The Image Cartoonifier SoC'23
Assignment 3

Dataset: Digit_Recognizer dataset

The Digit Recognizer dataset consists of images of the numbers 0-9. Each image is represented as a 28x28 matrix, with each pixel having a single pixel-value associated with it. 784 pixels in each 28x28 input image.

a) Split the data set into dev_set and train_set like X_dev, Y_dev for 1 to 1000 samples from the above data and X_train,Y_train from 1000 to m.
b) Extract the pixel values except the label values and Normalize the pixel values by dividing them by 255 to bring them into the range of 0 to 1. Store the normalized pixel values in variables called X_dev,X_train respectively.

Note: Do not use any Libraries like Tensorflow/pyTorch. You can use numpy,pandas and matplotlib.

Part 1: Neural Network Implementation:

i) Write a function named `init_params` that initializes the parameters for each layer of the neural network. The input layer (a[0]) should have 784 units, the second layer (a[1]) should have 120 units, the third layer (a[2]) should have 45 units, and the output layer (a[3]) should have 10 units. Initialize the weight matrices (W) with random values between 0 and 1, and the bias vectors (b) with random values between 0 and 1.

ii)Activation Functions:

a) Implement the **`ReLU`** activation function in a function called ReLU.
b) Implement the **`Softmax`** activation function in a function called Softmax.

iii)Forward Propagation:

a) Create a function named `forward_propagation` that takes the weights (W) and biases (b) as input and performs forward propagation through the neural network.

b) Implement the necessary calculations to compute the intermediate values (Z) and activations (A) for each layer.
c) Apply the `ReLU activation` function to the intermediate values for the hidden layers (a[1] and a[2]).
d) Apply the `Softmax activation` function to the intermediate values for the output layer (a[3]).
e) Return the intermediate values (Z) and activations (A) for each layer.

iv) One-Hot Encoding:

a) Implement a function called `one_hot` that converts an input array Y into its one-hot encoded representation.
b) Use numpy arrays to assign a **`binary vector`** to each element in Y, setting the index corresponding to the element's value to 1 and all others to 0.
c) Transpose the resulting array, where columns represent elements in Y and rows represent different classes.

Part 2: Backward Propagation and Model Training:

i)Backward Propagation:

a) Write a function named `backward_propagation` that performs backward propagation to calculate the gradients of the parameters.
b) Use the provided arguments Z1, A1, Z2, A2, Z3, A3, W1, W2, W3, X, and Y to calculate the gradients.
c) Apply the appropriate activation functions' derivatives in the backpropagation process.
d) Return the gradients dW1, db1, dW2, db2, dW3, and db3.

ii)Update Parameters:

a) Implement a function named `update_params` that updates the parameters of the neural network using gradient descent.
b) Use the provided arguments W1, b1, W2, b2, dW1, db1, dW2, db2, dW3, db3, and alpha (learning rate).
c) Update the parameters using the gradients and the learning rate.
d) Return the updated parameters W1, b1, W2, b2, W3, and b3.

iii)Get Prediction and Accuracy:

a) Create a function named `get_prediction` that takes the output activations (A3) as input and returns the predicted labels.
b) Use numpy's argmax function to find the index of the highest value in each column of A3.
c) Implement a function named get_accuracy that takes the predicted labels and true labels (Y) as input and calculates the accuracy.

d) Print the predicted labels and true labels in the required format

iv)Gradient Descent:

a) Write a function named `gradient_descent` that performs the training of the neural network using gradient descent.

b) Use the provided arguments X_train, Y_train, (alpha=0.1), and (num_iterations=1000).

c) Perform the iterations of gradient descent, updating the parameters and tracking the accuracy every 10th iteration.

d) Print the output layer prediction and accuracy during training.

Part 3: Model Evaluation:

i)Making Predictions:

a) Implement a function named `make_predictions` that takes the inputs (X) and trained parameters (W1, b1, W2, b2) as input.

b) Use the forward propagation function to obtain the predictions from the trained model.

c) Return the predictions.

ii)Testing Predictions:

a) Write a function named `test_prediction` that tests the model's predictions on a specific index of the training data.

b) Use the provided arguments index, W1, b1, W2, and b2.

c) Obtain the prediction and true label for the specified index.

d) Visualize the image data using Matplotlib and `print the prediction and true label.`

****** All The Best *****