**Level 5: AI Game Programming**

**Week 2**

**Lab 2**

RECORD ALL THAT YOU HAVE DONE IN YOUR LAB LOGBOOK (HARDCOPY ‘OR’ WORD FILE).

As explained in the lecture, just using and running other people's code/functions/libraries will not make you a good 'programmer'. Hence, in this unit, we will concentrate on fundamentals.

Of course, this will take longer, more difficult and more painful than just using ready-made software and almost instantly seeing wonderful results, pictures, etc. But it will provide you with a solid foundation so that you can, in the future, implement new methods and algorithms by yourself. You do not need to wait until other ‘real’ programmers write code to implement them for you.

*Solutions will be provided and explained ‘after’ you have done the experiments and tried to figure out the solutions by yourself. This would normally be in the following lecture or lab. This is the most effective way. Otherwise, you will not learn much.*

**Nuts and bolts - Some computational issues in AI programming**

In AI programming, you cannot avoid implementing mathematical equations or methods. All coding experience and expertise are useless if one cannot interpret algorithms and related mathematical equations.

Many AI programs fail because of programmers' tiny mistakes in implementing mathematical functions or programmers' lack of knowledge in computational issues.

In these labs, we will visit some mathematical and computational examples that are useful when you program AI algorithms.

**Notes:**

* **Focus on Fundamentals**: These tasks are designed to reinforce your understanding of the core building blocks of neural networks.
* **Experiment and Analyse**: Don't just implement the code; experiment with different parameters and analyse the results to gain deeper insights.
* **Use Libraries Wisely**: While libraries like TensorFlow, PyTorch, etc. can automate many aspects, it's crucial to understand the underlying concepts by performing manual calculations and comparisons.
* **Seek Clarification**: If you encounter any difficulties or have questions, don't hesitate to ask for assistance.

**1. Basic ANN activation functions**

In artificial neural networks (ANNs), there are many different functions which can be used to produce the outputs of neurons. These are called activation functions (you will learn more about this in the lecture). (You may have tried some of these functions in the math unit last year.)

**Task 1.1** Use a programming language of your choice (Python in Google Colab is recommended), write a function or a short piece of code to implement the following activation (mathematical) functions. Then plot the relevant input-output graph

Given *X* = input, *Y* = output and θ = 0.2 is a threshold. All of these variables are real number (float, not integer):

**Sign function**

**Step function**

**Hint:** When implementing, be aware of the X value when the Y is abruptly changed from 0 to 1 or -1 to +1.

**Sigmoid function**

NB: is a mathematical constant called Euler's number or natural number which is approximately equal to 2.71828.

**Task 1.2 Testing**

How would you know that your code is the correct implementation of the functions in Task 1.1? Develop a method to check and test.

**Task 1.3** **Conduct research and implement the following ANN activation functions**:

ReLU

Leaky ReLU

Linear function

Tanh (Hyperbolic Tangent)

**Task 1.4 Visualise the activation functions:**

* For each implemented function, generate a plot that illustrates its input-output relationship.
* Analyse the plotted graphs to understand the characteristics and effects of each activation function on neuron output.

**Task 2: Dissecting the neuron’s behaviour**

A single artificial neuron in an Artificial Neural Network (ANN) is a mathematical model inspired by biological neurons. It takes one or more inputs, processes them, and produces an output. (See more details in the lecture note.)

A diagram of a diagram

Description automatically generated

If the activation function is a Step function, here is the equation describing the neural model:

A mathematical equation with numbers and symbols

Description automatically generated

Here's a breakdown of the process:

* **Weighted Summation**: Each input to the neuron is multiplied by a corresponding weight, and these products are summed together. This can be represented as:
* **Activation Function**: The biased sum is then passed through an activation function. This function introduces non-linearity, enabling the neuron to learn complex patterns. The activation function determines whether the neuron should "fire" (produce a significant output) based on the input.
* **Implement the core logic of a single artificial neuron:**
  + Write code that simulates the weighted sum of inputs, incorporating bias.
  + Apply an activation function (from Task 3) to the neuron's weighted sum.
  + Experiment with different input values, weights, biases, and activation functions to observe their impact on the neuron's output.

**Task 3: Revisit Google Colab**

NB: Most of you have used Google Colab in the math unit last year already.

Python is a commonly used programming language in AI due to its flexibility, extensive libraries, and easy readability. Therefore, it is impossible to avoid Python. Later, towards the end of the unit, you will have a chance to try some complicated AI models, which, of course, are in Python. (Not now, so don’t panic!)

Therefore, it is useful to get familiar with the language and programming environment. Please note that this is not a Python programming unit. We will support you as much as you can with some examples and explanations, but you need to use parts of your 200 hours allocated for the unit to learn the language in parallel independently.

Don't feel you need to be a Python expert on day one! The programming examples are meant to supplement the conceptual foundations. We're here to support you every step of the way. Take it step-by-step, learn by doing, and leverage the many online Python tutorials available. With some dedicated practice time, you'll gain proficiency with this versatile language.

*Google Colab* is the Python programming environment chosen for this unit. Please use this lab to familiarise yourself with Colab.

**Google Colab**

This section will introduce students to Google Colab, a web-based platform for writing, running, and sharing Python code. Students will learn how to create a Colab notebook, write and execute Python code, and collaborate with others on Colab notebooks.

**Objectives**

* Learn what Google Colab is and its benefits.
* Create a Google Colab notebook.
* Write and run basic Python code in Google Colab.

**Prerequisites**

* A Google account.

**What is Google Colab?**

Google Colab is a free, cloud-based platform that allows you to write, run, and share Python code in your web browser. It provides you with access to powerful computing resources, including GPUs, without having to install any software on your local machine.

**Benefits of Google Colab**

* Free to use.
* No software installation required.
* Powerful computing resources.
* Easy to share and collaborate on notebooks.

**Instructions**

1. Go to Google Colab: <https://research.google.com/colaboratory/>.

Tip: It is more convenient to use Chrome browser and log into your Google account.

1. Click New Notebook to create a new notebook.
2. Your new notebook will open in a new tab.

A screenshot of a computer

Description automatically generated

1. Give your notebook a name and click Create. Just type your new name over the default name.

A screenshot of a computer

Description automatically generated

**Writing and Running Python Code in Google Colab**

To write Python code in Google Colab, type your code into the cells of your notebook. Each cell can contain a single line or multiple lines of code.

You can create a cell for code by clicking ‘+Code’ button. The ‘+Text’ button is for creating a cell to write text.

To run a cell of code, click the Run button (black circle with a triangle in the middle) in the toolbar (or press Shift+Enter). The output of the code will be displayed below the cell.

**Examples**

Here are some examples of how to write and run Python code in Google Colab.

A screenshot of a computer

Description automatically generated

Here is another example.

A screenshot of a computer

Description automatically generated

For more examples, please follow the following link:

<https://colab.research.google.com/?utm_source=scs-index>

**Task**

Use Python resources and tutorials listed below, convert your code in Task 1 into Python and run in Google Colab.

**Conclusion**

Google Colab is a powerful tool that can be used for a variety of tasks, including machine learning, data science, and scientific computing. It is easy to use and provides access to powerful computing resources without having to install any software on your local machine.

**Additional Exercises**

* Try writing some more Python code in your notebook and running it.
* Experiment with different types of data, such as numbers, strings, and lists.
* Try using some of the built-in Python functions, such as max(), min(), and sum().
* Try sharing your notebook with a friend or classmate and collaborating on it together.

**Tips**

* To learn more about Python, see the Python Tutorial:
  + <https://www.python.org/about/gettingstarted/>
  + <https://www.w3schools.com/python/>
* To learn more about Google Colab, see the Google Colab Documentation: https://research.google.com/colaboratory/.
* If you get stuck, don't be afraid to ask for help from your instructor or a classmate.

**For Students Who Are New to Python programming**

If you are new to Python, here are some additional tips:

* Start by learning the basics of Python syntax, such as variables, data types, and operators.
* Once you have a good understanding of the basics, you can start learning about more advanced Python topics, such as functions, classes, and modules.
* Practice writing Python code regularly. The more you code, the better you will become at it.
* Start with simple code and gradually work your way up to more complex code.
* Use comments to explain your code.
* Break your code down into smaller functions.
* Test your code as you go.
* Don't be afraid to ask for help.

**END**

**------------------------------------------------------------------------**