Lecture 2: Training a Neural Network

Definition

- Neural Networks are biologically inspired computational models that solve the regression/classification problem by activating processing units based on weights that connect these processing units throughout several layers.
- Prediction is done by computing values from the leftmost layer, feeding it to the layer to the right until it reaches the last layer.
- To optimize prediction, one has to train the network meaning it adjusts the weights that modifies values of neurons in consecutive layers until the final value is as close as possible to the actual value (sometimes called the target value).

Components

- **Neuron**: A processing unit of a neural network that contains a single value. A value of a neuron can be squished to a lower value range by using some activation function like sigmoid
- **Layer**: A layer is a container of neurons.
  - Input Layer: A container that contains neurons whose values are the feature vector of an instance of your data. For a typical neural network, it will have only 1 input layer (the leftmost layer representing a single vector of data – your input).
  - Hidden Layer: Layers in between input and output layer that contain neurons whose values depend on the values passed to it from the layer to the left of it. The neurons in the hidden layer are sometimes called latent variables. Latent variables are said to be a compressed version of the input if the number of latent variables are smaller than the number of inputs. Latent variables are said to be sparse (or a sparse representation of the input) if the number of latent variables in a hidden layer are greater than the number of neurons in the input layer
  - Output Layer: Final layer of the neural network containing the predicted values.
- **Weights**: Values that connect neurons from one layer to the next. Each neuron will have exactly n number of weights where n is the number of neurons to the layer to the right of the current layer.
- **Topology**: The schematic of a neural network. It is usually represented by a vector of integers. For example, topology [3, 2, 1] is a neural network with 3 layers. The input layer contains 3 neurons, the single hidden layer contains 2 neurons and the output layer contains 1 neuron.

The Math Needed

Sigmoid Function: Given a number x, we can fit the number into a range of 0 to 1 by using the following:

$$f(x)=\frac{x}{1+|x|}$$

Derivative of Sigmoid Function: To approximate the derivative value given by the sigmoid function f(x), we can use the following:

$$f'(x)=f(x)(1-f(x))$$

Matrix Transpose: An operation that flips a matrix over its diagonal. Usually denoted with a superscript T. For example:

$$A=\begin{bmatrix}1 & 4\\ 2 & 5\\ 3 & 6\end{bmatrix} \qquad A^T=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$$

Matrix Multiplication (uneven dimensions): To multiply two matrices together with uneven dimensions, we have to first follow simple rules:

1. We read a matrix as (**number of rows x number of columns**)
2. Number of **columns of the first matrix == number of rows in the second matrix**.
3. The resulting product is a matrix of size (**number of rows of first matrix x number of columns of second matrix**).

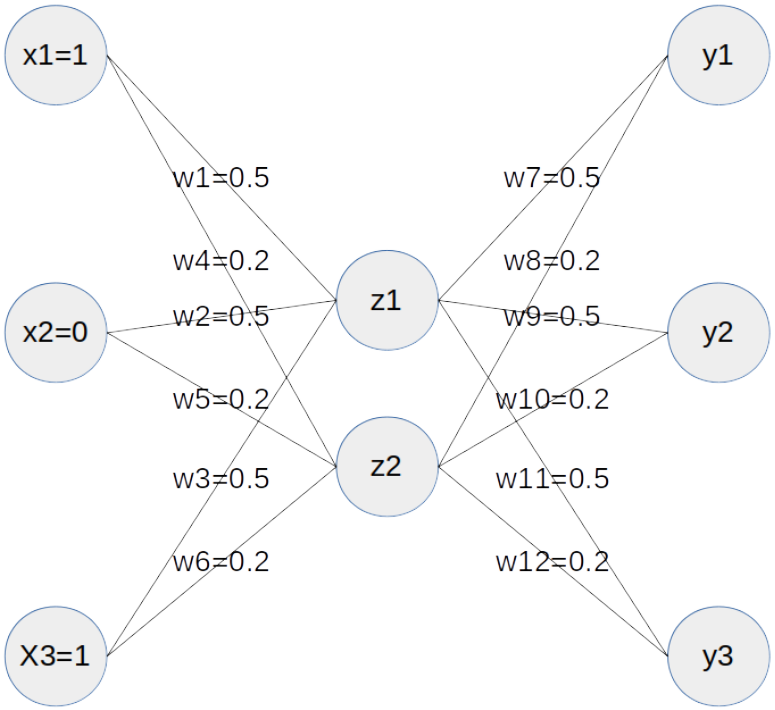
$$\begin{bmatrix}z_1 & z_2\end{bmatrix}=\begin{bmatrix}x_1 & x_2 & x_3\end{bmatrix}\cdot\begin{bmatrix}w_1 & w_4\\ w_2 & w_5\\ w_3 & w_6\end{bmatrix} \qquad z_1=x_1w_1+x_2w_2+x_3w_3 \qquad z_2=x_1w_4+x_2w_5+x_3w_6$$

Computing for MLP

In this section, we show a step by step computation on how to train a neural network by first doing a feedforward pass – meaning we predict the output using some random weights. Whatever answer we get, we check how far it is from the target value and use that information to adjust the weights in what's called a backward pass or backpropagation.

Example Problem:

- Topology: [3, 2, 3]
- Target Value: [1, 0, 1]
- Weights that connect the layers are initialized randomly



Feed Forward

This solves the problem from left to right until we get a set of y values by performing dot product with input vector and weight matrix between input and hidden.

$$\begin{bmatrix} z_1 & z_2 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \cdot \begin{bmatrix} w_1 & w_4 \\ w_2 & w_5 \\ w_3 & w_6 \end{bmatrix} \qquad z_1 = x_1 w_1 + x_2 w_2 + x_3 w_3 \qquad z_2 = x_1 w_4 + x_2 w_5 + x_3 w_6$$

$$z_1 = (1)(0.5) + (0)(0.5) + (1)(0.5) = 1$$

$$z_1 = f(1) = \frac{1}{1+|1|} = 0.5$$

$$z_2 = (1)(0.2) + (0)(0.2) + (1)(0.2) = 0.4$$

$$z_2 = f(0.4) = \frac{0.4}{1+|0.4|} = 0.2857$$

Hidden to Output

Do a dot product using the latent variables' values to excite the output layer's neurons' values according to the weights connected to it:

$$\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} z_1 & z_2 \end{bmatrix} \cdot \begin{bmatrix} w_7 & w_9 & w_{11} \\ w_8 & w_{10} & w_{12} \end{bmatrix}$$

$$y_1 = (0.5)(0.5) + (0.2857)(0.2) = 0.3071$$

$$y_1 = f(0.3071) = \frac{0.3071}{1 + |0.3071|} = 0.2350$$

$$y_2 = (0.5)(0.5) + (0.2857)(0.2) = 0.3071$$

$$y_2 = f(0.3071) = \frac{0.3071}{1 + |0.3071|} = 0.2350$$

$$y_3 = (0.5)(0.5) + (0.2857)(0.2) = 0.3071$$

$$y_3 = f(0.3071) = \frac{0.3071}{1 + |0.3071|} = 0.2350$$

Cost/Error

The cost/error is defined as the distance between the hypothesis (y values) and the target. It can be computed by the following equation:

$$\begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} - \begin{bmatrix} \widehat{y_1} & \widehat{y_2} & \widehat{y_3} \end{bmatrix}$$

$$\begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix} = \begin{bmatrix} 0.2350 & 0.2350 & 0.2350 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.765 & 0.235 & -0.765 \end{bmatrix}$$

$$E_{total} = -1.29508197$$

Back Propagation

Once we determine the error, we can use it to adjust the weights so we can minimize the error and bring the prediction as close to the target as possible. We do this by computing gradients to get the delta weight (change in weight) that we subtract from the original weights.

OUTPUT TO HIDDEN

Compute the Gradients

To compute the gradients from the output to the hidden layer, use the following equation:

$$\begin{bmatrix} g_{1_o} & g_{2_o} & g_{3_o} \end{bmatrix} = \begin{bmatrix} y'_1 & y'_2 & y'_3 \end{bmatrix} \begin{bmatrix} e_1 & \dot{e}_2 & e_3 \end{bmatrix} = \begin{bmatrix} 0.1798 & 0.1798 & 0.1798 \end{bmatrix} \begin{bmatrix} -0.765 & \dot{0.235} & -0.765 \end{bmatrix}$$

$$\begin{bmatrix} g_{1_o} & g_{2_o} & g_{3_o} \end{bmatrix} = \begin{bmatrix} -0.1375 & 0.0422 & -0.1374 \end{bmatrix}$$

Use the output gradients to compute for the delta weights for the hidden to output layer:

$$\begin{bmatrix} \widehat{w_7} & \widehat{w_8} \\ \widehat{w_9} & \widehat{w_{10}} \\ \widehat{w_{11}} & \widehat{w_{12}} \end{bmatrix}^T = \begin{bmatrix} g_{1_o} & g_{2_o} & g_{3_o} \end{bmatrix}^T \begin{bmatrix} z_1 & \dot{z}_2 \end{bmatrix} = \begin{bmatrix} g_{1_o} \\ g_{2_o} \\ g_{3_o} \end{bmatrix} \begin{bmatrix} z_1 & \dot{z}_2 \end{bmatrix}$$

$$\begin{bmatrix} \widehat{w_7} & \widehat{w_8} \\ \widehat{w_9} & \widehat{w_{10}} \\ \widehat{w_{11}} & \widehat{w_{12}} \end{bmatrix}^T = \begin{bmatrix} -0.1375 \\ 0.0422 \\ -0.1374 \end{bmatrix} \begin{bmatrix} 0.5 & \dot{0.2857} \end{bmatrix}$$

$$\begin{bmatrix} \widehat{w_7} & \widehat{w_8} \\ \widehat{w_9} & \widehat{w_{10}} \\ \widehat{w_{11}} & \widehat{w_{12}} \end{bmatrix}^T = \begin{bmatrix} -0.0688 & -0.0393 \\ 0.0211 & 0.0121 \\ -0.0688 & -0.0393 \end{bmatrix}^T$$

Update the weights for the hidden to output layer by subtracting the delta weights from it:

$$\begin{bmatrix} W_7 & W_9 & W_{11} \\ W_8 & W_{10} & W_{12} \end{bmatrix} = \begin{bmatrix} w_7 & w_9 & w_{11} \\ w_8 & w_{10} & w_{12} \end{bmatrix} - \begin{bmatrix} \widehat{w_7} & \widehat{w_9} & \widehat{w_{11}} \\ \widehat{w_8} & \widehat{w_{10}} & \widehat{w_{12}} \end{bmatrix}$$

$$\begin{bmatrix} W_7 & W_9 & W_{11} \\ W_8 & W_{10} & W_{12} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.2 & 0.2 & 0.2 \end{bmatrix} - \begin{bmatrix} -0.0688 & 0.0211 & -0.0688 \\ -0.0393 & 0.0121 & -0.0393 \end{bmatrix}$$

$$\begin{bmatrix} W_7 & W_9 & W_{11} \\ W_8 & W_{10} & W_{12} \end{bmatrix} = \begin{bmatrix} -0.56876085 & -0.47888059 & -0.56876085 \\ -0.23929192 & -0.18793177 & -0.23929192 \end{bmatrix}$$

HIDDEN TO INPUT

To compute the delta weights for the hidden to input layers, we first need to compute for the derivative of the hidden layer's neurons:

$$Z' = \begin{bmatrix} z_1' & z_2' \end{bmatrix} = \begin{bmatrix} \left(0.5\left(1-0.28571429\right)\right) & \left(0.28571429\left(1-0.28571429\right)\right) \end{bmatrix} = \begin{bmatrix} 0.25 & 0.20408163 \end{bmatrix}$$

To get the gradients for the hidden to input layer, we get the product of the gradient vector (the result of the error and y values from the previous computation) and the weights fanning off from each hidden neuron. We can express this as follows:

$$G_h = \begin{bmatrix} g_{1_h} & g_{2_h} \end{bmatrix} = \left(\sum_i G_o W_{z_i}^T \right) Z'$$

Breaking it down, we can compute it as follows:

$$g_{1_h} = \begin{bmatrix} g_{o_1} & g_{o_2} \end{bmatrix} \begin{bmatrix} w_7 & w_9 & w_{11} \end{bmatrix}^T = \begin{bmatrix} g_{o_1} & g_{o_2} & g_{o_3} \end{bmatrix} \begin{bmatrix} w_7 \\ w_9 \\ w_{11} \end{bmatrix} = \begin{bmatrix} -0.13752171 & 0.04223881 & -0.13752171 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

$$g_{1_h} = -0.1164023$$

$$g_{2_h} = \begin{bmatrix} g_{o_1} & g_{o_2} \end{bmatrix} \begin{bmatrix} w_8 & w_{10} & w_{12} \end{bmatrix}^T = \begin{bmatrix} g_{o_1} & g_{o_2} & g_{o_3} \end{bmatrix} \begin{bmatrix} w_8 \\ w_{10} \\ w_{12} \end{bmatrix} = \begin{bmatrix} -0.13752171 & 0.04223881 & -0.13752171 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \end{bmatrix}$$

$$g_{2_h} = -0.04656092$$

$$G_h = \begin{bmatrix} -0.1164023 & -0.04656092 \end{bmatrix} \begin{bmatrix} 0.25 & 0.20408163 \end{bmatrix} = \begin{bmatrix} -0.02910058 & -0.00950223 \end{bmatrix}$$

To finally get the delta weights, we multiply the transposed gradients to the input using the following:

$$\begin{bmatrix} \widehat{w_1} & \widehat{w_3} & \widehat{w_5} \\ \widehat{w_2} & \widehat{w_4} & \widehat{w_6} \end{bmatrix}^T = \begin{bmatrix} g_{h_1} & g_{h_2} \end{bmatrix}^T \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} = \begin{bmatrix} g_{h_1} \\ g_{h_2} \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

$$\begin{bmatrix} \widehat{w_1} & \widehat{w_3} & \widehat{w_5} \\ \widehat{w_2} & \widehat{w_4} & \widehat{w_6} \end{bmatrix}^T = \begin{bmatrix} -0.02910058 \\ -0.00950223 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.02910058 & 0 & -0.02910058 \\ -0.00950223 & 0 & -0.00950223 \end{bmatrix}^T$$

$$\begin{bmatrix} \widehat{w_1} & \widehat{w_3} & \widehat{w_5} \\ \widehat{w_2} & \widehat{w_4} & \widehat{w_6} \end{bmatrix}^T = \begin{bmatrix} -0.02910058 & -0.00950223 \\ 0 & 0 \\ -0.02910058 & -0.00950223 \end{bmatrix}$$

We then subtract it from the original weights to get the new set of weights:

$$\begin{bmatrix} W_1 & W_3 & W_5 \\ W_2 & W_4 & W_6 \end{bmatrix}^T = \begin{bmatrix} 0.5 & 0.2 \\ 0.5 & 0.2 \\ 0.5 & 0.2 \end{bmatrix} - \begin{bmatrix} -0.02910058 & -0.00950223 \\ 0 & 0 \\ -0.02910058 & -0.00950223 \end{bmatrix} = \begin{bmatrix} 0.52910058 & 0.20950223 \\ 0.5 & 0.2 \\ 0.52910058 & 0.20950223 \end{bmatrix}$$

Testing the New Weights

At this point, we have the updated weights for both the input to hidden and hidden to output layers. Using the same feed forward process, we can determine if our final error lowered from the initial prediction.

1. Feed Forward Input to Hidden

$$\begin{bmatrix} z_1 & z_2 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \cdot \begin{bmatrix} w_1 & w_4 \\ w_2 & w_5 \\ w_3 & w_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.52910058 & 0.20950223 \\ 0.5 & 0.2 \\ 0.52910058 & 0.20950223 \end{bmatrix} = \begin{bmatrix} 1.05820115 & 0.41900446 \end{bmatrix}$$

$$\begin{bmatrix} z_1 & z_2 \end{bmatrix} = f\left(\begin{bmatrix} z_1 & z_2 \end{bmatrix}\right) = \left[\left(\frac{1.05820115}{1+|1.05820115|}\right) \quad \left(\frac{0.41900446}{1+|0.41900446|}\right)\right] = \begin{bmatrix} 0.51413884 & 0.29528058 \end{bmatrix}$$

2. Feed Forward Hidden to Output

$$\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} W_7 & W_9 & W_{11} \\ W_8 & W_{10} & W_{12} \end{bmatrix}$$

$$\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} 0.51413884 & 0.29528058 \end{bmatrix} \begin{bmatrix} 0.56876085 & 0.47888059 & 0.56876085 \\ 0.23929192 & 0.18793177 & 0.23929192 \end{bmatrix}$$

$$\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} 0.363080301 & 0.301703715 & 0.36308031 \end{bmatrix}$$

$$\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = f\left(\begin{bmatrix} 0.363080301 & 0.301703715 & 0.36308031 \end{bmatrix}\right) = \begin{bmatrix} 0.266367507 & 0.231776026 & 0.266367507 \end{bmatrix}$$

3. Determine the error

$$\begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix} = \begin{bmatrix} 0.266367507 & 0.231776026 & 0.266367507 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix} = \begin{bmatrix} -0.733632493 & 0.231776026 & -0.733632493 \end{bmatrix}$$

$$E_{total} = -1.23548896$$

We can see that the error has been lowered from -1.29508197 to -1.23548896 after weights have been updated. After several more iterations, the network should be able to learn the optimized weights needed to reach Y = [1, 0, 1] from X = [1, 0, 1].