Spring 2025 CPSC 483.04 INTRODUCTION TO MACHINE LEARNING

HW6 (35 points)

Submission deadline: 11:59 PM, Friday, April 25

Neural Networks

A typical neural network consists of an input layer, one or more hidden layers, and an output layer. A perceptron is the simplest form of a neural network. It's essentially a single neuron model, developed by Frank Rosenblatt in the 1950s.

The input of a neural network is denoted by the variables $x_1, x_2, \ldots, x_n$. The output of a single unit perceptron is denoted by O. It is obtained by applying a function g to a weighted sum h of the input:

$$O = g(h)$$

$$h = \sum_{i=1}^{n} w_i x_i$$

The input to the learning process is a set of training examples $e_1, e_2, \ldots, e_t$. An example is a set of feature values of $x_1, x_2, \ldots, x_n$ with a target value y. The objective of a learning algorithm is to obtain a set of weights $w_1, w_2, \ldots, w_n$ such that the output O is a good approximation of the target value y for all the examples. Specifically, we try to minimize an error/loss function such as:

$$E = \sum_{i=1}^{t} (O_i - y_i)^2$$

where the summation is over all the examples. O_i is the value of O that is obtained by putting the values of $x_1, x_2, \ldots, x_n$ of the example e_i into the function g, and y_i is the target value of the example e_i.

The error E can be minimized by the *gradient descent* algorithm. This algorithm is implemented by repeatedly changing the values of the weights $w_1, w_2, \ldots, w_n$ for a single example $e_i = (x_1, x_2, \ldots, x_n, y)$ according to the update form:

$$w_i \leftarrow w_i + \epsilon \delta x_i \quad \text{for } i = 1, 2, \ldots, n$$

$$\text{where } \delta = (y - O) g'(h)$$

$O = g(h)$

$\int g'(h) \rightarrow \sum_{i=1}^{n} w_i x_i$

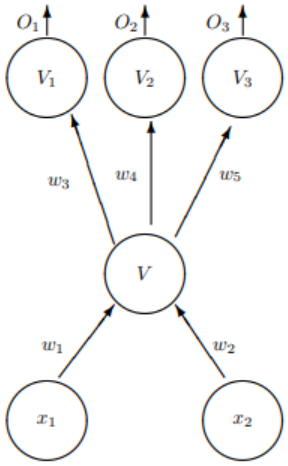
And in this form, it is called *the Delta Rule*. Sometimes it is called *the Adaline Rule*, or *the Widrow-Hoff Rule.*

Reference used for problems: The *Delta Rule* for a sigmoidal unit is:

$$w_i \leftarrow w_i + \epsilon\delta x_i \quad \text{for } i = 1, 2, \ldots, n$$
$$\text{where } \delta = O(1 - O)(y - O)$$

Problem 1 (25 pts)

You are given the following neural network with two layers (one hidden layer), two inputs x_1, x_2, three outputs O_1, O_2, O_3, and all neurons are implemented with the sigmoid function. There are NO bias connections.



Problem 1.1 Give explicit expressions to the values of all neurons in ***forward propagation*** when the network is given the input $x_1 = 3$, $x_2 = 9$, with the target output $y_1 = 1$, $y_2 = 0$, $y_3 = 1$. Your answer should be in terms of the old weights w_1, w_2, w_3, w_4, w_5. You may use the notation $S(\cdot)$ instead of explicitly computing sigmoid values.

$x_1 = 3$, $x_2 = 9$

(7 pts) Answer:

$V = S(w_1 \cdot x_1 + w_2 x_2)$
$V_1 = S(w_3 \times V)$
$V_2 = S(w_4 \times V)$
$V_3 = S(w_5 \times V)$
$O_1 = V_1$
$O_2 = V_2$
$O_3 = V_3$

1.2 Give explicit expressions to how the weights change by ***backpropagation*** when the network is given the same example as above. Use $\epsilon = 0.1$.Your answer should be in terms of the old weights w_1, w_2, w_3, w_4, w_5, and the node values V, V_1, V_2, V_3, O_1, O_2, O_3, that were computed in Problem **1.1**. You may use the notation $S(\cdot)$ instead of explicitly computing sigmoid values. You may use extra temporary variables in your answer to represent the updated weights, but make sure that they are defined in terms of the given variables or given numerical values.

(8 pts) Answer: (δ_1, δ_2, δ_3, and δ correspond to neurons V_1, V_2, V_3, and V, respectively.)

$\delta_1 =$ $\delta_1 = O_1(1-O_1)(y_1 - O_1)$

$\delta_2 =$ $\delta_2 = O_2(1-O_2)(y_2 - O_2)$

$\delta_3 =$ $\delta_3 = O_3(1-O_3)(y_3 - O_3)$

$\delta =$ $\delta = V(1-V)(w_3\delta_1 + w_4\delta_2 + w_5\delta_3)$

(10 pts) Answer:

new $w_1 =$ $w_1 = w_1 + (0.1)(\delta)(x_1)$

new $w_2 =$ $w_2 = w_2 + (0.1)(\delta)(x_2)$

new $w_3 =$ $w_3 = w_3 + (0.1)(\delta_1)(V)$

new $w_4 =$ $w_4 = w_4 + (0.1)(\delta_2)(V)$

new $w_5 =$ $w_5 = w_5 + (0.1)(\delta_3)(V)$

Problem 2: Training NN and CNN using TensorFlow and Keras (10 pts)

The provided sample Jupyter notebook loads the Fashion MNIST training data from Keras, defines a fully connected neural network with two hidden layers, and trains and evaluates the network.

Your tasks are:

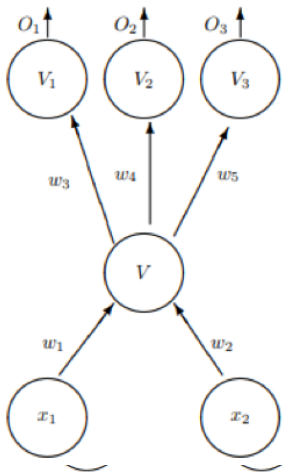
1. **[5 pts]** Implement the provided sample fully connected NN and print out the values of "Test loss" and "Test accuracy".
2. **[5 pts]** Using the provided sample NN code as a reference, design the model to implement a Convolutional Neural Network (CNN) using the Keras Sequential API. The CNN should include two 2D convolutional layers with 256 and 128 filters respectively, both using 3x3 kernel and ReLu activation. Follow each convolutional layer with a max pooling layer of size (2,2). Then, add a flatten layer and a dropout layer with a rate of 0.5. Ensure the model includes input and output layers. Configure the model to use the "adam" optimizer and train it with a batch size = 128 for 10 epochs. Lastly, print out the values "Test loss" and "Test accuracy".

Your submission:

Submit a Jupyter notebook file that includes the required printed values for both models, along with your full name and CWID#.

Problem 1 (25 pts)

You are given the following neural network with two layers (one hidden layer), two inputs x_1, x_2, three outputs O_1, O_2, O_3, and all neurons are implemented with the sigmoid function. There are NO bias connections.



Problem 1.1 Give explicit expressions to the values of all neurons in ***forward propagation*** when the network is given the input $x_1 = 3$, $x_2 = 9$, with the target output $y_1 = 1$, $y_2 = 0$, $y_3 = 1$. Your answer should be in terms of the old weights w_1, w_2, w_3, w_4, w_5. You may use the notation $S(\cdot)$ instead of explicitly computing sigmoid values.

$$\begin{bmatrix} x_1 = 3 \\ x_2 = 9 \end{bmatrix} \quad \begin{bmatrix} y_1 = 1 \\ y_2 = 0 \\ y_3 = 1 \end{bmatrix} \quad \begin{bmatrix} w_1 \\ \vdots \\ w_5 \end{bmatrix} \quad \sigma(z) = \frac{1}{1+e^{-z}}$$

$$V = S(h) = S(w_1 x_1 + w_2 x_2)$$

$$O_i = V_i \;\therefore\; O_1 = V_1,\ O_2 = V_2,\ \cdots,\ O_n = V_n$$

1.2 Give explicit expressions to how the weights change by ***backpropagation*** when the network is given the same example as above. Use $\epsilon = 0.1$.Your answer should be in terms of the old weights w_1, w_2, w_3, w_4, w_5, and the node values V, V_1, V_2, V_3, O_1, O_2, O_3, that were computed in Problem **1.1**. You may use the notation $S(\cdot)$ instead of explicitly computing sigmoid values. You may use extra temporary variables in your answer to represent the updated weights, but make sure that they are defined in terms of the given variables or given numerical values.

$$\epsilon = 0.1, \quad \begin{bmatrix} w_1 \\ \vdots \\ w_5 \end{bmatrix},$$

$$\delta_i = O_i(1 - O_i)(y_i - O_i)$$

$O = g(h)$

$$\rightarrow \int g'(h) \rightarrow \sum_{i=1}^{n} w_i x_i$$

$$\delta = V(1 - V)(w_3 \delta_1 + w_4 \delta_2 + w_4 \delta_3)$$

$$\delta_1 = O_1 (1-O_1)(y_1 - O_1)$$
$$\delta_2 = O_2 (1-O_2)(y_2 - O_2)$$
$$\delta_3 = O_3 (1-O_3)(y_3 - O_3)$$
$$\delta = V(1-V)(w_3 \delta_1 + w_4 \delta_2 + w_5 \delta_3)$$

Weights

$$w_1 = w_1 + (\epsilon)(\delta)(x_1)$$
$$w_2 = w_2 + (\epsilon)(\delta)(x_2)$$
$$w_3 = w_3 + (\epsilon)(\delta)(V)$$
$$w_4 = w_4 + (\epsilon)(\delta)(V)$$
$$w_5 = w_5 + (\epsilon)(\delta)(V)$$

New weights

$$w_1 = w_1 + (0.1)(\delta)(x_1^3)$$
$$w_2 = w_2 + (0.1)(\delta)(x_2^9)$$
$$w_3 = w_3 + (0.1)(\delta_1)(V)$$
$$w_4 = w_4 + (0.1)(\delta_2)(V)$$
$$w_5 = w_5 + (0.1)(\delta_3)(V)$$