Assignment 5 Report

1 ANN with backpropagation

1.1 Part 1

Give structure of ann

A basic feedforward and backpropagation artificial neural network was implemented with one input layer, one hidden layer, and one output layer. The input layer size is 784, corresponding to each pixel of a 28x28 image. The hidden layer size is a flexible hyperparameter, with the best size found to be 140. The output layer has 10 nodes, each representing a digit from 0 to 9. ReLU is used in the hidden layer for computational efficiency, non-linearity, mitigating vanishing gradients, and promoting sparse activation, while Softmax in the output layer converts raw scores into a probability distribution, ideal for multi-class classification and facilitating decision-making. The epoch number is set to 100. The learning rate, like the hidden layer size, was optimized by testing different values and selecting the best-performing one.

1.2 Part 2

Give the equations that you used to update the weights explaining all the parameters on them.

Weights are updated in the backpropagation part of the code.

- "*" used for multiplication
- "$\cdot$" used for the dot product
- "x^T" used for getting transpose of a matrix

Output Error Calculation

$$\text{output\_error} = \text{predicted\_output} - \text{targets}$$

- **predicted_output**: The outputs coming from forward propagation for each input sample.
- **targets**: The actual target values (one-hot encoded) for each input sample.

- **output_error**: The difference between predicted and true outputs, a matrix of shape (num samples, output size).

Output Error Derivative

$$\text{output\_error\_der} = \frac{\text{output\_error}}{\text{num\_samples}}$$

- **output_error**: The difference between the predicted output and the actual target values.
- **num_samples**: The number of samples in the input batch, used to normalize the error.
- **output_error_der**: The normalized output error

By normalizing the error, we make sure the gradient updates are properly scaled, stopping very large updates that could make the training process unstable.

Gradient of Weights from Hidden to Output Layer

$$\text{gradient\_weights\_hidden\_output} = \text{hidden\_layer\_output}^T \cdot \text{output\_error\_der}$$

- **hidden_layer_output**: The output from the hidden layer after applying the ReLU activation function.
- **output_error_der**: The normalized output error.
- **gradient_weights_hidden_output**: The gradient for weights between hidden and output layers.

Hidden Error Derivative

$$\text{hidden\_error\_der} = (\text{output\_error} \cdot \text{weights\_hidden\_output}^T) * \text{ReLU\_derivative}(\text{hidden\_layer\_input})$$

- **output_error**: The difference between the predicted output and the actual target values.
- **weights_hidden_output**: The weights connecting the hidden layer to the output layer.
- **ReLU_derivative(hidden_layer_input)**: The derivative of the ReLU activation function applied to the input of the hidden layer. This is calculated as 1 for positive values and 0 for non-positive values.
- **hidden_error_der**: The error derivative for the hidden layer

Gradient of Weights from Input to Hidden Layer

$$\text{gradient\_weights\_input\_hidden} = \frac{\text{inputs}^T \cdot \text{hidden\_error\_der}}{\text{num\_samples}}$$

- **inputs**: The input data batch.
- **hidden_error_der**: The derivative of the error with respect to the hidden layer.
- **gradient_weights_input_hidden**: The gradient for weights between input and hidden layers
- **num_samples**: The number of samples in the input batch, used to normalize the gradient.

Update Weights with Gradient Descent

$\text{weights\_input\_hidden} \leftarrow \text{weights\_input\_hidden} - \text{learning\_rate} * \text{gradient\_weights\_input\_hidden}$

$\text{weights\_hidden\_output} \leftarrow \text{weights\_hidden\_output} - \text{learning\_rate} * \text{gradient\_weights\_hidden\_output}$

- **weights_input_hidden**: The weights connecting the input layer to the hidden layer.
- **weights_hidden_output**: The weights connecting the hidden layer to the output layer.
- **learning_rate**: A hyperparameter that determines the step size for each iteration of gradient descent.
- **gradient_weights_input_hidden**: The gradient of the loss with respect to the weights between the input and hidden layers.
- **gradient_weights_hidden_output**: The gradient of the loss with respect to the weights between the hidden and output layers.

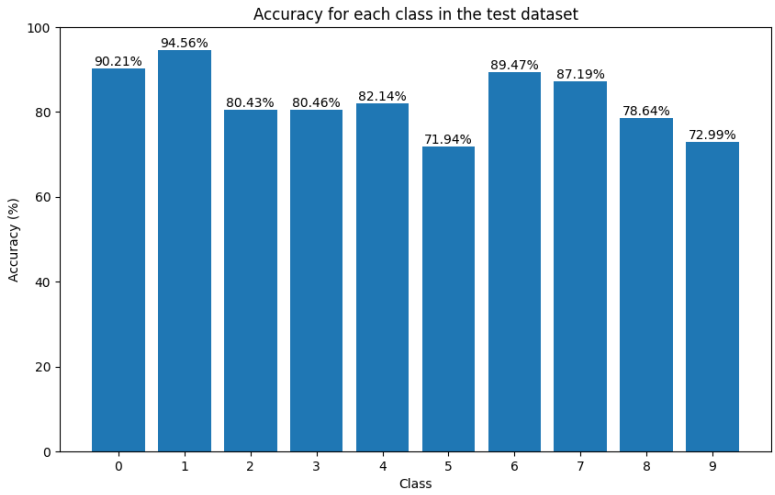

Figure 1: Class Accuracy

1.3 Part 3

Give the percentage of correctness of the total test data set (20% of all cases) and give the percentage of correctness of each of the classes in the test data set.

The hidden layer size was arranged to 140 and the learning rate to 0.22. The model was then trained for 100 epochs. After training, the model was tested with 20% of the dataset, achieving an accuracy of 83.10%. Figure 1 shows the percentage of correctness for each class in the test dataset.

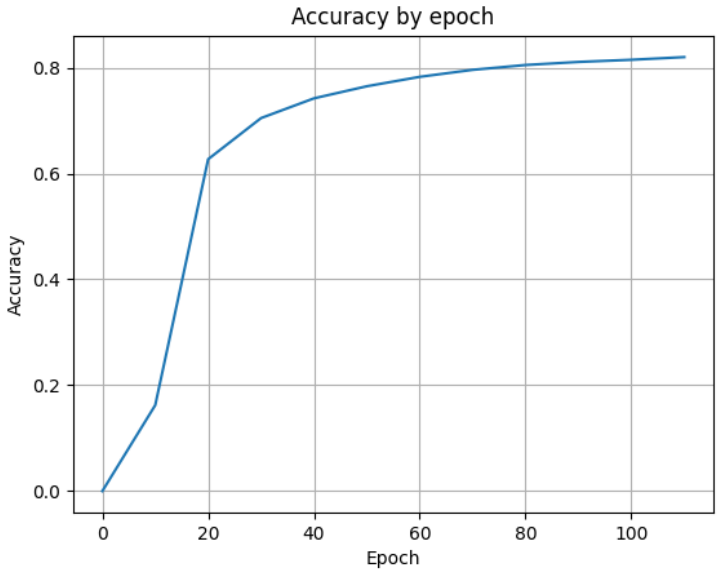

Figure 2: Accuracy by epoch

1.4 Part 4

Give a figure showing how the accuracy in the validation set is changing during the training process.

Figure 2 shows the accuracy in the validation set that changes during the training process.