MNIST classifier

Objection:

implement a classifier from scratch (using only torch tensor manipulations) for the MNIST dataset with more than 90% accuracy

---

The model I used for this task is fully connected neural network with two hidden layers.

<u>The weights used:</u>

$$W_1 \in R^{784\times128}, \quad b_1 \in R^{1\times128}$$

$$W_2 \in R^{128\times10}, \quad b_2 \in R^{1\times10}$$

The weights were first initialized to random normal variables and were normalized to prevent overflow in the following steps.

<u>The hyperparameters used:</u>

$$batch\ size = 100, \quad epochs = 10, \quad learning\ rate = 0.01$$
$$hidden\ layer\ size = 128$$

The hyperparameters were chosen by experiment so the running time and learning speed will be fast as possible.

<u>The prediction was done as follows:</u>

$$Z_1 = W_1X + b_1, \qquad A_1 = \tanh(Z_1)$$

$$Z_2 = W_2A_1 + b_2, \qquad A_2 = \mathrm{softmax}(Z_2) = \hat{Y}$$

<u>The loss was measured by:</u>

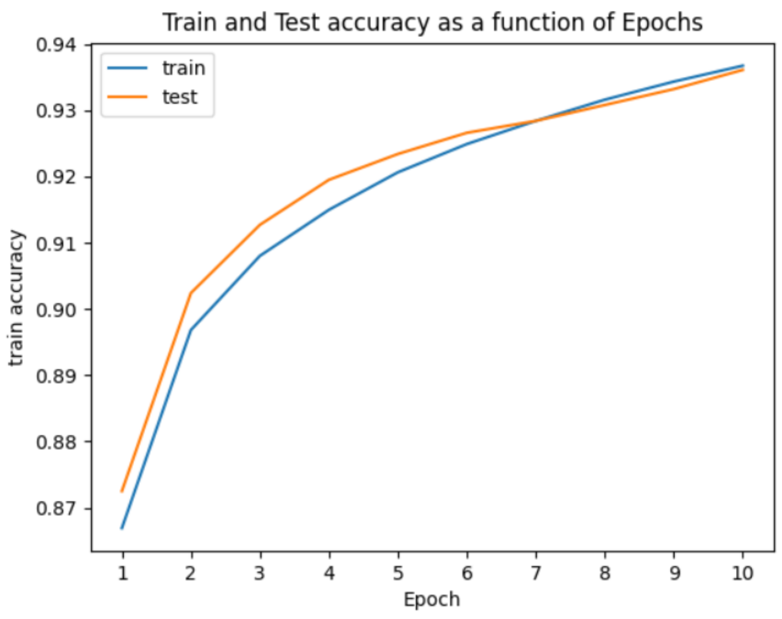
$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}\left(\left\|Y_i - \hat{Y}_i\right\|\right)^2$$

<u>The learning process was performed as follows:</u>

in each epoch, the training set was divided into batches containing 100 training samples each. For each batch, the network performs a forward pass in order to get a probability prediction for each label. Afterward, the network performs a backward pass by calculating the derivatives and updating the weights accordingly.

At the end of each epoch, the accuracy of the network on both the training set and test set is calculated.
(The process is shown in the graph below)

Finally, at the end of the training process, a forward pass was performed on the training set with the final weights. The final accuracy achieved was 93% .



The attached files:

- $train.py$ – train the model and save the final weights to pkl file
- $eval.py$ – load the final weights and test the model on the test-set
- $weights.zip$ – contains $weights.pkl$ with final weights