Homework#3

CS576 Machine Learning, Fall 2023

Due: November 17

Instruction

- Compile your work into a single file, naming it "*YourLastName_FirstName*_CS576_HW3.zip"
- Structure your submission with individual directories (or files) for each part, namely Part I, Part II, Part III, etc.
- Label your answer clearly with the corresponding question number such as Part I Q1 (a), Part II Q2 (1), etc.

Part I. Problem Solving

Provide your answers to the following questions:

1. The figure below illustrates the topology of a feedforward neural network with two input neurons (Neurons 1 and 2), two hidden processing neuron (Neurons 3, and 4), and two output neurons (Neurons 5 and 6). All processing neurons, including the output neurons, use a **rectifier linear** activation function. The input (x_1 and x_2) to the network is Neuron 1 = 0.3 and Neuron 2 = 0.6. The desired output for this input is Neuron 5 = 0.7 and Neuron 6 = 0.4. The initial weights are provided in Figure 1.

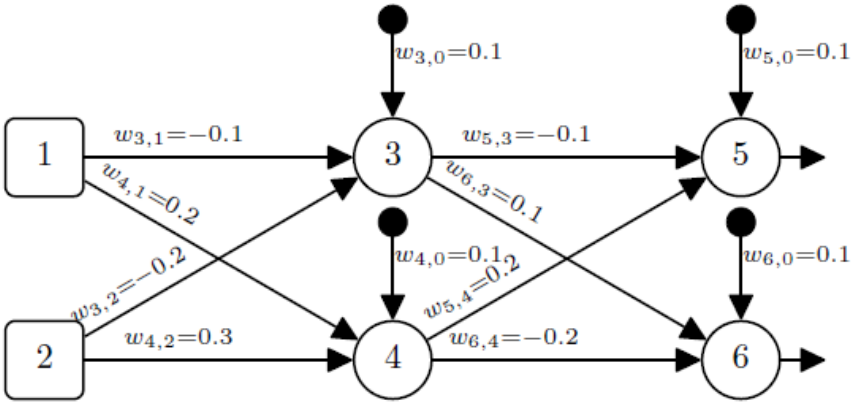


Figure 1. A network definition

(a) Calculate the output generated by the network in response to this input. Show the process of output generation, including the ***z* and *a* values for each neuron** in the network.

(b) Calculate the **sum of squared errors** for this network on this input.

(c) Calculate the **error delta (δ)** for each of the processing neurons in the network (i.e., $\delta_6, \delta_5, \delta_4, \delta_3$).

(d) Using the δ values you have calculated, determine the **sensitivity of the error of the network to changes in each of the weights** (i.e., $\frac{\partial \mathcal{E}}{\partial w_{6,4}}, \frac{\partial \mathcal{E}}{\partial w_{6,3}}, \frac{\partial \mathcal{E}}{\partial w_{6,0}}, \frac{\partial \mathcal{E}}{\partial w_{5,4}}, \frac{\partial \mathcal{E}}{\partial w_{5,3}}, \frac{\partial \mathcal{E}}{\partial w_{5,0}}, \frac{\partial \mathcal{E}}{\partial w_{4,2}}, \frac{\partial \mathcal{E}}{\partial w_{4,1}}, \frac{\partial \mathcal{E}}{\partial w_{4,0}}, \frac{\partial \mathcal{E}}{\partial w_{3,2}}, \frac{\partial \mathcal{E}}{\partial w_{3,1}}, \frac{\partial \mathcal{E}}{\partial w_{3,0}}$)

(e) Assuming **a learning rate (α) of 0.1**, update the **weights** of the network ($w_{6,4}, w_{6,3}, w_{6,0}, w_{5,4}, w_{5,3}, w_{5,0}, w_{4,2}, w_{4,1}, w_{4,0}, w_{3,2}, w_{3,1}, w_{3,0}$) after processing this input data.

(f) Calculate **the new output** (***z* and *a* values for each neuron**) of this network on the input with the updated weights.

(g) Calculate the **sum of squared errors** of this network on the input with the new weights.

(h) Show the **reduction in error** achieved by the network with new weights, compared to using the original weights.

Part II. Neural Network from Scratch

2. Develop a program which trains the neural network based on the network description in Part I, and generates the output for questions Q1 (a) – (h) in Part I and additional (i) below:

(i) Generate **a plot showing the sum of squared errors** of the network changed during training.

The network architecture and training specifications are as follows:

- The network has two input neurons (Neurons 1 and 2), two hidden neuron (Neurons 3, and 4), and two output neurons (Neurons 5 and 6).
- All neurons use a *rectifier activation function* (i.e., rectified linear units (ReLU)).
- The input to the network is Neuron 1 = 0.3 and Neuron 2 = 0.6.
- The desired output for this input is Neuron 5 = 0.7 and Neuron 6 = 0.4.
- The initial weights are provided in the Figure 1.
- The learning rate is 0.1

Submit the following:

(1) The program code. Annotate the relevant lines in your code with the corresponding task numbers for clarity.

(2) The execution results of tasks (a) - (i).

(3) Compare the outputs of (a) - (h) from your program execution and your answers for (a) - (h) in Part I. **Ensure they are the same.**

3. Develop the neural network program based on the network description in Q2 but with the ***logistic* (*sigmoid*) activation function, randomly sampled initial weighs from a range of [-0.5, +0.5]** , and **100 epochs**. Generate a plot showing the sum of squared errors of the network changed during training.

The architecture and training specifications of this network are as follows:

- The network has two input neurons (Neurons 1 and 2), two hidden neuron (Neurons 3, and 4), and two output neurons (Neurons 5 and 6)
- **All neurons use a *logistic* (*sigmoid*) activation function.**
- The input to the network is Neuron 1 = 0.3 and Neuron 2 = 0.6
- The desired output for this input is Neuron 5 = 0.7 and Neuron 6 = 0.4.
- **Randomly sampled initial weighs from a range of [-0.5, +0.5]**
- The learning rate is 0.1
- **Epochs = 100**

Submit the following:
(1) The program code.

(2) The generated plot

4. Analyze the performance of the network model from Q2 and the network model from Q3. Which network shows a better performance?

Part III. Application: Wine Quality Prediction

You are tasked with developing a neural network model to predict the quality of wine based on physicochemical tests. You are free to use any neural networks APIs such as Keras for this task.

0. Data

- You will utilize the "Wine Quality" dataset, a small dataset that contains various attributes of wines along with a quality rating.
- To begin, download a wine quality dataset from UCI Machine Learning Repository, https://archive.ics.uci.edu/dataset/186/wine+quality or attached 'winequality-red.csv'.
- For a detailed description of the data, please refer to the webpage description. It consists of 11 descriptive variables and one output variable, which is 'quality' (with score between 0 and 10).

1. Data Preparation

(a) For this experiment, you will use only the red win dataset for this experiment. Load the red wine dataset ('**winequality-red.csv**').
(b) **Standardizes** the features by removing the mean and scaling to unit variance.
(c) Split the data into **a training set (80%) and a testing set (20%).**

2. Model Design

(d) Design an artificial neural network with the following specifications:

- **A fully connected neural network of a depth of 3.**
  - The input layer has 11 neurons corresponding to the 11 descriptive variables.
  - Two hidden layers : the first hidden dense layer with 11 neurons and the second hidden dense layer with 8 neurons
  - The output dense layer with a single neuron
- Utilize the Rectified Linear Unit (**ReLU**) for all layers, except the output layer.
- **For the output layer, use no activation function** (or a linear activation function), which is appropriate for this regression problem. The output neuron's output is a linear combination of its inputs.

3. Model Training

(e) Configure the model for training as follows:

- Use the mean squared error (**MSE**) as the loss (error) function for the network.
- Employee a first-order gradient-based optimization algorithm for updating the weights, such as Gradient Descent, which serves as the basic form of gradient descent or "**Adam**", a more advanced optimization approach.
- Set the **batch size to 10** examples per gradient update for training, ensuring that the model's weights are updated after every 10 examples have passed through the network.
- Train the model by passing the entire dataset through the neural network for **25 epochs**).
- Set the **learning rate to 0.001**.
- Set aside **10% of the training data as validation data.**
- Specify both the mean squared error (**MSE**) and the mean absolute error (**MAE**) to assess the model's performance. These metrics are typically calculated for both the training and validation sets during the training process.

4. Model Evaluation

(f) Present the results of two performance metrics (**MSE** and **MAE**) **calculated on the validation set**, as well as the **final loss function value (MSE)** of the model on the validation set**.**
The loss function value should be the value after the model has completed training for the specified number of epochs.

(g) Assess the trained model's performance **on the test set** by computing MSE, MAE and R^2.
Report the values for MSE, MAE and R^2 .

5. Visualization

(h) Create a plot illustrating the **loss curves**, also known as learning curves. These curves will visualize how the network's error (loss values) changes over epochs **for both the training set and the validation set**.

(i) Construct **a scatter plot** to display the relationship between **actual and predicted "quality" scores,** with "quality" is the target variable.

Submit the following:

(1) Program code that addresses tasks (a) through (i). Please annotate the relevant lines in your code with the corresponding task numbers for clarity.

(2) Execution results for tasks (f) , (g), (h) and (i).

Part IV. Comparison of Network Models

You are tasked with developing two distinct neural network models, each with a different number of hidden layers, and subsequently comparing their performance.

0. Data

For this task, you will continue using the same "Wine Quality" dataset as in Part III

1. Data Preparation

Data preparation remains consistent with Part III.

(a) Load the red wine dataset (‘**winequality-red.csv’**)
(b) Standardizes the features by centering them on the mean and scaling to have unit variance.
(c) Divide the data into a training set (80%) and a testing set (20%)

2. Model Design

(d) Design two artificial neural networks with the following specifications:

i. **NetworkTwoHidden**:
This model retains the same architecture as the network in Part II
- A fully connected neural network with a depth of 3.
  - The input layer accommodates 11 descriptive variables.
  - **Two hidden layers consists of the first hidden dense layer with 11 neurons and the second hidden dense layer with 8 neurons.**
  - The output dense layer comprises a single neuron.

ii. **NetworkOneHidden**:
- A fully connected neural network with a depth of 2.
  - The input layer supports 11 descriptive variables.
  - **It contains one hidden dense layer with 11 neurons**
  - The output dense layer includes a single neuron

Both networks employee the following specifications:
- Rectified Linear Unit (ReLU) for all layers, except the output layer.
- The output layer uses either no activation function or a linear activation function.

3. Model Training

(e) Configure both networks for training using the same specification described in **Part III, 3. Model Training, (e)**

4. Model Comparison

(f) Create a plot to illustrate the **loss curves** for both neural network models, enabling visualization of the change in training loss values over epochs.

(g) Create a plot to display the **Mean Squared Error (MSE) curves** for both neural network models, facilitating comparison of the change in MSE values over the epochs.

5. Analysis

(h) Analyze and compare the performance of the two network models based on the plots generated in (f) and (g). Discuss your findings concerning network depth, validation loss, and any observed trends or differences in model learning.

Submit the following:

(1) Program code addressing tasks (a) through (g).

(2) Execution results for tasks (f) and (g)

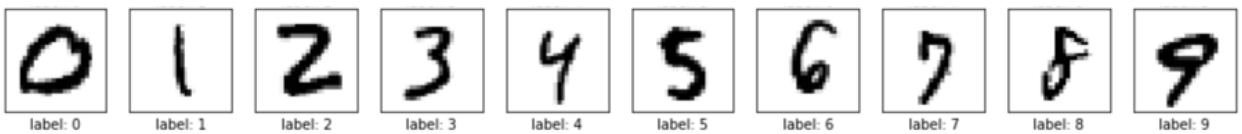
(3) Your response for task (h)

Part V. Application: Handwritten Digit Classification

You will be using a neural network for handwritten digit classification. Build the neural network classification models based on the descriptions below and compare their performance.

0. Data

- You will be using the MNIST data. The MNIST dataset includes the 10 digit labels, from 0 to 9. Each (grayscale) image is a 28 x 28 matrix of pixels, with values between 0 and 255. Each pixel is converted to a value in the interval [0, 1] by dividing by 255. An example of each digit from the MNIST dataset is shown below. The output labels are categorical values that denote the digits from 0 to 9.



- Download the MINST dataset from http://yann.lecun.com/exdb/mnist/. The dataset consists of four files: training set images, training set labels, test set images, and test set labels.

1. Data Preparation

(a) Load the **MINST dataset** into the training and testing data structures of your program**.** The dataset consists of 60,000 training images and 10,000 testing images.

(b) Since images are 2-dimensional matrices of 28×28 pixel values, you need to **flatten** them into a vector $\mathbf{x} \in \mathbb{R}^{784}$ with 784 values. This is done by simply concatenating all of the rows of the images to obtain one long vector.

(c) Each pixel in the images has an intensity value between 0 and 255. **Normalize** these intensity values to scale them to be **between 0 and 1**.

2. Model Design

(d) Design an artificial neural network with the following specifications:

- The input layer has 784 input variables.
- **One fully connected hidden layer** with 128 neurons. The processing neurons use Rectified Linear Units (**ReLUs**).
- The output dense layer has 10 neurons for the digits (0 to 9). The activation function is the ***softmax*** function for multiclass classification.

3. Model Training

(e) Configure the model for training as following:

- For this network, use **sparse_categorical_crossentropy** or any similar loss function for multiclass classification.
- For the optimization algorithm to change the weights of the network, use "**Adam**" or a first-order gradient-based optimization algorithm such as Gradient Descent.
- Set the **learning rate to 0.001.**
- Use a training **batch size of 128**.
- Train the model for **10 epochs**.
- Set aside **20% of the training data as validation data.**
- Specify **accuracy** as model performance metric during training

4. Model Evaluation

(f) Evaluate the trained model **on the separate test set**. Report the **precision, recall, F1 score for** each class label (0 – 9), and the overall test **accuracy**.

(g) Report the **confusion matrix**

5. Visualization

(h) Create a plot to display both **training and validation loss curves**

(i) Display **any 5 correctly classified images and 5 incorrectly classified images** with their actual labels and predicted labels.

6. Analysis

(j) After examining the loss curves of training and validation, describe any issues with the trained network and suggest alternative techniques to address these problem.

Submit the following:

(1) Program code addressing tasks (a) through (i).

(2) Execution results for tasks (f), (g), (h), and (i)

(3) Your response for task (j)

Part VI. Network Overfitting Handling

You will develop a neural network with the same configuration as given in Part V, but include a dropout option in the network architecture.

0. Data and Data Preprocess

(a) Load the same MNIST data. Perform the same data preprocessing steps.

1. Network Configuration and Model Training

(b) Design a network with the same architecture as given in Part V, **including a dropout layer with 50% deactivation rate**, and train the network.

2. Evaluation and Visualization

(c) Create a plot to display both **training and validation loss curves**

3. Analysis

(d) Compare the loss curve plot in Part V with this loss curve plot, describe your observations, and describe the effectiveness of the dropout layer.

Submit the following:

(1) Program code addressing tasks (a) through (c).

(2) Execution results for task (c)

(3) Your response for task (d)