Aprendizagem 2023

Labs 7-9: Neural Networks

Practical exercises

I. Multi-layer Perceptron

1. Consider a network with three layers: 5 inputs, 3 hidden units and 2 outputs where all units use a sigmoid activation function.

a) Initialize connection weights to 0.1 and biases to 0. Using the squared error loss do a stochastic gradient descent update (with learning rate η=1) for the training example

$$\{\mathbf{x} = [1\ 1\ 0\ 0\ 0]^T, \mathbf{z} = [1\ 0]^T\}$$

$$\mathbf{W}^{[1]} = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix},\ \mathbf{W}^{[2]} = \begin{pmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{pmatrix},\ \mathbf{b}^{[1]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},\ \mathbf{b}^{[2]} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

We are now ready to do forward propagation

$$\mathbf{z}^{[1]} = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \end{pmatrix},\quad \mathbf{x}^{[1]} = \sigma\begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \end{pmatrix}$$

$$\mathbf{z}^{[2]} = \begin{pmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{pmatrix} \begin{pmatrix} \sigma(0.2) \\ \sigma(0.2) \\ \sigma(0.2) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.16495 \\ 0.16495 \end{pmatrix},\quad \mathbf{x}^{[2]} = \sigma\begin{pmatrix} 0.16495 \\ 0.16495 \end{pmatrix} = \begin{pmatrix} 0.5411 \\ 0.5411 \end{pmatrix}$$

Now we want to do the backward phase using the squared error measure:

$$E(\mathbf{x}^{[2]}, \mathbf{z}) = \frac{1}{2}\left(\mathbf{x}^{[2]} - \mathbf{t}\right)^2$$

In general, we will need to know how to derive all functions in our network.
Let us compute them beforehand:

$$\frac{\partial E(\mathbf{x}^{[2]}, \mathbf{t})}{\partial \mathbf{x}^{[2]}} = \frac{1}{2}\left(2\mathbf{x}^{[2]} - 2\mathbf{t}\right) = \mathbf{x}^{[2]} - \mathbf{t}$$

$$\frac{\partial \mathbf{x}^{[i]}(\mathbf{z}^{[i]})}{\partial \mathbf{z}^{[i]}} = \sigma(\mathbf{z}^{[i]})(1 - \sigma(\mathbf{z}^{[i]}))$$

$$\frac{\partial \mathbf{z}^{[i]}(\mathbf{W}^{[i]}, \mathbf{b}^{[i]}, \mathbf{x}^{[i-1]})}{\partial \mathbf{W}^{[i]}} = \mathbf{x}^{[i-1]}$$

$$\frac{\partial \mathbf{z}^{[i]}(\mathbf{W}^{[i]}, \mathbf{b}^{[i]}, \mathbf{x}^{[i-1]})}{\partial \mathbf{b}^{[i]}} = \mathbf{1}$$

$$\frac{\partial \mathbf{z}^{[i]}(\mathbf{W}^{[i]}, \mathbf{b}^{[i]}, \mathbf{x}^{[i-1]})}{\partial \mathbf{x}^{[i-1]}} = \mathbf{W}^{[i]}$$

To start the recursion, we need the delta from the last layer:

$$\delta^{[2]} = \frac{\partial E}{\partial \mathbf{x}^{[2]}} \circ \frac{\partial \mathbf{x}^{[2]}}{\partial \mathbf{z}^{[2]}} = (\mathbf{x}^{[2]} - \mathbf{z}^{[2]}) \circ \sigma(\mathbf{z}^{[2]})(1 - \sigma(\mathbf{z}^{[2]}))$$

$$= \left(\begin{pmatrix}0.5411\\0.5411\end{pmatrix} - \begin{pmatrix}1\\0\end{pmatrix}\right) \circ \sigma\begin{pmatrix}0.16495\\0.16495\end{pmatrix} \circ \left(1 - \sigma\begin{pmatrix}0.16495\\0.16495\end{pmatrix}\right) = \begin{pmatrix}-0.11394\\0.13437\end{pmatrix}$$

Now, we can use the recursion to compute the delta from the hidden layer

$$\delta^{[1]} = \left(\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{x}^{[1]}}\right)^T \cdot \delta^{[2]} \circ \frac{\partial \mathbf{x}^{[1]}}{\partial \mathbf{z}^{[1]}} = \mathbf{W}^{[2]^T} \cdot \delta^{[2]} \circ \sigma(\mathbf{z}^{[1]}) \circ (1 - \sigma(\mathbf{z}^{[1]}))$$

$$= \begin{pmatrix}0.1 & 0.1\\0.1 & 0.1\\0.1 & 0.1\end{pmatrix} \cdot \begin{pmatrix}-0.11394\\0.13437\end{pmatrix} \circ \sigma\begin{pmatrix}0.2\\0.2\\0.2\end{pmatrix} \circ \left(1 - \sigma\begin{pmatrix}0.2\\0.2\\0.2\end{pmatrix}\right) = \begin{pmatrix}0.00050575\\0.00050575\\0.00050575\end{pmatrix}$$

Finally, we can go to the last phase and perform the updates. We start with the first layer

$$\frac{\partial E}{\partial \mathbf{W}^{[1]}} = \delta^{[1]} \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}} = \delta^{[1]} (\mathbf{x}^{[0]})^T = \begin{pmatrix}0.00050575\\0.00050575\\0.00050575\end{pmatrix} (1 \quad 1 \quad 1 \quad 0 \quad 0) = \begin{pmatrix}0.00050575 & 0.00050575 & 0 & 0 & 0\\0.00050575 & 0.00050575 & 0 & 0 & 0\\0.00050575 & 0.00050575 & 0 & 0 & 0\end{pmatrix}$$

$$\mathbf{W}^{[1]} = \mathbf{W}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[1]}} = \begin{pmatrix}0.1 & 0.1 & 0.1 & 0.1 & 0.1\\0.1 & 0.1 & 0.1 & 0.1 & 0.1\\0.1 & 0.1 & 0.1 & 0.1 & 0.1\end{pmatrix} - \begin{pmatrix}0.00050575 & 0.00050575 & 0 & 0 & 0\\0.00050575 & 0.00050575 & 0 & 0 & 0\\0.00050575 & 0.00050575 & 0 & 0 & 0\end{pmatrix}$$

$$= \begin{pmatrix}0.0995 & 0.0995 & 0.1 & 0.1 & 0.1\\0.0995 & 0.0995 & 0.1 & 0.1 & 0.1\\0.0995 & 0.0995 & 0.1 & 0.1 & 0.1\end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[1]}} = \delta^{[1]} \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{b}^{[1]}} = \delta^{[1]} = \begin{pmatrix}0.00050575\\0.00050575\\0.00050575\end{pmatrix}$$

$$\mathbf{b}^{[1]} = \mathbf{b}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{b}^{[1]}} = \begin{pmatrix}0\\0\\0\end{pmatrix} - \begin{pmatrix}0.00050575\\0.00050575\\0.00050575\end{pmatrix} = \begin{pmatrix}-0.00050575\\-0.00050575\\-0.00050575\end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{W}^{[2]}} = \delta^{[2]} \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{W}^{[2]}} = \delta^{[2]} (\mathbf{x}^{[1]})^T = \begin{pmatrix}-0.11394\\0.13437\end{pmatrix} \sigma\begin{pmatrix}0.2\\0.2\\0.2\end{pmatrix} = \begin{pmatrix}-0.062647 & -0.062647 & -0.062647\\0.073881 & 0.073881 & 0.073881\end{pmatrix}$$

$$\mathbf{W}^{[2]} = \mathbf{W}^{[2]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[2]}} = \begin{pmatrix}0.1 & 0.1 & 0.1\\0.1 & 0.1 & 0.1\end{pmatrix} - \begin{pmatrix}-0.062647 & -0.062647 & -0.062647\\0.073881 & 0.073881 & 0.073881\end{pmatrix}$$

$$= \begin{pmatrix}0.162647 & 0.162647 & 0.162647\\0.026119 & 0.026119 & 0.026119\end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[2]}} = \delta^{[2]} \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{b}^{[2]}} = \delta^{[2]} = \begin{pmatrix}-0.11394\\0.13437\end{pmatrix}$$

$$\mathbf{b}^{[2]} = \mathbf{b}^{[2]} - \eta \frac{\partial E}{\partial \mathbf{b}^{[2]}} = \begin{pmatrix}0\\0\end{pmatrix} - \begin{pmatrix}-0.11394\\0.13437\end{pmatrix} = \begin{pmatrix}0.11394\\-0.13437\end{pmatrix}$$

b) Compute the MLP class for the query point $\mathbf{x}_{new} = [1\ 0\ 0\ 0\ 1]^T$

We use the weights and biases from the previous exercise. To get the class label we need to get the MLP output for the point. To get that, we just need to do forward propagation:

$$\mathbf{z}^{[1]} = \begin{pmatrix}0.0995 & 0.0995 & 0.1 & 0.1 & 0.1\\0.0995 & 0.0995 & 0.1 & 0.1 & 0.1\\0.0995 & 0.0995 & 0.1 & 0.1 & 0.1\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\\1\end{pmatrix} + \begin{pmatrix}0.00050575\\0.00050575\\0.00050575\end{pmatrix} = \begin{pmatrix}0.19949\\0.19949\\0.19949\end{pmatrix}, \quad \mathbf{x}^{[1]} = \sigma\begin{pmatrix}0.19949\\0.19949\\0.19949\end{pmatrix}$$

$$\mathbf{z}^{[2]} = \begin{pmatrix}0.16495\\0.16495\end{pmatrix} \sigma\begin{pmatrix}0.19949\\0.19949\\0.19949\end{pmatrix} + \begin{pmatrix}0.11394\\-0.13437\end{pmatrix} = \begin{pmatrix}0.382162\\-0.091297\end{pmatrix}, \mathbf{x}^{[2]} = \sigma\begin{pmatrix}0.382162\\-0.091297\end{pmatrix} = \begin{pmatrix}0.59439\\0.47719\end{pmatrix}$$

$$label = argmax_i(\mathbf{x}^{[2]}) = 0$$

2. Consider a network with four layers with the following numbers of units 4, 4, 3, 3. Assume all units use the *hyperbolic tangent* activation function.

a) Initialize all connection weights and biases to 0.1. Using the squared error loss do a *stochastic gradient descent* update (with learning rate η=0.1) for the training example:

$$\{\boldsymbol{x} = [1\ 0\ 1\ 0]^T, \boldsymbol{z} = [0\ 1\ 0]^T\}$$

Before moving forward, we recall derive the tanh activation function.

$$\tanh(x) = \frac{2}{1+\exp(-2x)} - 1$$

$$\frac{\partial \tanh(x)}{\partial x} = \frac{\partial}{\partial x}\left(\frac{2}{1+\exp(-2x)} - 1\right) = 1 - \tanh(x)^2$$

We start by writing the connection weights and the biases:

$$\mathbf{W}^{[1]} = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix}, \quad \mathbf{b}^{[1]} = \begin{pmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{pmatrix}$$

$$\mathbf{W}^{[2]} = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix}, \quad \mathbf{b}^{[2]} = \begin{pmatrix} 0.1 \\ 0.1 \\ 0.1 \end{pmatrix}$$

$$\mathbf{W}^{[3]} = \begin{pmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{pmatrix}, \quad \mathbf{b}^{[3]} = \begin{pmatrix} 0.1 \\ 0.1 \\ 0.1 \end{pmatrix}$$

We are now ready to do forward propagation:

$$\mathbf{z}^{[1]} = \begin{pmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.3 \end{pmatrix}, \quad \mathbf{x}^{[1]} = tanh\begin{pmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.3 \end{pmatrix}$$

$$\mathbf{z}^{[2]} = \begin{pmatrix} 0.2165 \\ 0.2165 \\ 0.2165 \end{pmatrix}, \quad \mathbf{x}^{[2]} = tanh\begin{pmatrix} 0.2165 \\ 0.2165 \\ 0.2165 \end{pmatrix}$$

$$\mathbf{z}^{[3]} = \begin{pmatrix} 0.16396 \\ 0.16396 \\ 0.16396 \end{pmatrix}, \quad \mathbf{x}^{[3]} = tanh\begin{pmatrix} 0.16396 \\ 0.16396 \\ 0.16396 \end{pmatrix} = \begin{pmatrix} 0.1625 \\ 0.1625 \\ 0.1625 \end{pmatrix}$$

Now we want to do the backward phase:

$$\frac{\partial E(\mathbf{x}^{[2]}, \mathbf{t})}{\partial \mathbf{x}^{[2]}} = \frac{1}{2} 2\left(\mathbf{x}^{[2]} - \mathbf{t}\right) = \mathbf{x}^{[2]} - \mathbf{t}$$

$$\frac{\partial \mathbf{x}^{[i]}(\mathbf{z}^{[i]})}{\partial \mathbf{z}^{[i]}} = 1 - tanh(\mathbf{z}^{[i]})^2$$

$$\frac{\partial \mathbf{z}^{[i]}(\mathbf{W}^{[i]}, \mathbf{b}^{[i]}, \mathbf{x}^{[i-1]})}{\partial \mathbf{W}^{[i]}} = \mathbf{x}^{[i-1]}$$

$$\frac{\partial \mathbf{z}^{[i]}(\mathbf{W}^{[i]}, \mathbf{b}^{[i]}, \mathbf{x}^{[i-1]})}{\partial \mathbf{b}^{[i]}} = \mathbf{1}$$

$$\frac{\partial \mathbf{z}^{[i]}(\mathbf{W}^{[i]}, \mathbf{b}^{[i]}, \mathbf{x}^{[i-1]})}{\partial \mathbf{x}^{[i-1]}} = \mathbf{W}^{[i]}$$

To start the recursion, we need the delta from the last layer:

$$\delta^{[3]} = \frac{\partial E}{\partial \mathbf{x}^{[3]}} \circ \frac{\partial \mathbf{x}^{[3]}}{\partial \mathbf{z}^{[3]}} = (\mathbf{x}^{[3]} - \mathbf{z}) \circ \left(1 - tanh(\mathbf{z}^{[3]})^2\right) = \begin{pmatrix} 0.15822 \\ -0.81538 \\ 0.15822 \end{pmatrix}$$

Now, we can use the recursion to compute the delta from the hidden layers

$$\delta^{[2]} = \left(\frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{x}^{[2]}}\right)^T \cdot \delta^{[3]} \circ \frac{\partial \mathbf{x}^{[2]}}{\partial \mathbf{z}^{[2]}} = \mathbf{W}^{[3]^T} \cdot \delta^{[3]} \circ \left(1 - tanh(\mathbf{z}^{[2]})^2\right) = \begin{pmatrix} -0.0476 \\ -0.0476 \\ -0.0476 \end{pmatrix}$$

$$\delta^{[1]} = \left(\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{x}^{[1]}}\right)^T \cdot \delta^{[2]} \circ \frac{\partial \mathbf{x}^{[1]}}{\partial \mathbf{z}^{[1]}} = \mathbf{W}^{[2]^T} \cdot \delta^{[2]} \circ \left(1 - tanh(\mathbf{z}^{[1]})^2\right) = \begin{pmatrix} -0.0131 \\ -0.0131 \\ -0.0131 \\ -0.0131 \end{pmatrix}$$

Finally, we can go to the last phase and perform the updates. We start with the first layer:

$$\frac{\partial E}{\partial \mathbf{W}^{[1]}} = \delta^{[1]} \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}} = \delta^{[1]} (\mathbf{x}^{[0]})^T = \begin{pmatrix} -0.0131 & 0 & -0.0131 & 0 \\ -0.0131 & 0 & -0.0131 & 0 \\ -0.0131 & 0 & -0.0131 & 0 \\ -0.0131 & 0 & -0.0131 & 0 \end{pmatrix}$$

$$\mathbf{W}^{[1]} = \mathbf{W}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[1]}} = \begin{pmatrix} 0.10131 & 0.1 & 0.10131 & 0.1 \\ 0.10131 & 0.1 & 0.10131 & 0.1 \\ 0.10131 & 0.1 & 0.10131 & 0.1 \\ 0.10131 & 0.1 & 0.10131 & 0.1 \end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[1]}} = \delta^{[1]} \frac{\partial \mathbf{z}^{[1]^T}}{\partial \mathbf{b}^{[1]}} = \delta^{[1]} = \begin{pmatrix} -0.0131 \\ -0.0131 \\ -0.0131 \\ -0.0131 \end{pmatrix}, \quad \mathbf{b}^{[1]} = \mathbf{b}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{b}^{[1]}} = \begin{pmatrix} 0.10131 \\ 0.10131 \\ 0.10131 \\ 0.10131 \end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{W}^{[2]}} = \delta^{[2]} \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{W}^{[2]}} = \delta^{[2]} (\mathbf{x}^{[1]})^T = \begin{pmatrix} -0.01387 & -0.01387 & -0.01387 & -0.01387 \\ -0.01387 & -0.01387 & -0.01387 & -0.01387 \\ -0.01387 & -0.01387 & -0.01387 & -0.01387 \end{pmatrix}$$

$$\mathbf{W}^{[2]} = \mathbf{W}^{[2]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[2]}} == \begin{pmatrix} 0.101387 & 0.101387 & 0.101387 & 0.101387 \\ 0.101387 & 0.101387 & 0.101387 & 0.101387 \\ 0.101387 & 0.101387 & 0.101387 & 0.101387 \end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[2]}} = \delta^{[2]} \frac{\partial \mathbf{z}^{[2]^T}}{\partial \mathbf{b}^{[2]}} = \delta^{[2]} = \begin{pmatrix} -0.0476 \\ -0.0476 \\ -0.0476 \end{pmatrix}, \quad \mathbf{b}^{[2]} = \mathbf{b}^{[2]} - \eta \frac{\partial E}{\partial \mathbf{b}^{[2]}} = \begin{pmatrix} 0.10476 \\ 0.10476 \\ 0.10476 \end{pmatrix}$$

All that is left is to update the parameters for the output layer

$$\frac{\partial E}{\partial \mathbf{W}^{[3]}} = \delta^{[3]} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{W}^{[3]}} = \delta^{[3]} (\mathbf{x}^{[2]})^T = \begin{pmatrix} 0.03373 & 0.03373 & 0.03373 \\ -0.17384 & -0.17384 & -0.17384 \\ 0.03373 & 0.03373 & 0.03373 \end{pmatrix}$$

$$\mathbf{W}^{[3]} = \mathbf{W}^{[3]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[3]}} = \begin{pmatrix} 0.096627 & 0.096627 & 0.096627 \\ 0.117384 & 0.117384 & 0.117384 \\ 0.096627 & 0.096627 & 0.096627 \end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[3]}} = \delta^{[3]} \frac{\partial \mathbf{z}^{[3]^T}}{\partial \mathbf{b}^{[3]}} = \delta^{[3]} = \begin{pmatrix} 0.15822 \\ -0.81538 \\ 0.15822 \end{pmatrix}, \quad \mathbf{b}^{[3]} = \mathbf{b}^{[3]} - \eta \frac{\partial E}{\partial \mathbf{b}^{[3]}} = \begin{pmatrix} 0.084178 \\ 0.181538 \\ 0.084178 \end{pmatrix}$$

b) Reusing the computations from the previous exercise do a *gradient descent update* (with learning rate η=0.1) for the batch with the training example from the a) and the following:

$$\{\mathbf{x} = [0\ \ 0\ \ 10\ \ 0]^T, \boldsymbol{z} = [0\ 0\ 1]^T\}$$

For ~~previous~~ second observation:

$$\delta^{[3](2)} = \begin{pmatrix} 0.2057 \\ 0.2057 \\ -0.7478 \end{pmatrix}, \quad \delta^{[2](2)} = \begin{pmatrix} -0.0283 \\ -0.0283 \\ -0.0283 \end{pmatrix}, \quad \delta^{[1](2)} = \begin{pmatrix} -0.00305 \\ -0.00305 \\ -0.00305 \\ -0.00305 \end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{W}^{[1]}} = \delta^{[1](1)}\frac{\partial \mathbf{z}^{[1](1)}}{\partial \mathbf{W}^{[1]}} + \delta^{[1](2)}\frac{\partial \mathbf{z}^{[1](2)}}{\partial \mathbf{W}^{[1]}} = \delta^{[1](1)}\left(\mathbf{x}^{[0](1)}\right)^T + \delta^{[1](2)}\left(\mathbf{x}^{[0](2)}\right)^T$$

$$= \begin{pmatrix}-0.0131\\-0.0131\\-0.0131\\-0.0131\end{pmatrix}\begin{pmatrix}1 & 0 & 1 & 0\end{pmatrix} + \begin{pmatrix}-0.00305\\-0.00305\\-0.00305\\-0.00305\end{pmatrix}\begin{pmatrix}0 & 0 & 10 & 0\end{pmatrix} = \begin{pmatrix}-0.10131 & 0 & -0.0436 & 0\\-0.10131 & 0 & -0.0436 & 0\\-0.10131 & 0 & -0.0436 & 0\\-0.10131 & 0 & -0.0436 & 0\end{pmatrix}$$

$$\mathbf{W}^{[1]} = \mathbf{W}^{[1]} - \eta\frac{\partial E}{\partial \mathbf{W}^{[1]}} = \begin{pmatrix}0.10131 & 0.1 & 0.10436 & 0.1\\0.10131 & 0.1 & 0.10436 & 0.1\\0.10131 & 0.1 & 0.10436 & 0.1\\0.10131 & 0.1 & 0.10436 & 0.1\end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[1]}} = \delta^{[1](1)}\frac{\partial \mathbf{z}^{[1](1)^T}}{\partial \mathbf{b}^{[1]}} + \delta^{[1](2)}\frac{\partial \mathbf{z}^{[1](2)^T}}{\partial \mathbf{b}^{[1]}} = \delta^{[1](1)} + \delta^{[1](2)} = \begin{pmatrix}-0.0131\\-0.0131\\-0.0131\\-0.0131\end{pmatrix} + \begin{pmatrix}-0.00305\\-0.00305\\-0.00305\\-0.00305\end{pmatrix} = \begin{pmatrix}-0.0161\\-0.0161\\-0.0161\\-0.0161\end{pmatrix}$$

$$\mathbf{b}^{[1]} = \mathbf{b}^{[1]} - \eta\frac{\partial E}{\partial \mathbf{b}^{[1]}} = \begin{pmatrix}0.1\\0.1\\0.1\\0.1\end{pmatrix} - 0.1\begin{pmatrix}-0.0131\\-0.0131\\-0.0131\\-0.0131\end{pmatrix} = \begin{pmatrix}0.10161\\0.10161\\0.10161\\0.10161\end{pmatrix}$$

After updates:

$$\mathbf{W}^{[2]} = \begin{pmatrix}0.103656 & 0.103656 & 0.103656 & 0.103656\\0.103656 & 0.103656 & 0.103656 & 0.103656\\0.103656 & 0.103656 & 0.103656 & 0.103656\end{pmatrix}, \quad \mathbf{b}^{[2]} = \begin{pmatrix}0.107597\\0.107597\\0.107597\end{pmatrix}$$

$$\mathbf{W}^{[3]} = \begin{pmatrix}0.08846 & 0.08846 & 0.08846\\0.10922 & 0.10922 & 0.10922\\0.12632 & 0.12632 & 0.12632\end{pmatrix}, \quad \mathbf{b}^{[3]} = \begin{pmatrix}0.0636\\0.1609\\0.1589\end{pmatrix}$$

c) Consider the learned MLPs from (a) and (b).
Compute the MLP class for the query point $\mathbf{x}_{new} = [1\ 1\ 1\ 0]^T$
Which has smallest squared error? Which model has better classification accuracy?

$$\mathbf{z}^{[1]} = \begin{pmatrix}0.10131 & 0.1 & 0.10131 & 0.1\\0.10131 & 0.1 & 0.10131 & 0.1\\0.10131 & 0.1 & 0.10131 & 0.1\\0.10131 & 0.1 & 0.10131 & 0.1\end{pmatrix}\begin{pmatrix}1\\1\\1\\0\end{pmatrix} + \begin{pmatrix}0.10131\\0.10131\\0.10131\\0.10131\end{pmatrix} = \begin{pmatrix}0.4039\\0.4039\\0.4039\\0.4039\end{pmatrix}$$

$$\mathbf{z}^{[2]} = \begin{pmatrix}0.101387 & 0.101387 & 0.101387 & 0.101387\\0.101387 & 0.101387 & 0.101387 & 0.101387\\0.101387 & 0.101387 & 0.101387 & 0.101387\end{pmatrix} tanh\begin{pmatrix}0.4039\\0.4039\\0.4039\\0.4039\end{pmatrix} + \begin{pmatrix}0.10476\\0.10476\\0.10476\end{pmatrix} = \begin{pmatrix}0.2602\\0.2602\\0.2602\end{pmatrix}$$

$$\mathbf{z}^{[3]} = \begin{pmatrix}0.096627 & 0.096627 & 0.096627\\0.117384 & 0.117384 & 0.117384\\0.096627 & 0.096627 & 0.096627\end{pmatrix} tanh\begin{pmatrix}0.2602\\0.2602\\0.2602\end{pmatrix} + \begin{pmatrix}0.084178\\0.181538\\0.084178\end{pmatrix} = \begin{pmatrix}0.1579\\0.2712\\0.1579\end{pmatrix}$$

$$\mathbf{x}^{[3]} = tanh\begin{pmatrix}0.1579\\0.2712\\0.1579\end{pmatrix} = \begin{pmatrix}0.1567\\0.2647\\0.1567\end{pmatrix}$$

$$E(\mathbf{x}^{[3]}, \mathbf{t}) = \frac{1}{2}\sum_{i=1}^{2}\left(\mathbf{x}_i^{[3]} - \mathbf{t}_i\right)^2 = 0.8058$$

$$\mathbf{z}^{[1]} = \begin{pmatrix}0.4073\\0.4073\\0.4073\\0.4073\end{pmatrix}, \mathbf{z}^{[2]} = \begin{pmatrix}0.2677\\0.2677\\0.2677\end{pmatrix}, \mathbf{z}^{[3]} = \begin{pmatrix}0.1330\\0.2467\\0.2581\end{pmatrix}, \mathbf{x}^{[3]} = \begin{pmatrix}0.1322\\0.2418\\0.2525\end{pmatrix}$$

$$E(\mathbf{x}^{[3]}, \mathbf{z}) = \frac{1}{2}\sum_{i=1}^{2}\left(\mathbf{x}_i^{[3]} - \mathbf{t}_i\right)^2 = 0.6347$$

Model (b) has a lower error and is the only one that classifies the point correctly. However, the difference between a correct and an incorrect classification is small in terms of squared error.

3. Repeat the exact same exercise, but this time with following adaptations:

- the output units have a *softmax* activation function
- the error function is *cross-entropy*

What are the major differences between using squared error and cross-entropy?

Softmax depends not only on its input but on the inputs of the other units:

$$\text{softmax}([z_1 \quad z_2 \quad \dots \quad z_d]^T) = [x_1 \quad x_2 \quad \dots \quad x_d]^T \quad \text{where } x_i = \frac{\exp(z_i)}{\sum_{k=1}^{d}\exp(z_k)}$$

$$\mathbf{x}^{[3]} = softmax\begin{pmatrix}0.16396\\0.16396\\0.16396\end{pmatrix} = \begin{pmatrix}0.(3)\\0.(3)\\0.(3)\end{pmatrix}$$

Considering cross-entropy:

$$E(\mathbf{x}^{[3]},\mathbf{t}) = -\sum_{i=1}^{d}\mathbf{t}_i\, log\left(\mathbf{x_i}^{[3]}\right)$$

The derivative of the error function, $\frac{\partial E}{\partial \mathbf{W}^{[3]}} = \frac{\partial E}{\partial \mathbf{z}^{[3]}}\frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{W}^{[3]}}$, will yield the following component

$$\delta^{[3]} = \frac{\partial E}{\partial \mathbf{z}^{[3]}} = \frac{\partial}{\partial \mathbf{z}^{[3]}}\left(-\sum_{i=1}^{d}\mathbf{t}_i\, log\left(\mathbf{x}_i{}^{[3]}\right)\right) = \dots some\ important\ steps \dots = \mathbf{x}^{[3]} - \mathbf{t}$$

We can further show that the remaining derivatives can be computed as before (omitted). Assuming this, let us then compute the updates,

$$\delta^{[3]} = \mathbf{x}^{[3]} - \mathbf{t} = \begin{pmatrix}0.(3)\\0.(3)\\0.(3)\end{pmatrix} - \begin{pmatrix}0\\1\\0\end{pmatrix} = \begin{pmatrix}0.(3)\\-0.(6)\\0.(3)\end{pmatrix}$$

Then recursion

$$\delta^{[2]} = \begin{pmatrix}0\\0\\0\end{pmatrix}, \delta^{[1]} = \begin{pmatrix}0\\0\\0\\0\end{pmatrix}$$

meaning that $\mathbf{W}^{[1]}$, $\mathbf{b}^{[1]}$, $\mathbf{W}^{[2]}$ and $\mathbf{b}^{[2]}$ will be unchanged. So the output layer:

$$\frac{\partial E}{\partial \mathbf{W}^{[3]}} = \delta^{[3]}\frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{W}^{[3]}} = \delta^{[3]}(\mathbf{x}^{[2]})^T = \begin{pmatrix}0.(3)\\-0.(6)\\0.(3)\end{pmatrix}\begin{pmatrix}0.2132\\0.2132\\0.2132\end{pmatrix} = \begin{pmatrix}0.0711 & 0.0711 & 0.0711\\-0.1421 & -0.1421 & -0.1421\\0.0711 & 0.0711 & 0.0711\end{pmatrix}$$

$$\mathbf{W}^{[3]} = \mathbf{W}^{[3]} - \eta\frac{\partial E}{\partial \mathbf{W}^{[3]}} = \begin{pmatrix}0.0929 & 0.0929 & 0.0929\\0.1142 & 0.1142 & 0.1142\\0.0929 & 0.0929 & 0.0929\end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[3]}} = \delta^{[3]}\frac{\partial \mathbf{z}^{[3]T}}{\partial \mathbf{b}^{[3]}} = \delta^{[3]} = \begin{pmatrix}0.(3)\\-0.(6)\\0.(3)\end{pmatrix}, \quad \mathbf{b}^{[3]} = \mathbf{b}^{[3]} - \eta\frac{\partial E}{\partial \mathbf{b}^{[3]}} = \begin{pmatrix}0.0667\\0.1667\\0.0667\end{pmatrix}$$

II. (*Optional*) Model Complexity

4. [*optional*] For the following scenarios which has the smallest number of parameters?

a) three-dimensional real inputs classified by

   i. MLP with one hidden layer with the following units per layer 3 2 2

   ii. simple Bayesian classifier with multivariate gaussian likelihood function

We know that the VC dimension is a measure of the degrees of freedom of a classifier. A good proxy for the degrees of freedom is the number of parameters. So, let us estimate the number of parameters per scenario.

1. Between the input layer and the hidden layer we will have:

- a 2×3 weight matrix W[1], so 6 parameters
- a 3×1 bias vector b[1], so 3 parameters

Between the hidden layer and the output layer we will have:

- a 2×2 weight matrix W[2], so 4 parameters
- a 2×1 bias vector b[2], so 2 parameters

There are no more parameters, so the total amounts to 6 + 3 + 4 + 2 = 15.

2. For the simple Bayesian classifier, we need to estimate prior and likelihood.
The prior is a distribution table with two entries (one for each class): $p(z = 0) = 1, p(z = 1) = 1 - p$

So, we have one parameter for the prior.

The likelihood $p(y|z = 0)$, being a multivariate gaussian, will require a mean vector and a covariance matrix:

- for three dimensions the mean vector is a 3 × 1 vector, so 3 parameters.
- for three dimensions, the covariance is a 3 × 3 matrix Σ, however the matrix is symmetric so, we only need to count the diagonal and upper diagonal part of the matrix. So, 6 parameters.

The likelihood $p(y|z = 1)$, being a multivariate gaussian, requires the same number of parameters.

There are no more parameters to estimate, so the total is $2 \times (3 + 6) + 1 = 19$.

3. Finally, we can produce our guess on the VC dimensions of both classifiers. We would approximately say that the MLP classifier has a smaller VC dimension, d_{VC}(MLP) < d_{VC}(Bayesian).

b) *N*-dimensional real inputs classified by
  i. perceptron
  ii. MLP with two hidden layers with the following units per layer N, $\frac{N}{2}$, $\frac{N}{2}$, 2
  iii. naive Bayes with Gaussian likelihoods
  iv. simple Bayesian classifier with multivariate gaussian likelihood function

1. The perceptron has a weight vector $\mathbf{w} = (\mathrm{w}_1\ \mathrm{w}_2 \ldots \mathrm{w}_N)^T$ and a bias w_0. So, it has $N + 1$ parameters.

2. Between the input layer and the first hidden layer we will have:
– a N/2 × N weight matrix W[1], so $\mathrm{N}^2/2$ parameters
– a N/2 × 1 bias vector b[1], so N/2 parameters

Between the first hidden layer and the second one we will have:
– a N/2 × N/2 weight matrix W[2], so $\mathrm{N}^2/4$ parameters
– a N/2 × 1 bias vector b[2], so N/2 parameters

Between the second hidden layer and the output layer we will have:
– A 2 × N/2 weight matrix W[3], so N parameters
– A 2 × 1 bias vector b[3], so 2 parameters
There are no more parameters, so the total amounts $\frac{3N^2+8N+8}{4}$

3. For the Naive Bayesian classifier, we need to estimate prior and likelihoods. One parameter for the prior.

The likelihoods are one dimensional gaussians with two parameters each: mean and standard deviation. So for each class c and feature y_j we have a distribution $p(y_j \mid z = c)$ that requires two parameters. Since there are 2 classes and N features, we have $2 \times N \times 2 = 4N$ parameters for the likelihoods.

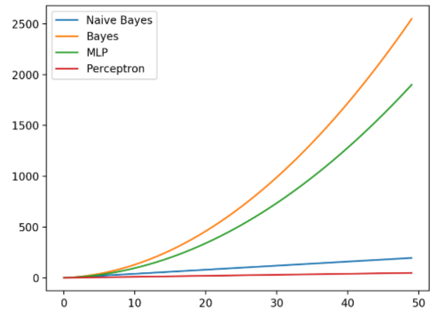
There are no more parameters to estimate, so the total is $1 + 4N$.

4. For the simple Bayesian classifier, we need to estimate prior and likelihood. We have one parameter for the prior. The likelihood $p(y \mid z=0)$, being a multivariate gaussian, require a mean vector and covariance matrix:
– for N dimensions the mean vector is an $N \times 1$ vector, so N parameters.
– for three dimensions, the covariance is an $N \times N$ matrix, however the matrix is symmetric so, we only need to count the diagonal and upper diagonal part of the matrix.
So, $N + \frac{N^2-N}{2}$ parameters. The likelihood $p(y \mid z=1)$ requires the same number of parameters.
There are no more parameters to estimate, so the total is

$$1 + 2 \times \left(N + N + \frac{N^2 - N}{2}\right) = 1 + 2N + 2N + N^2 - N = 1 + 3N + N^2.$$

5. Finally, we can plot our guesses on the VC dimensions of the four scenarios

We would approximately order VC dimensions as follows
d_{VC}(Perceptron) < d_{VC}(NB) < d_{VC}(MLP) < d_{VC} (Bayes)



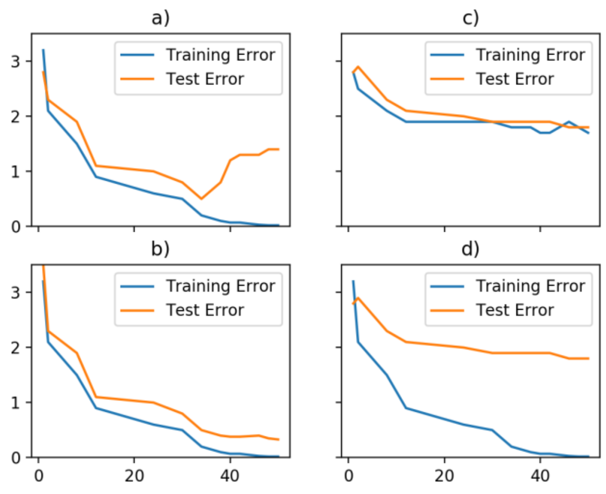
5. [*optional*] Choose between increase, decrease, maintain for each of the following factors:

- training data
- regularization
- number of parameters

when considering the behavior of the learned network at the 50th iteration.

Justify each decision.

a) Increase, increase, maintain/decrease
b) Increase, maintain, maintain
c) Increase, decrease, increase
d) Increase, increase, maintain/decrease



Programming quest

6. Consider a 10-fold CV, and MLPs with a single hidden layer with 5 nodes. Using *sklearn*:

a) assess the classification accuracy of the MLP on the *iris* data using a cross-entropy loss
b) assess the MAE of the MLP on the *housing* data using a squared error loss

Resource: https://scikit-learn.org/stable/modules/neural_networks_supervised.html