AIGP Assignment Advice and Hints

General Advice

-Don't panic. It is not as difficult as you might think!
-Please email me if you need extra help. Don't keep the problems to yourself and be stressed.
-I'm here to help.

I'll be an extremely happy man if you pass. I don't want to fail any students. However, I have to maintain academic standards. Otherwise, your grades and degree mean nothing. Therefore, please help me by doing some work so that I can give you some marks!

In AIGP, the lectures, labs and the assignment are carefully designed to make you fully equipped with important intellectual skills required for a real programmer at the degree level, different from other general coders. So, it is not just writing a routine piece of code.

Marking Criteria, Rubric and Tasks
Please carefully pay attention to the Marking criteria, Rubric and Tasks in the assignment brief. I will mark your assignment according to those elements. Therefore, please make sure that you cover all of them.

For example, if you concentrate only on the coding part of the Assignment A1 (12%) and completely forget about other elements. That means, even if your code is excellent, you could still get only 12% maximum for the whole assignment because other elements are missing. At BSc level, coding alone is not enough. You must demonstrate other required intellectual skills such as analytical skills, evaluation, etc.

Code
Some students worry that if their code doesn't work 100%, they will get 0 marks for this part. That is not true. I will carefully look at the amount of work you put into this and see what you have written. So explain clearly what you have done and what the problems are. The marks will then be awarded accordingly. It doesn't mean you will get a complete zero if your code doesn't work. (Please also see and follow the Marking Rubric in the brief.)

You may notice that the results from your code are used in the subsequent tasks, e.g. the evaluation part. In the worst-case scenario, if your code doesn't produce any results, you should still be able to complete those tasks by 'assuming' if you had the results, what you would do in detail.

Report/Video

-The report is not long. The video is just to show, step-by-step, that your code works/partially works (or, in the worst-case scenario, does not work). So it is not at all a gigantic task.

Please remember, I cannot give you marks for nothing. So please provide me with reasonable details in your report and code as much as you can. This will give me an opportunity to give you some marks and, at the same time, maintain the academic standards.

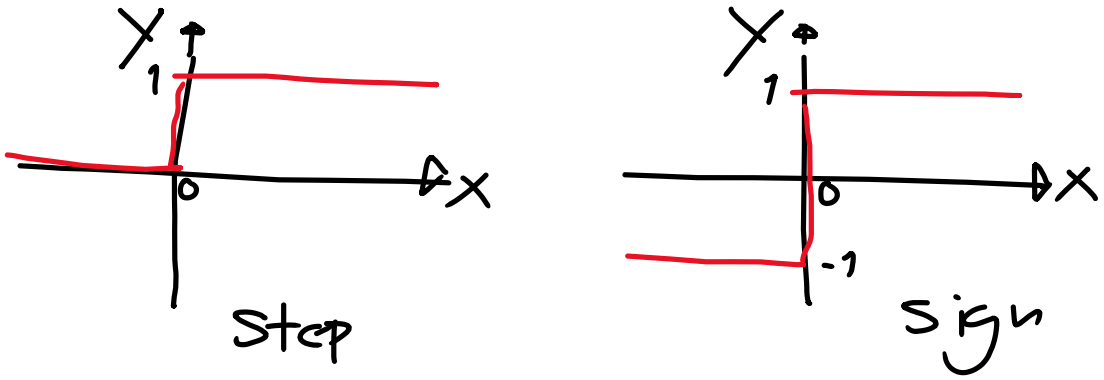
Specific Advice and hints

-The whole **Assignment A1** is based on **Lab Week 5**. It is designed to help students pass the assignment. You can base your code on the pseudocode in Lab 5. If you completed the lab during the semester, then you only need to adjust some parts of your code. If you are struggling, please look at this lab and the relevant lecture notes on back-propagation carefully. Of course, if you didn't bother to turn up to do the lab during the semester, then there is more work to be done. I can't help much.

- The Assignment is also about conducting your own investigation about what would happen if you use different activation functions. It doesn't mean that all the functions would work. You just need to report the result and describe it.

- For the alternative activation functions (Step and Sign), you don't need to do a very long calculation by yourself to find out what are the derivatives of Step and Sign functions. These are standard functions, and you could easily search textbooks or the internet to find out what their derivatives are.

- Hints on Step and Sign activation functions:



As you know, in the back-propagation algorithm, we use the derivative of the activation function. A derivative means the rate of change, in this case, of y against x. You can see from the figure that, in both the Step and Sign functions, if x is not equal to 0, then there is no change in y at all. That means the derivative of these functions is equal to zero.

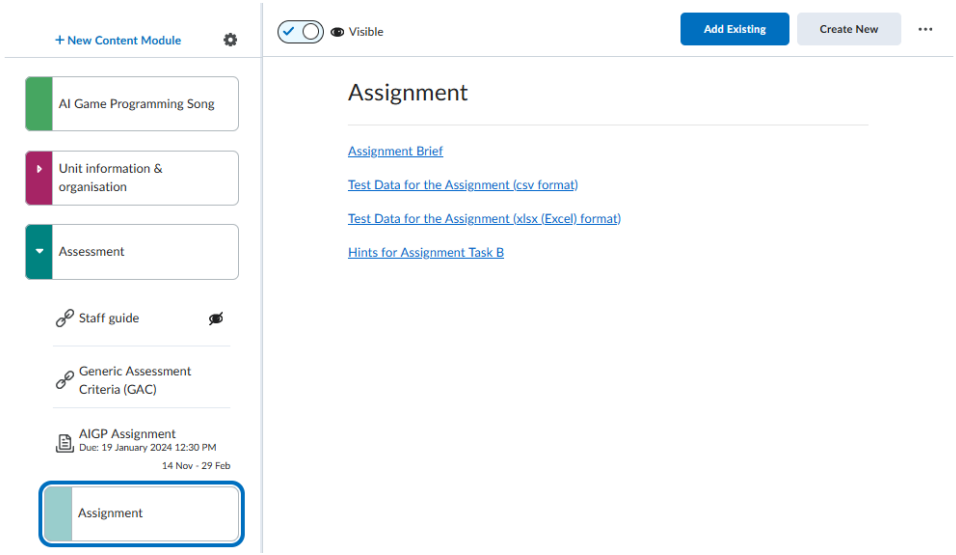
What about when x is equal to zero? You could see the sudden change in y value at this point. Therefore, in theory, approximately the derivative should be equal to infinity. But in programming, you could just use a very large number to represent the derivative at this point.

In the assignment, you need to replace the Sigmoid activation with Step or Sign functions and its derivative in Lab 5 code. If you can't remember how to do this, please see the examples in Lab 8; all explanations are in there. I have included Lab 8 here for your convenience.

- See more details in the Appendix 1 at the end of this file.

- **Assignment A2** is based on **Lab Week 6**. You can use the code you did in the lab and modify it. Again, if you didn't bother to turn up to do the lab, then there is more work for you.

- Please note that you need to use the data file provided:



If you don't know how to read the data file, please see an example in Lab Week 10.

Appendix 1

Lab discussion based on the code in Lab Week 5

Don't forget.
You should be able to implement algorithms/methods -> code
Also, the other way around, understand code and can link it to the algorithms/methods.

Effects of activation functions

- In the original code the activation function is Sigmoid.
- In Backpropagation algorithm:
  - Error gradient = derivative of the activation function multiplied by the error (cost) at that neuron's output.
- Sigmoid activation function:
  - $y = \frac{1}{1+e^{-(X-\theta)}}$

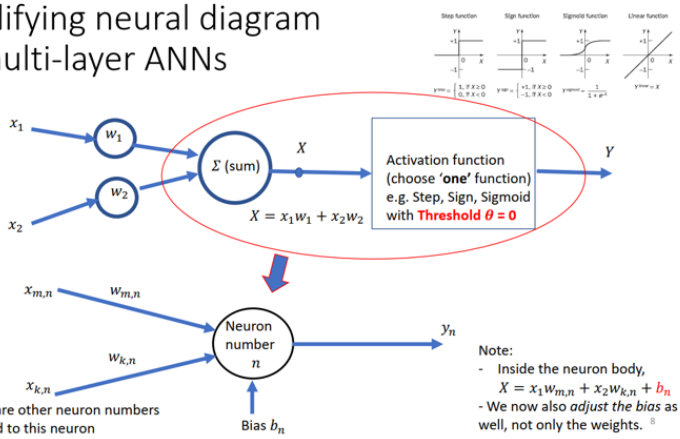
Sigmoid activation function

- $y = \frac{1}{1+e^{-(X-\theta)}} = \frac{1}{1+e^{-X}}$

$\theta = 0$ (threshold) in this case

because we have the bias already.

Simplifying neural diagram for multi-layer ANNs



Examples: neurons 4 and 5 in the pseudocode

```
% Calculate output for neuron 4
X(4) = x1(i)*w(1,4)+x2(i)*w(2,4)+Bias(4);
% Activation Function - Sigmoid function
y(4) = 1/( 1 + (exp(1))^(-(X(4))));

% Calculate output for neuron 5
X(5) = y(3)*w(3,5)+y(4)*w(4,5)+Bias(5);
% Activation Function - Sigmoid function
y(5) = 1/( 1 + (exp(1))^(-(X(5))));
```

Error gradient – Sigmoid as activation fuction

- Error gradient = derivative of the activation function multiplied by the error (cost) at that neuron's output.

$\delta = \frac{\partial y}{\partial X} \times error$

Sigmoid function $y = \frac{1}{1+e^{-X}}$

$\delta = \frac{\partial\left\{\frac{1}{1+e^{-X}}\right\}}{\partial X} \times error$

$\delta = y \times (1-y) \times error$

Example in the pseudocode

```
Delta(5) = y(5)*(1-y(5))*e(5);
Wcurrent(3,5) = w(3,5); % safe the current value before updating it
Wcurrent(4,5) = w(4,5); % safe the current value before updating it
w(3,5) = w(3,5) + Alpha*y(3)*Delta(5);
w(4,5) = w(4,5) + Alpha*y(4)*Delta(5);
Bias(5) = Bias(5) + Alpha*(1)*Delta(5);
```

```
% Neuron 3
Delta(3) = y(3)*(1-y(3))*Delta(5)*Wcurrent(3,5);
w(1,3) = w(1,3) + Alpha*x1(i)*Delta(3);
w(2,3) = w(2,3) + Alpha*x2(i)*Delta(3);
Bias(3) = Bias(3) + Alpha*(1)*Delta(3);

% Neuron 4
Delta(4) = y(4)*(1-y(4))*Delta(5)*Wcurrent(4,5);
w(1,4) = w(1,4) + Alpha*x1(i)*Delta(4);
w(2,4) = w(2,4) + Alpha*x2(i)*Delta(4);
Bias(4) = Bias(4) + Alpha*(1)*Delta(4);
```

Hence from our example (now activation function is Sigmoid):

new weight = old weight + (learning rate x input x error gradient)

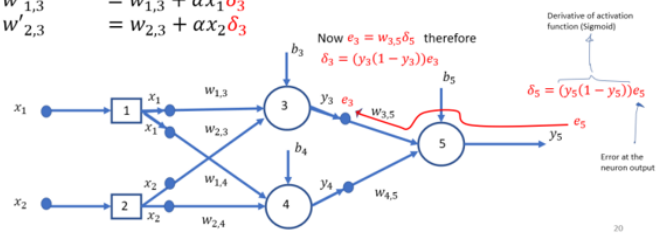
$w'_{1,3} = w_{1,3} + \alpha x_1 e_3$

$w'_{2,3} = w_{2,3} + \alpha x_2 e_3$

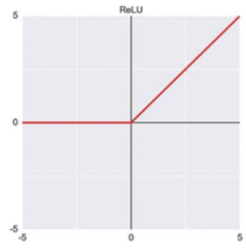
$w'_{1,3} = w_{1,3} + \alpha x_1 \delta_3$

$w'_{2,3} = w_{2,3} + \alpha x_2 \delta_3$

***Error gradient** = **derivative** of the activation function multiplied by the **error** at the neuron output*



ReLU activation function



$y = \begin{cases} x & if\ x > 0 \\ 0 & otherwise \end{cases}$

or $y = \max(0, x)$

What abou the derivative of this ReLU activation function?

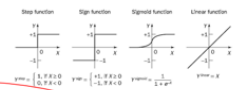
The derivative is: $y' = \begin{cases} 0 & if\ x < 0 \\ 1 & if\ x > 0 \end{cases}$

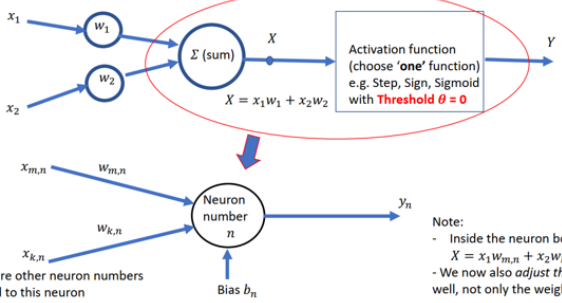
And undefined when $x = 0$, because the discontinuity; the left and right derivative are not equal.

In practice, when $x = 0$, we assume that the derivative is 0 also.

Modify the code in Lab 5 and replace the Sigmoid function with ReLU function (of course, and their derivatives). Investigate how the training process behaves.

Simplifying neural diagram for multi-layer ANNs





Examples: neurons 4 and 5 in the pseudocode.
What do you need to change?

```
% Calculate output for neuron 4
X(4) = x1(i)*w(1,4)+x2(i)*w(2,4)+Bias(4);
% Activation Function - Sigmoid function
y(4) = 1/( 1 + (exp(1))^(-(X(4))));

% Calculate output for neuron 5
X(5) = y(3)*w(3,5)+y(4)*w(4,5)+Bias(5);
% Activation Function - Sigmoid function
y(5) = 1/( 1 + (exp(1))^(-(X(5))));
```

Error gradient – ReLU as activation fuction

- Error gradient = derivative of the activation function multiplied by the error (cost) at that neuron's output.

$$\delta = \frac{\partial y}{\partial X} \times error$$

$$y = \begin{cases} x & if\ x > 0 \\ 0 & otherwise \end{cases}$$

$$\frac{\partial y}{\partial X} = \begin{cases} 0 & if\ x \leq 0 \\ 1 & if\ x > 0 \end{cases}$$

Example in the pseudocode, what to change?

```
Delta(5) = y(5)*(1-y(5))*e(5);
Wcurrent(3,5) = w(3,5); % safe the current value before updating it
Wcurrent(4,5) = w(4,5); % safe the current value before updating it
w(3,5) = w(3,5) + Alpha*y(3)*Delta(5);
w(4,5) = w(4,5) + Alpha*y(4)*Delta(5);
Bias(5) = Bias(5) + Alpha*(1)*Delta(5);
```

```
% Neuron 3
Delta(3) = y(3)*(1-y(3))*Delta(5)*Wcurrent(3,5);
w(1,3) = w(1,3) + Alpha*x1(i)*Delta(3);
w(2,3) = w(2,3) + Alpha*x2(i)*Delta(3);
Bias(3) = Bias(3) + Alpha*(1)*Delta(3);

% Neuron 4
Delta(4) = y(4)*(1-y(4))*Delta(5)*Wcurrent(4,5);
w(1,4) = w(1,4) + Alpha*x1(i)*Delta(4);
w(2,4) = w(2,4) + Alpha*x2(i)*Delta(4);
Bias(4) = Bias(4) + Alpha*(1)*Delta(4);
```

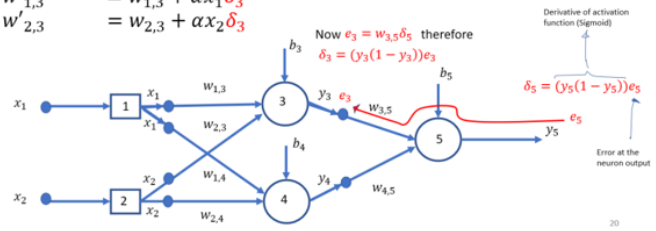
Hence from our example (now activation function is Sigmoid):

new weight = old weight + (learning rate x input x error gradient)

$$w'_{1,3} = w_{1,3} + \alpha x_1 e_3$$
$$w'_{2,3} = w_{2,3} + \alpha x_2 e_3$$

$$w'_{1,3} = w_{1,3} + \alpha x_1 \delta_3$$
$$w'_{2,3} = w_{2,3} + \alpha x_2 \delta_3$$

***Error gradient** = **derivative** of the activation function multiplied by the **error** at the neuron output*



For Assignment 1, you need to figure out the derivatives of Step and Sign functions, then modify the code accordingly.

Continue to work on your assignment.

End