COMP4660/8420 Lab 3

Neural Networks Adv.

Q1. **Use pseudocode to describe the backpropagation learning algorithm and how it works**

1. Apply inputs to network and work out the output (randomise initial weights)
2. Calculate errors of output neurons
3. Change output layer weights to adjust for errors
4. Calculate (back-propagate) errors from output layer to the hidden layer neurons. Take errors from output neurons and run them back through the weights to get the hidden layer errors.
5. Change hidden layer weights
6. Repeat

Q2. **For the neural network shown in Figure 1, with a squared error loss function, perform one pass of backpropagation and calculate the new weights. Assume the target is 1 and the learning rate is 1.**

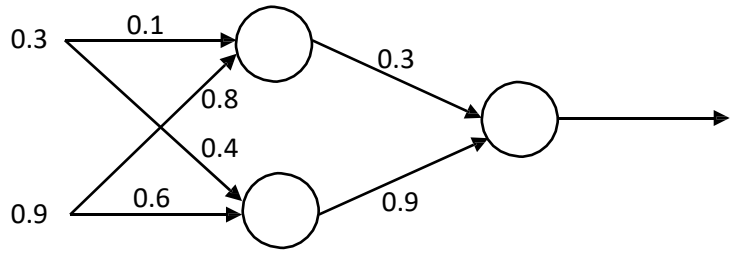
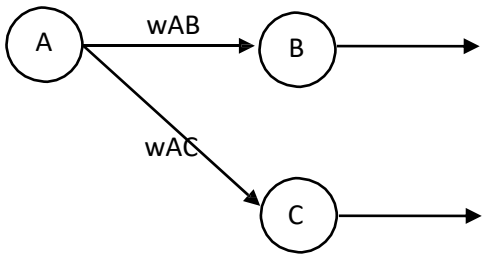


Figure 1. **A multilayer feedforward neural network**

Hint:



Loss function:

$$\frac{1}{2}\sum_i (Target_i - Output_i)^2$$

Output error:

$$\frac{dL}{dOutput_B} = -(Target_B - Output_B)$$
$$\frac{dL}{dZ_B} = Output_B * (1 - Output_B) * (Output_B - Target_B)$$

Calculate new weights:

$$W_{AB} = W_{AB} - \eta * Output_A * \frac{dL}{dZ_B}$$

Hidden layer derivative:

$$\frac{dL}{dZ_A} = Output_A * (1 - Output_A) * \left(\frac{dL}{dZ_B} * W_{AB} + \frac{dL}{dZ_C} * W_{AC}\right)$$

1) **Perform Forward Pass**

See lab 1.

2) Calculate output derivatives

$$\frac{dL}{dZ_{o1}} = output * (1 - output) * (output - target) = 0.69 * (1 - 0.69) * (0.69 - 1)$$
$$= -0.066.$$

3) Change hidden to output layer weights

$$w'_{H1O1} = w_{H1O1} - \left(\eta * \frac{dL}{dZ_{O1}} * output_{H1}\right) = 0.3 - (1 * -0.066x0.68) = 0.34$$

$$w'_{H2O1} = w_{H2O1} - \left(\eta * \frac{dL}{dZ_{O1}} * output_{H2}\right) = 0.9 + (1 * -0.066x0.66) = 0.94$$

4) Calculate hidden layer derivatives

$$\frac{dL}{dZ_{H1}} = output_{H1} * (1 - output_{H1}) * \frac{dL}{dZ_{O1}} * w_{H1O1} = 0.68 * (1 - 0.68) * -0.066 * 0.3$$
$$= -0.0043$$

$$\frac{dL}{dZ_{H2}} = output_{H2} * (1 - output_{H2}) * \frac{dL}{dZ_{O1}} * w_{H2O1} = 0.66 * (1 - 0.66) * -0.066 * 0.9$$
$$= -0.013$$

5) Change input to hidden layer weights

$$w'_{I1H1} = w_{I1H1} - \left(\eta * \frac{dL}{dZ_{H1}} * output_{I1}\right) = 0.1 - (1 * -0.0043 * 0.3) = 0.1013$$

$$w'_{I2H1} = w_{I2H1} - \left(\eta * \frac{dL}{dZ_{H1}} * output_{I2}\right) = 0.8 - (1 * -0.0043 * 0.9) = 0.8039$$

$$w'_{I1H2} = w_{I1H2} - \left(\eta * \frac{dL}{dZ_{H2}} * output_{I1}\right) = 0.4 - (1 * -0.013 * 0.3) = 0.4039$$

$$w'_{I2H2} = w_{I2H2} - \left(\eta * \frac{dL}{dZ_{H2}} * output_{I2}\right) = 0.6 - (1 * -0.013 * 0.9) = 0.6117$$

6) Calculate the new output

Hidden neuron 1:

I: (0.3 x 0.1013) + (0.9 x 0.8039) = 0.7539 O: $\frac{1}{1+e^{-0.7539}} = 0.68003$

Hidden neuron 2:

I: (0.3 x 0.4039) + (0.9 x 0.6117) = 0.6717 O: $\frac{1}{1+e^{-0.6717}} = 0.6619$

Output neuron 1:

I: (0.68 x 0.34) + (0.6621 x 0.94) = 0.85 O: $\frac{1}{1+e^{-0.85}} = 0.70$

The old error was 0.048 and the new error is 0.045 so the loss has been reduced.

Programming Task - Using PyTorch for Regression

In this lab, you will build a model to perform regression using PyTorch. The script given is an example of building a regression model using PyTorch. It aims to build a regression model for y = 3x + 3.

Task

Your task now is to construct a regression model for the Wine Quality data set: https://archive.ics.uci.edu/ml/datasets/Wine+Quality

Familiarise yourself with the data set and perform any pre-processing or normalisation needed. Use PyTorch to implement a regression model for the wine quality dataset and find the average error of predictions. Work on making your predictor as accurate as possible.

Q1. How does regression differ from classifications?

You're predicting a real value rather than a class.

Q2. What does your output look like?

A float

Q3. Where should you define your regression model?

The same place you would usually define your neural network. Everything is the same except you have no activation function on the output layer.

Q4. In the previous tasks we were calculating misclassification error. What error value might you use for a regression task?

Mean squared error loss.