Machine Learning & Discovery *ECS 271, Fall 2022*

HW2

Single-layer neural network:

1. Implement a single-layer fully connected neural network model to classify MNIST images. It inputs a raw image as a 1D vector of length 784=28x28 and outputs a vector of length 10 (each dimension corresponds to a class). You may want to add a bias term to the input, but that is optional for this assignment. The output is connected to the first layer simply by a set of linear weights. The second layer uses Softmax function as the non-linearity function. Softmax is a simple function that converts a n-dimensional input (z) to a n-dimensional output (o) where the output sums to one and each element is a value between 0 and 1. It is defined as

$$y_i = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)}$$

When we apply this function to the output of the network, o, it predicts a vector which can be seen as the probability of each category given the input x:

$$P(c_i|x) = \frac{exp(o_i)}{\sum_{j=1}^{n} exp(o_j)}$$

where n is the number of categories, 10, in our case. We want the i'th output to mimic $P(c_i|x)$, the probability of the input x belonging to the category i. We can represent the desired probability distribution as the vector gt where $gt(i)$ is one only if the input is from the i'th category and zero otherwise. This is called one-hot encoding. Assuming x is from y'th category, $gt(y)$ is the only element in gt that is equal to one. Then, we want the output probability distribution to be similar to the desired one (ground-truth). Hence, we use cross-entropy loss to compare these two probability distributions, P and gt:

$$L(x, y, w) = \sum_{i=1}^{n} -gt(i)log(P(c_i|x))$$

where n is the number of categories. Since gt is one hot encoding, we can remove the terms of gt that are zero, keeping only the y'th term. Since $gt(y) = 1$, we can remove it in the multiplication to come up with the following loss which is identical to the above one:

$$L(x, y, w) = -log(P(c_y|x))$$

This is the loss for one input only, so the total loss on a mini-batch is:

$$L = \sum_{k=1}^{N} -log(P(c_{y_k}|x_k))$$

where N is the size of mini-batch, number of training data being processed at this iteration.

Please implement stochastic gradient descend (SGD) algorithm from scratch to train the model. You may use NumPy, but should not use PyTorch, TensorFlow, or any similar deep learning framework. Use mini-batch of 10 images per iteration. Then, train it on all MNIST train dataset and plot the accuracy on all test data for every n iteration. Please choose n small enough so that the graph shows the progress of learning and large enough so that testing does not take a lot of time. You may use smaller n initially and then increase it gradually as the learning progresses. Choose a learning rate so that the loss goes down.

2. For each class, visualize the 10 images that are misclassified with the highest score along with their predicted label and score. These are very confident wrong predictions.

3. Please reduce the number of training data to 1 example per class (chosen randomly from training data) and plot the curve (accuracy vs, iterations). The whole training data will be 10 images only.

4. Try different mini-batch sizes (1, 10, 100) for the original case and plot the results. Which one is better and why?

5. Instead of using random sampling, sort the data before training so that all "1"s appear before "2"s and so on. Then, sample sequentially in running SGD instead of random sampling. Does this work well, why?

6. (Bonus point) Add a hidden layer with 11 hidden neurons and ReLU activation function. Then, plot the accuracy curve to see how the accuracy changes.