Tutorial Sections 13-14

1. A home automation company is developing an ML model to predict the energy efficiency of houses. The prediction is based on two main factors: insulation quality and window size. The company aims to provide homeowners with an energy efficiency score, which helps them understand how improvements in these areas could lead to better energy efficiency.

   (a) Sketch the architecture of a neural network considering the two inputs and a bias term, one output, and one hidden layer with 3 neurons and a bias. What are the meanings of the bias terms in the network?

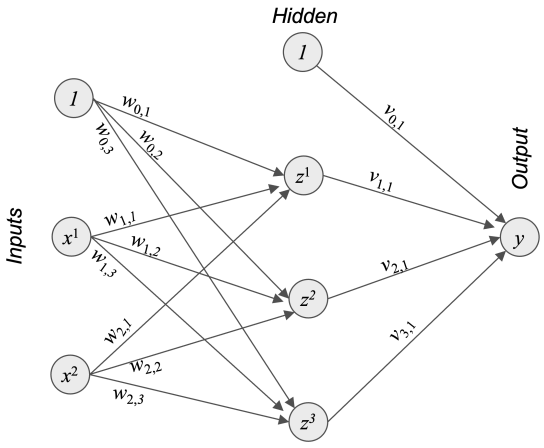
   (b) Consider the initial weights for the hidden layer, W, and for the perceptron at the output layer, v, are given by:

$$W = \begin{bmatrix} 0.1 & 0.2 & 0.1 \\ 0.1 & 0.2 & 0.1 \\ 0.2 & 0.1 & 0.2 \end{bmatrix} \text{ and } v^T = \begin{bmatrix} 0.2 & 0.1 & 0.2 & 0.1 \end{bmatrix}. \tag{1}$$

   Using this information, implement forward propagation to compute the network output for a specific input of 0.7 and 0.5 for insulation quality and window size, respectively. Assume the activation function for the hidden layer is a ReLU, and the output layer activation function is a softplus.

Solution: Although this is not a classification problem, it provides an operational understanding of how the networks perform calculations with the opportunity to practice different operational aspects discussed in lectures 13 and 14.

(a) The sketch of the NN is shown in the figure below, along with the weights from one layer to the other. In this NN, we included two bias terms (1 in the figure). The bias term gives each neuron in the hidden and output layers an additional parameter to adjust during the SGD algorithm. It allows the neurons to shift their activation function to the left or right to fit the input data better.



$$W^T = \begin{bmatrix} w_{0,1} & w_{1,1} & w_{2,1} \\ w_{0,2} & w_{1,2} & w_{2,2} \\ w_{0,3} & w_{1,3} & w_{2,3} \end{bmatrix} \qquad v^T = \begin{bmatrix} v_{0,1} & v_{1,1} & v_{2,1} & v_{3,1} \end{bmatrix}$$

$$z = \max(0, W^T x) \qquad y = \max(0, v^T z)$$

(b) Let's assume

$$x = \begin{bmatrix} x^0 \\ x^1 \\ x^2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.7 \\ 0.5 \end{bmatrix}.$$

Then, the hidden input is given by

$$W^T x = \begin{bmatrix} 0.1 & 0.1 & 0.2 \\ 0.2 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.2 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0.7 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.1 \times 1 + 0.1 \times 0.7 + 0.2 \times 0.5 \\ 0.2 \times 1 + 0.2 \times 0.7 + 0.1 \times 0.5 \\ 0.1 \times 1 + 0.1 \times 0.7 + 0.2 \times 0.5 \end{bmatrix} = \begin{bmatrix} 0.27 \\ 0.39 \\ 0.27 \end{bmatrix},$$

and applying the ReLU activation function for the hidden layer gives

$$z = \max\left(0, W^T x\right) = \begin{bmatrix} 0.27 \\ 0.39 \\ 0.27 \end{bmatrix}$$

The input to the output layer will be given by:

$$v^T z = \begin{bmatrix} 0.2 & 0.1 & 0.2 & 0.1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0.27 \\ 0.39 \\ 0.27 \end{bmatrix} = 0.2 \times 1 + 0.1 \times 0.27 + 0.2 \times 0.39 + 0.1 \times 0.27 = 0.332.$$

The Softplus activation function results in

$$y = \ln\left(1 + e^{v^T z}\right) = \ln\left(1 + e^{0.332}\right) = 0.873$$

Therefore, the predicted energy efficiency score is approximately 0.873.

2. Imagine a smart home security system designed to make decisions based on multiple sensors. Two of these sensors might be a motion detector and a window sensor. The security system is programmed to trigger an alarm if either the motion detector is activated (someone is moving inside the house) or the window sensor is tripped (a window is opened), but not both. This is where the XOR logic gate comes into play: it outputs T only when the inputs differ (one True, one False). Design and manually calculate the output of a simple multilayer neural network that mimics the behaviour of an XOR logic gate.

Solution: This problem builds upon the example of Section 13.3 of the lecture notes. It gives another opportunity to practice using deep networks to compute logical functions within a contextualised problem.
The 'xor' function can be written as $\text{XOR}(A, B) = (A \wedge \neg B) \vee (\neg A \wedge B)$, which in terms of the neural network functions is:

$$y(u, v) = f_{\text{or}}\left(f_{\text{and}}(u, f_{\text{not}}(v)), f_{\text{and}}(f_{\text{not}}(u), v)\right).$$

We can use the $\text{sign}(x)$ function as the activation function to operate in an input layer with two inputs x^1 and x^2; each can be either -1 (*False*) or 1 (*True*).
We can implement the 'not' function as:

$$f_{\text{not}}(x) = \text{sign}\left(-x\right),$$

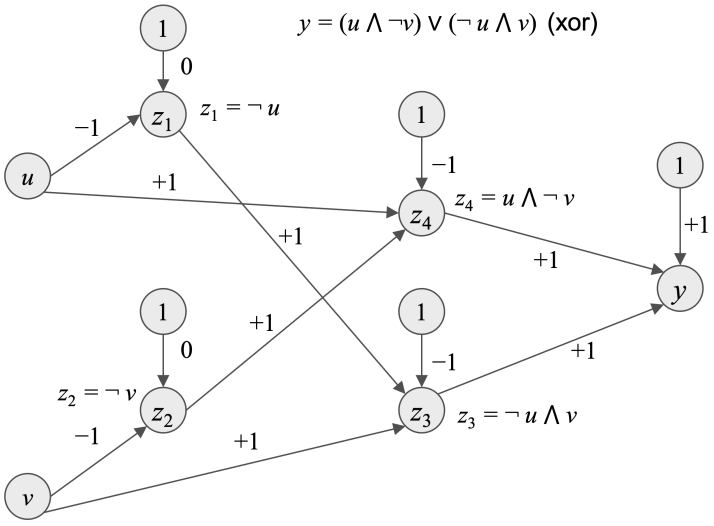
and the 'and' operation as:

$$f_{\text{and}}(x^1, x^2) = \text{sign}\left(-1 + x^1 + x^2\right).$$

Similarly, the 'or' operation can be represented as:

$$f_{\text{or}}(x^1, x^2) = \text{sign}\left(1 + x^1 + x^2\right).$$

The representations of each of these functions using single-layer linear neurons with sign activation functions are shown in Figure 13.4 of the notes. Putting everything together results in the multilayer neural network in Figure below:



3. A telecommunications company is looking to predict customer churn based on two key factors: average monthly usage (in hours) and average monthly bill (in pounds). The goal is to identify customers likely to leave the service so proactive steps can be taken to retain them.

   (a) Consider a two-layer neural network that predicts the likelihood of customer churn (churn = 1, no churn = −1) based on the two mentioned factors. Write the forward model for this neural network.

   (b) Assume that the weights are given by:

$$W = \begin{bmatrix} 0.15 & 0.25 \\ 0.20 & 0.30 \end{bmatrix}, \text{ and } v = \begin{bmatrix} 0.40 \\ 0.45 \end{bmatrix}.$$

   Calculate one iteration of weight updates for a two-layer neural network using automatic differentiation (AD) considering a customer that averaged a monthly usage of 10 hours, paid an average monthly bill of £60, and did not churn. Use dual numbers to calculate the output gradient with respect to each weight in the network. Assume a ReLU activation function for all layers, and the perceptron loss as the error function.

   (c) Explain how these gradients can be used to adjust the weights in a gradient descent step. Perform one iteration of weight updates using a simple learning rate $\alpha = 0.01$.

Solution:

(a) Let's define x^1 as the average monthly usage (hours) and x^2 as the average monthly bill (£) as

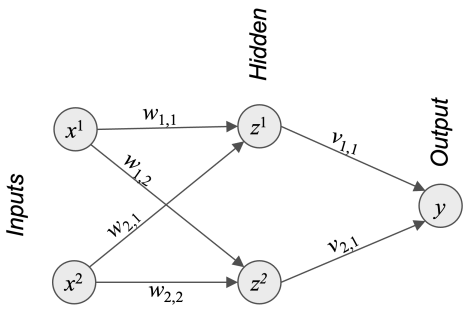
our input layer. The hidden layer consists of two neurons with ReLU activation:

$$z = \max\left(0, W^T x\right) \longrightarrow z^1 = \max\left(0, w_{1,1}x^1 + w_{2,1}x^2\right)$$
$$z^2 = \max\left(0, w_{1,2}x^1 + w_{2,2}x^2\right).$$

The output layer (churn prediction) consists of one neuron:

$$y = \max\left(0, v_1 z^1 + v_2 z^2\right)$$

We can visually see the neural network in the Figure below:



$$W^T = \begin{bmatrix} w_{1,1} & w_{2,1} \\ w_{1,2} & w_{2,2} \end{bmatrix} \qquad v^T = \begin{bmatrix} v_{1,1} & v_{2,1} \end{bmatrix}$$

$$z = \max(0,\, W^T x) \qquad y = \max(0,\, v^T z)$$

(b) The input is given by $\bar{x}^1 = (10, 0)$, $\bar{x}^2 = (60, 0)$. Since the customer did not churn, $\bar{y} = (-1, 0)$. The linear weights can be written as dual numbers. With respect to $w_{1,1}$, we would have:

$$\bar{W} = \begin{bmatrix} (0.15, 1) & (0.25, 0) \\ (0.20, 0) & (0.30, 0) \end{bmatrix}, \bar{v} = \begin{bmatrix} (0.40, 0) \\ (0.45, 0) \end{bmatrix}$$

The inputs of the hidden layer would be:

$$\bar{w}_{1,1}\bar{x}^1 + \bar{w}_{2,1}\bar{x}^2 = (0.15, 1) \cdot (10, 0) + (0.20, 0) \cdot (60, 0) = (13.5, 10)$$
$$\bar{w}_{1,2}\bar{x}^1 + \bar{w}_{2,2}\bar{x}^2 = (0.25, 0) \cdot (10, 0) + (0.30, 0) \cdot (60, 0) = (20.5, 0)$$

The outputs of the hidden layer are:

$$\bar{z}^1 = \max\left((0, 0), (13.5, 10)\right) = (13.5, 10)$$
$$\bar{z}^2 = \max\left((0, 0), (20.5, 0)\right) = (20.5, 0),$$

so that the input of the output node is:

$$\bar{v}_1\bar{z}^1 + \bar{v}_2\bar{z}^2 = (0.4, 0) \cdot (13.5, 10) + (0.45, 0) \cdot (20.5, 0) = (14.6, 4),$$

Using the ReLU activation function, the predicted output is given by:

$$\hat{y} = \max\left(0, v^T z\right) = \max\left((0, 0), (14.6, 4)\right) = (14.6, 4)$$

so that the perceptron loss is given by:

$$F(w_{1,1}) = \max\left(0, -y\hat{y}\right) = \max\left((0, 0), (-1, 0) \times (-1, 0) \times (14.6, 4)\right) = (14.6, 4)$$

With respect to $w_{1,2}$, we have:

$$\bar{W} = \begin{bmatrix} (0.15, 0) & (0.25, 1) \\ (0.20, 0) & (0.30, 0) \end{bmatrix}, \bar{v} = \begin{bmatrix} (0.40, 0) \\ (0.45, 0) \end{bmatrix}$$

The inputs of the hidden layer would be:

$$\bar{w}_{1,1}\bar{x}^1 + \bar{w}_{2,1}\bar{x}^2 = (0.15, 0) \cdot (10, 0) + (0.20, 0) \cdot (60, 0) = (13.5, 0)$$
$$\bar{w}_{1,2}\bar{x}^1 + \bar{w}_{2,2}\bar{x}^2 = (0.25, 1) \cdot (10, 0) + (0.30, 0) \cdot (60, 0) = (20.5, 10)$$

The outputs of the hidden layer are:

$$\bar{z}^1 = \max\left((0, 0), (13.5, 0)\right) = (13.5, 0)$$
$$\bar{z}^2 = \max\left((0, 0), (20.5, 10)\right) = (20.5, 10),$$

so that the input of the output node is:

$$\bar{v}_1\bar{z}^1 + \bar{v}_2\bar{z}^2 = (0.4, 0) \cdot (13.5, 0) + (0.45, 0) \cdot (20.5, 10) = (14.6, 4.5)$$

Using the ReLU activation function, we have the predicted output as:

$$\hat{y} = \max\left(0, v^T z\right) = \max\left((14.6, 4.5)\right) = (14.6, 4.5)$$

The perceptron loss is given by:

$$F(w_{1,2}) = \max\left(0, -y\hat{y}\right) = \max\left((0, 0), (-1, 0) \times (-1, 0) \times (14.6, 4.5)\right) = (14.6, 4.5)$$

Repeating the same strategy, we have that wrt $w_{2,1}$,

$$\bar{W} = \begin{bmatrix} (0.15, 0) & (0.25, 0) \\ (0.20, 1) & (0.30, 0) \end{bmatrix}, \bar{v} = \begin{bmatrix} (0.40, 0) \\ (0.45, 0) \end{bmatrix}$$

so that

$$\bar{z}^1 = (13.5, 60),\ \bar{z}^2 = (20.5, 0)$$
$$F(w_{2,1}) = (14.6, 24)$$

For the remaining weights, we have

$$F(w_{2,2}) = (14.6, 27), F(v_1) = (14.6, 13.5), F(v_2) = (14.6, 20.5).$$

(c) The gradient of the output wrt $w_{1,1}$ is 4. This means that a small change in $w_{1,1}$ will result in approximately four times that change in the network output, given the current input value and other weights. This gradient can be used to update $w_{1,1}$ using a gradient descent step:

$$w^1_{1,1} = w^0_{1,1} - \alpha F_w(w_{1,1}) = 0.15 - 0.01 \times 4 = 0.15 - 0.04 = 0.11.$$

Performing similar calculations for all other weights, you can check that:

$$w^1_{1,2} = w^0_{1,2} - \alpha F_w(w_{1,2}) = 0.25 - 0.01 \times 4.5 = 0.205$$
$$w^1_{2,1} = w^0_{2,1} - \alpha F_w(w_{2,1}) = 0.2 - 0.01 \times 24 = -0.04$$
$$w^1_{2,2} = w^0_{2,2} - \alpha F_w(w_{2,2}) = 0.3 - 0.01 \times 27 = 0.03$$
$$v^1_1 = v^0_1 - \alpha F_w(v_1) = 0.4 - 0.01 \times 13.5 = 0.265$$
$$v^1_2 = v^0_2 - \alpha F_w(v_2) = 0.45 - 0.01 \times 20.5 = 0.245,$$

so that we can write the new weight matrices as

$$W^1 = \begin{bmatrix} 0.11 & 0.205 \\ -0.04 & 0.03 \end{bmatrix}, \text{ and } v^1 = \begin{bmatrix} 0.265 \\ 0.245 \end{bmatrix}.$$