Lab 4: Weight Training: A Numerical Example

ECSE 421: Embedded Systems
Department of Electrical and Computer Engineering
McGill University
Version 1.1

Adam Cavatassi

Winter 2018

This document demonstrates a numerical example of one training iteration of the neural network example in the Lab 4 instructions. Recall the hypothetical neural network in Fig. 1. This network has 2 inputs, a hidden layer with 2 nodes, and an output layer with 2 nodes. The hidden and output layer both use the sigmoid function as an activation. The network is to be trained using the mean squared error cost function. This example will utilize stochastic gradient descent with a batch size of 1 and a learning rate $\alpha = 0.1$. All numbers in this example were selected at random. Lets say that hypothetically, the batch for this iteration of training has the following input and target data:

$$\mathbf{i} = \begin{bmatrix} 0.25 \\ -0.31 \end{bmatrix} \quad \mathbf{t} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{1}$$

Your two weight matrices are initialized at random such that $\mathbf{W_k} \in U[-1, 1]$:

$$\mathbf{W_0} = \begin{bmatrix} 0.81 & -0.03 \\ -0.63 & -0.21 \end{bmatrix} \quad \mathbf{W_1} = \begin{bmatrix} -0.76 & 0.55 \\ 0.21 & -0.11 \end{bmatrix} \tag{2}$$

We are now ready to complete a full iteration of training.

Forward pass:

First we need to compute the dot product for the hidden nodes:

$$\mathbf{net^0} = \begin{bmatrix} net_0^0 \\ net_1^0 \end{bmatrix} = \begin{bmatrix} i_0 w_0^0 + i_1 w_0^1 \\ i_0 w_1^0 + i_1 w_1^1 \end{bmatrix} = \begin{bmatrix} (0.25)(0.81) + (-0.31)(-0.63) \\ (0.25)(-0.03) + (-0.31)(-0.21) \end{bmatrix} = \begin{bmatrix} 0.3978 \\ 0.0576 \end{bmatrix} \tag{3}$$

Next, we compute the activations for the hidden nodes:

$$\mathbf{y^0} = \begin{bmatrix} y_0^0 \\ y_1^0 \end{bmatrix} = \begin{bmatrix} \sigma(net_0^0) \\ \sigma(net_1^0) \end{bmatrix} = \begin{bmatrix} \sigma(0.3978) \\ \sigma(0.0576) \end{bmatrix} = \begin{bmatrix} 0.5982 \\ 0.5144 \end{bmatrix} \tag{4}$$

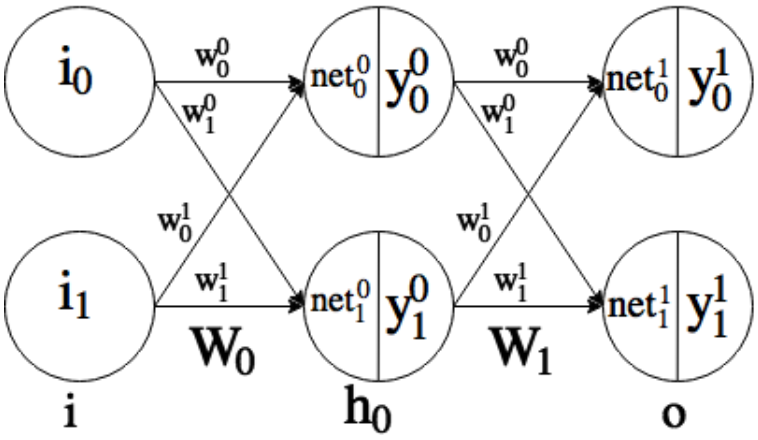

Figure 1: Example neural network.

Proceed to compute the dot product for the output layer:

$$\mathbf{net^1} = \begin{bmatrix} net_0^1 \\ net_1^1 \end{bmatrix} = \begin{bmatrix} y_0^0 w_0^0 + y_1^0 w_0^1 \\ y_0^0 w_1^0 + y_1^0 w_1^1 \end{bmatrix} = \begin{bmatrix} (0.5982)(-0.76) + (0.5144)(0.21) \\ (0.5982)(0.55) + (0.5144)(-0.11) \end{bmatrix} = \begin{bmatrix} -0.3466 \\ 0.2724 \end{bmatrix} \tag{5}$$

Finally, compute the activation for the output layer:

$$\mathbf{y^1} = \begin{bmatrix} y_0^1 \\ y_1^1 \end{bmatrix} = \begin{bmatrix} \sigma(net_0^1) \\ \sigma(net_1^1) \end{bmatrix} = \begin{bmatrix} \sigma(-0.3466) \\ \sigma(0.2724 \end{bmatrix} = \begin{bmatrix} 0.4142 \\ 0.5677 \end{bmatrix} \tag{6}$$

The forward pass is now complete, and we can move on to the backward pass.

Backward pass:

Working backwards, we begin by computing the gradient for the output layer:

$$\begin{aligned}
\nabla J(\mathbf{W_1}) &= \boldsymbol{\delta^1} \times \mathbf{y^0}^\top \\
&= [(\mathbf{y^1} - \mathbf{t})\sigma(\mathbf{net^1})(1 - \sigma(\mathbf{net^1})] \times \mathbf{y^0}^\top \\
&= \left[\left(\begin{bmatrix} 0.4142 \\ 0.5677 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \sigma \begin{bmatrix} -0.3466 \\ 0.2724 \end{bmatrix} \left(1 - \sigma \begin{bmatrix} -0.3466 \\ 0.2724 \end{bmatrix} \right) \right] \times [0.5982 \quad 0.5144] \\
&= \begin{bmatrix} -0.1421 \\ 0.1393 \end{bmatrix} \times [0.5982 \quad 0.0.5144] \\
&= \begin{bmatrix} -0.085 & -0.0730 \\ 0.0833 & 0.0716 \end{bmatrix}
\end{aligned}$$

And then the gradient for the hidden layer:

$$
\begin{aligned}
\nabla J(\mathbf{W_0}) &= \boldsymbol{\delta^0} \times \mathbf{i}^\top \\
&= [(\mathbf{W_1} \times \boldsymbol{\delta^1})\sigma(\mathbf{net^0})(1-\sigma(\mathbf{net^0})] \times \mathbf{i}^\top \\
&= \left[\left(\begin{bmatrix} -0.76 & 0.55 \\ 0.21 & -0.11 \end{bmatrix} \times \begin{bmatrix} -0.1421 \\ 0.1393 \end{bmatrix}\right)\sigma\begin{bmatrix} 0.3978 \\ 0.0576 \end{bmatrix}\left(1-\sigma\begin{bmatrix} 0.3978 \\ 0.0576 \end{bmatrix}\right)\right] \times [0.25 \quad -0.31] \\
&= \left[\begin{bmatrix} 0.1846 \\ -0.0452 \end{bmatrix}\begin{bmatrix} 0.5982 \\ 0.5144 \end{bmatrix}\left(1-\begin{bmatrix} 0.5982 \\ 0.5144 \end{bmatrix}\right)\right] \times [0.25 \quad -0.31] \\
&= \begin{bmatrix} 0.0444 \\ -0.0113 \end{bmatrix} \times [0.25 \quad -0.31] \\
&= \begin{bmatrix} 0.0111 & -0.0138 \\ -0.0028 & 0.0035 \end{bmatrix}
\end{aligned}
$$

Now we can update the weights for both layers:

$$
\begin{aligned}
\mathbf{W_0}_{new} &= \mathbf{W_0}_{old} - \alpha\nabla J(\mathbf{W_0}) \\
&= \begin{bmatrix} 0.81 & -0.03 \\ -0.63 & -0.21 \end{bmatrix} - 0.1\begin{bmatrix} 0.0111 & -0.0138 \\ -0.0028 & 0.0035 \end{bmatrix} \\
&= \begin{bmatrix} 0.81 & -0.03 \\ -0.63 & -0.21 \end{bmatrix} - \begin{bmatrix} 0.00111 & -0.00138 \\ -0.00028 & 0.00035 \end{bmatrix} \\
&= \begin{bmatrix} 0.80889 & -0.02862 \\ -0.62972 & -0.21035 \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{W_1}_{new} &= \mathbf{W_1}_{old} - \alpha\nabla J(\mathbf{W_1}) \\
&= \begin{bmatrix} -0.76 & 0.55 \\ 0.21 & -0.11 \end{bmatrix} - 0.1\begin{bmatrix} -0.085 & -0.0730 \\ 0.0833 & 0.0716 \end{bmatrix} \\
&= \begin{bmatrix} -0.76 & 0.55 \\ 0.21 & -0.11 \end{bmatrix} - \begin{bmatrix} -0.0085 & -0.00730 \\ 0.00833 & 0.00716 \end{bmatrix} \\
&= \begin{bmatrix} 0.7515 & -0.0.5573 \\ 0.20167 & -0.11716 \end{bmatrix}
\end{aligned}
$$

We have now completed one full training iteration. This process must be repeated until the training and validation error are sufficiently low.