EE 443/543 Neural Networks, Spring 2025 - Project #5
Due: 03/25/2025, 11pm.

Submit codes and reports as usual.
You should use your own code without any help from any external neural network/machine learning algorithms. That is, write your own code to calculate the derivatives.

As a preparation for the problem, we review the formulation of a general classification problem. Suppose that we have a set $\mathcal{C}$ of classes. For example, $\mathcal{C} = \{0, 1, \ldots, 9\}$ for the digit classification problem, or we could have $\mathcal{C} = \{\text{cat}, \text{dog}, \text{dragon}\}$ for a problem where we wish to determine whether a given picture contains a cat, dog, or a dragon. In the supervised learning setup, we have a training set $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with the corresponding predetermined classes/labels $c_1, \ldots, c_n \in \mathcal{C}$. We wish to design a neural network that provides the hopefully-correct class of any given input $\mathbf{x}$.

Suppose that you consider a network with m output neurons to solve the classification problem described above. The first thing to do is to assign, for each class $i \in \mathcal{C}$, a representative output vector $\mathbf{d}_i \in \mathbb{R}^m$. Let $\mathbf{D} = \{\mathbf{d}_i : i \in \mathcal{C}\}$ denote the set of all representative output vectors of classes. For example, for the digit classification problem in Homework 2, we considered $m = 10$ output neurons with representative output vectors $\mathbf{d}_0 = [1\,0\,0\cdots 0]^T$, $\mathbf{d}_1 = [0\,1\,0\cdots 0]^T$, and so on. In particular, for the training sample $\mathbf{x}_i$, the class label is c_i, so that the desired output vector would be $\mathbf{d}_{c_i}$.

The next thing to do is to consider an energy function of the form $\frac{1}{n}\sum_{i=1}^{n} D(\mathbf{d}_{c_i}, f(\mathbf{x}_i, \mathbf{w}))$. Here,

- i is the training sample index,
- n is the number of training samples,
- $\mathbf{d}_{c_i} \in \mathbf{C}$ is the desired output for training sample i,
- $f(\mathbf{x}_i, \mathbf{w})$ is the network output given input (training sample) $\mathbf{x}_i$ and weights $\mathbf{w}$,
- $D(\cdot, \cdot)$ is some arbitrary distance function (metric). In particular, we use the function $D(\mathbf{d}_{c_i}, f(\mathbf{x}_i, \mathbf{w}))$ to measure how far off the network output $f(\mathbf{x}_i, \mathbf{w})$ is from the desired output $\mathbf{d}_{c_i}$. Typically, we choose $D(\mathbf{z}_0, \mathbf{z}_1) = \|\mathbf{z}_0 - \mathbf{z}_1\|^2$.

The next thing to do (which is the harder part) is to use the backpropagation algorithm to find some optimal weights, say $\mathbf{w}_0$, that minimize $\frac{1}{n}\sum_{i=1}^{n} D(\mathbf{d}_{c_i}, f(\mathbf{x}_i, \mathbf{w}))$. Having done this, the question is now how to determine the class of some arbitrary input pattern $\mathbf{x}$? Intuitively, we should choose the class whose representative output vector is closest to the output provided by $\mathbf{x}$ according to distance function $D(\cdot, \cdot)$. In other words, given some arbitrary input pattern $\mathbf{x}$ and weights $\mathbf{w}_0$ (or any weights in general), we first calculate the network output $f(\mathbf{x}, \mathbf{w}_0)$. We can then estimate the class of $\mathbf{x}$ as

$$\arg\min_{i \in \mathcal{C}} D(\mathbf{d}_i, f(\mathbf{x}, \mathbf{w}_0)).$$

In other words, the estimated class i for input pattern $\mathbf{x}$ should minimize $D(\mathbf{d}_i, f(\mathbf{x}, \mathbf{w}_0))$.

1. **Q1 (100 pts)** In this computer project, we will design a neural network for digit classification using the backpropagation algorithm (see the notes above). You should use the MNIST data set (https://github.com/mkolod/MNIST) that consists of 60000 training images and 10000 test images. **The training set should only be used for training, and the test set should only be used for testing.** You can use an existing library to parse the data. Your final report should include the following:
   - The details of your architecture/design, including
     - Your network topology, i.e. how many layers, how many neurons in each layer, etc. (Obviously you will have 784 neurons in the input layer).

  - The way you represented digits $0, \ldots, 9$ in the output layer. For example, one may use the same setup as in Homework 2, 10 output neurons, with $[1\ 0\ \cdots 0]$ representing a 0, $[0\ 1\ 0 \cdots 0]$ representing a 1 and so on. Another person may have just one output neuron with an output of 0 representing a 0, an output of 1 representing a 1, and so on.
  - Neuron activation functions, learning rates for each neuron, and any dynamic update of learning rates (as explained in the question above) if it exists.
  - The energy/distance functions of your choice.
  - Any other tricks such as regularization, dropout, momentum method, etc.
- The reasons as to why you chose a particular hyperparameter the way it is (e.g. why did you choose 100 hidden neurons, but not 50, why do you have 1 hidden layer but not 2?)
- Your design process. It is unlikely that the first network you train will actually work. Write about your design process, your failures, together with your comments on why do you think a particular approach failed (e.g. I began with $\eta = 100$ and the algorithm just diverged, η was just too large).
- A pseudocode of your final algorithm as described in (f) of Question 1 above.
- A plot that shows epoch number vs the number of classification errors on both training and test images. Another plot that shows the epoch number vs the energy on both training and test images. Your test set should include all 10000 test images.

Your final network should achieve a decent success rate. You should be able to achieve at least 95% success rate on the test set (all 10000 images). Very poor network performance will also result in a lower grade. It is very likely that the success rate will be much higher.