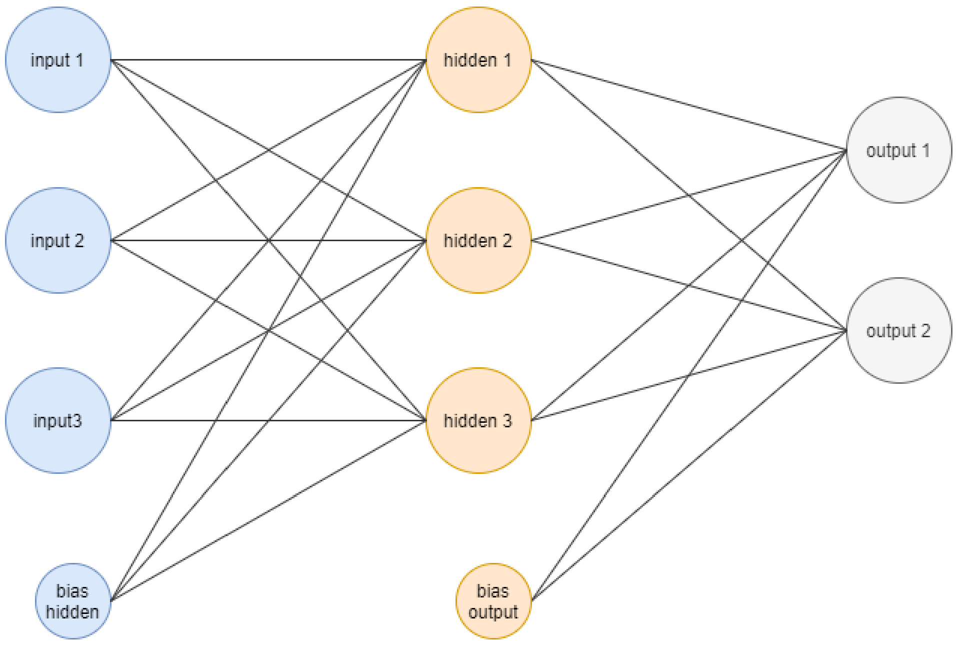
AIM 5007 Neural Networks and Deep Learning
Problem Set 2

Introduction

In this problem set, we will develop an understanding of the basic calculations that occur in a neural network. We will also explore just a bit of the mathematics that we use. Before we get into the details, I will make you this promise: I will not make you do an entire neural network pass by hand (even just one full pass can be tedious) again after this! You may choose to do one for practice, but that's up to you. In a future assignment, we will develop code to program this process. Later still, we will use built-in packages to do the work for us.

The first two problems of the assignment will ask you to perform some relatively simple mathematical manipulations. The third problem (the bulk of your work here), in multiple parts, will have you complete one full pass through the neural network. For these problems, we will use the following simple network:



As you can see, there are three input nodes plus a bias for the hidden layer, three hidden nodes plus a bias for the output layer, and two output nodes. The initial weights that we will have for the network are given in the tables below.

For the first and third problems, we will use the sigmoid activation function:

Bramer's Notation

$$\boxed{sigmoid(X) = \frac{1}{1+e^{-x}}} = \frac{e^x}{e^x+1} = \sigma(x)$$

Note that other textbooks often denote the sigmoid function by $\sigma(x)$ and that there are other mathematically equivalent ways to express the quantity. We will learn more modern activation functions later in the course. For this assignment, we will use the notation used by Max Bramer in the chapter 23 reading posted in Canvas.

Here are the initial weights:

WEIGHTS		
Link from node	**Link to node**	**Weight**
Input 1	Hidden 1	0.1
Input 1	Hidden 2	0.3
Input 1	Hidden 3	−0.2
Input 2	Hidden 1	−0.4
Input 2	Hidden 2	0.1
Input 2	Hidden 3	0.2
Input 3	Hidden 1	−0.1
Input 3	Hidden 2	−0.2
Input 3	Hidden 3	0.4
Hidden 1	Out 1	0.5
Hidden 1	Out 2	0.2
Hidden 2	Out 1	−0.3
Hidden 2	Out 2	−0.3
Hidden 3	Out 1	0.2
Hidden 3	Out 2	0.1

BIAS WEIGHTS		
	Layer	**Weight**
	Hidden	0.2
	Output	0.3

With all the above in place, complete the following problems. Your submission can be typed or (neatly) handwritten and should be ***<u>submitted as a PDF file in Canvas</u>***.

Problem 1

Show that the derivative of the sigmoid function given above is given by:

$$\frac{\partial\sigma(x)}{\partial x} = \sigma(x)\big(1 - \sigma(x)\big)$$

Problem 2

Suppose we use the identity function as our activation function. (In Bramer's notation, we would say that $Thidj = Whidj$ and $Toutk = Woutk$.) Show that the values for the outputs ($Tout1$ and $Tout2$) can be expressed as a simple linear combination of the input values and biases.

Note: This is what we mean when we say that we need a nonlinear activation function to model a nonlinear function. If we just use the identity, there is no real point to including hidden layers.

Problem 3

This problem will be the bulk of the calculation work for this assignment. We will complete a single pass (forward and backward) through our neural network. To help keep it manageable, we will break it into four steps (plus a quick repeat of the first two parts). The notation used here is consistent with the notation used by Bramer.

Part A: Forward Propagation

Given the neural network architecture and initial weights above, and using the sigmoid activation function for both the hidden and the output layers, calculate the output values given for the following training instance:

Input Values			Target Values	
Input 1	Input 2	Input 3	Output 1	Output 2
4	8	6	0.65	0.4

Your answer to this part should be the complete version of this table (note that it is the transformed value that will go into this table for the hidden and output nodes):

Input 1	Input 2	Input 3	Hidden 1	Hidden 2	Hidden 3	Output 1	Output 2
4	8	6					

Part B: Determine the Error

Using the calculated output values from part A above and the true target values given for the training instance in the table above, calculate the sum of squared errors. Note: For consistency with the Bramer chapter, use the following formula for the error:

$$E = 0.5 * \sum_{k} (targk - Toutk)^2$$

Your answer to this part will be a single number: $E = \cdots$.

Note two things. First, the 0.5 in front is purely a convenience to make gradient calculations easier later. Essentially, we are using a scale transformation of the sum of squared errors, preserving all the relevant information.

Second, this error equation should look familiar to you from linear regression. This is a commonly used error function in practical applications involving regression problems. For classification problems, different error formulas are often more appropriate. We'll see these details in coming weeks.

Part C: Calculate the Gradients

Calculating the gradients is where the bulk of the pencil and paper work is for this task. We need to calculate the gradient for each of the weights in our network (there are 17 including the two biases). We will further break this part down into four derivations that verify what Bramer gives, followed by the actual calculations for our data.

Note that I am expecting you to use the formulas and perform the calculus to verify what Bramer has given.

1. Show that for the weights from hidden to output nodes, the gradient is given by:

$$g(E, W_{jk}) = (Toutk - targk) \cdot Toutk \cdot (1 - Toutk) \cdot Thidj.$$

2. Show that for the bias term for the output nodes, the gradient is given by:

$$g(E, biasO) = \sum_k [(Toutk - targk) \cdot Toutk \cdot (1 - Toutk)].$$

3. Show that for the weights from input to hidden nodes, the gradient is given by:

$$g(E, w_{ij}) = \sum_k \left[(Toutk - targk) \cdot Toutk \cdot (1 - Toutk) \cdot W_{jk}\right] \cdot Thidj \cdot (1 - Thidj) \cdot inpi$$

4. Show that for the bias term for the hidden nodes, the gradient is given by:

$$g(E, biasH) = \sum_k \left[(Toutk - targk) \cdot Toutk \cdot (1 - Toutk) \cdot W_{jk}\right] \cdot Thidj \cdot (1 - Thidj)$$

Note that in the case of (1) we are deriving Bramer's equation I, in the case of (2) we are deriving Bramer's equation J, in the case of (3) we are deriving a combination of Bramer's equations K and L, and in the case of (4) we are deriving a combination of Bramer's equations K and M.

Once you have derived the gradient formulas above, use them to calculate the actual gradient values for our first pass through our network. Your answer to this part should be the completed version of the following table (like the one on Bramer p. 452):

Link from node	Link to node	Gradient
Input 1	Hidden 1	
Input 1	Hidden 2	
Input 1	Hidden 3	
Input 2	Hidden 1	
Input 2	Hidden 2	
Input 2	Hidden 3	
Input 3	Hidden 1	
Input 3	Hidden 2	
Input 3	Hidden 3	
Hidden 1	Out 1	
Hidden 1	Out 2	
Hidden 2	Out 1	
Hidden 2	Out 2	
Hidden 3	Out 1	
Hidden 3	Out 2	

	Layer	(Bias)
	Hidden	
	Output	

The good news is that part C is, by far, the bulk of the work. We are almost done! When we imagine using hundreds of thousands of epochs and large data sets for much more complex network architectures, it quickly becomes clear why deep learning methods were not feasible until computing power recently caught up.

Part D: Update the Weights

Using the gradient values calculated above and ***a learning rate of $\alpha = \mathbf{0.2}$***, update the weights. Recall that the formula for updating a weight is:

$$NewWeight = OldWeight - \alpha \cdot Gradient$$

So, for instance, if the old weight is 0.4, the gradient is 0.25, and the learning rate is 0.2, the new weight would be $0.4 - (0.2) \cdot (0.25) = 0.35$.

Your answer to this part should be the completed version of the following table:

NEW WEIGHTS		
Link from node	**Link to node**	**Weight**
Input 1	Hidden 1	
Input 1	Hidden 2	
Input 1	Hidden 3	
Input 2	Hidden 1	
Input 2	Hidden 2	
Input 2	Hidden 3	
Input 3	Hidden 1	
Input 3	Hidden 2	
Input 3	Hidden 3	
Hidden 1	Out 1	
Hidden 1	Out 2	
Hidden 2	Out 1	
Hidden 2	Out 2	
Hidden 3	Out 1	
Hidden 3	Out 2	

NEW BIAS WEIGHTS		
	Layer	**Weight**
	Hidden	
	Output	

Once we have filled in the table above, we have completed one pass through our neural network. Believe it or not, that's all there is to neural networks and deep learning from a conceptual standpoint. (Of course, there's a lot more detail about choosing the correct architectures, learning rates, and so on, not to mention evaluating the output and the usefulness of the model. Otherwise, this would be a very short course!)

Let's do just a bit more…

Part A Redux: Forward Propagation

As we did in part A above but using the new weights that we have just calculated in part D, determine the output values for a new forward pass through the neural network. You should again complete the following table:

Input 1	Input 2	Input 3	Hidden 1	Hidden 2	Hidden 3	Output 1	Output 2
4	8	6					

Part B Redux: Determine the Error

As we did in part B above, calculate the sum of squared errors (including multiplying by 0.5 again as well). How does the new error compare to the previous error? Did it decrease or not?