activity.act.edu/lms/course/view.php?id=34 [Accessed 4 Dec. 2022].

Dertat, A. (2017). *Applied Deep Learning - Part 1: Artificial Neural Networks*. [online] Medium. Available at: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6.

Lee, K.D. (2014). *Python Programming Fundamentals*. London Springer London.

Nicosia, G., Ojha, V., Emanuele La Malfa, Jansen, G., Sciacca, V., Panos Pardalos, Giuffrida, G. and Umeton, R. (2021). *Machine Learning, Optimization, and Data Science*. Springer.

www.youtube.com. (n.d.). *Neural Networks from Scratch - P.1 Intro and Neuron Code*. [online] Available at: https://www.youtube.com/watch?v=Wo5dMEP_BbI&t=40s [Accessed 4 Dec. 2022].