Assigned: 09 October

Homework #4

EE 541: Fall 2023

Due: Friday, 20 October at 23:59. Submission instructions will follow separately on canvas.

1. The MNIST dataset of handwritten digits is one of the earliest and most used datasets to benchmark machine learning classifiers. Each datapoint contains 784 input features – the pixel values from a 28×28 image – and belongs to one of 10 output classes – represented by the numbers 0-9.

   In this problem you will use numpy to program the feed-forward phase for a deep multilayer perceptron (MLP) neural network. We provide you with a pre-trained MLP using the MNIST dataset. The MLP has an input layer with 784-neurons, 2 hidden layers of 200 and 100 neurons, and a 10-neuron output layer. The model assumes ReLU activation for the hidden layers and softmax function in the output layer.

   (a) Extract the weights and biases of the pre-trained network from `mnist_network_params.hdf5`. The file has 6 keys corresponding to: `W_1`, `b_1`, `W_2`, `b_2`, `W_3`, `b_3`. Verify the dimension of each numpy array with the shape property.

   (b) The file `mnist_testdata.hdf5` contains 10,000 images. `xdata` holds pixel intensities and `ydata` contains the corresponding class labels. Extract these. Note: each image vector is 784-dimensions and the label is 10-dimensional with one-hot encoded, *i.e.,* label `[0,0,0,1,0,0,0,0,0,0]` means the image is class "3".

   (c) Write functions to calculate ReLU and softmax:

   - $\mathrm{ReLU}(x) = \max(0, x)$.
   - $\mathrm{Softmax}(x) = \left[\frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}}, \frac{e^{x_2}}{\sum_{i=1}^{n} e^{x_i}}, \ldots, \frac{e^{x_n}}{\sum_{i=1}^{n} e^{x_i}}\right]$.

     The softmax function takes a vector of size n and returns another vector of size n that you can interpret as a probability distribution. For example: `Softmax([0; 1; 2]) = [0:09; 0:24; 0:67]` so you can conclude that the 3rd element is the most likely outcome.

   (d) Use numpy to create an MLP to classify 784-dimensional images into the target 10-dimensional output. Use ReLU activation for the two hidden layers and softmax the output layer. Format and write your results to file `result.json` according to:

```
data = [
  {"index": XXX, "activations": [YYY,..., YYY], "classification": ZZZ},
  ...
]

import json
with open("result.json", "w") as f:
  f.write(json.dumps(data))
```

Output `index` and `activation` as integers (not string) and include only the 10 final-layer output activations as numeric floats.

(e) Compare your prediction with the (true) `ydata` label. Count the classification as correct if the position of the maximum element in your prediction matches with the position of the 1 in `ydata`. Tally the number of correctly classified images from the whole set of 10,000. [hint: 9790 correct].

(f) Identify and investigate several datapoints that your MLP classified correctly and several it classified incorrectly. Inspect them visually. Is the correct class obvious to you in the incorrect cases? Use `matplotlib` to visualize:

```
import matplotlib.pyplot as plt
plt.imshow(xdata[i].reshape(28,28))
plt.show()
# the index i selects which image to visualize
# xdata[i] is a 784x1 numpy array
```

2. **Backprop by Hand**

Consider an MLP with three input nodes, two hidden layers, and three outputs. The hidden layers use the ReLU activation function and the output layer users softmax. The weights and biases for this MLP are:

$$\mathbf{W}^{(1)} = \begin{bmatrix} 1 & -2 & 1 \\ 3 & 4 & -2 \end{bmatrix}, \qquad \mathbf{b}^{(1)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$
$$\mathbf{W}^{(2)} = \begin{bmatrix} 1 & -2 \\ 3 & 4 \end{bmatrix}, \qquad \mathbf{b}^{(2)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$\mathbf{W}^{(3)} = \begin{bmatrix} 2 & 2 \\ 3 & -3 \\ 2 & 1 \end{bmatrix}, \qquad \mathbf{b}^{(3)} = \begin{bmatrix} 0 \\ -4 \\ -2 \end{bmatrix}$$

(a) **Feedforward Computation:** Perform the feedforward calculation for the input vector $\mathbf{x} = [\ +1\ \ -1\ \ +1\ ]^{\mathrm{T}}$. Fill in the following table. Follow the notation used in the slides, *i.e.,* $\mathbf{s}^{(l)}$ is the linear activation, $\mathbf{a}^{(l)} = \underline{h}(\mathbf{s}^{(l)})$, and $\dot{\mathbf{a}}^{(l)} = \underline{\dot{h}}(\mathbf{s}^{(l)})$.

l:	1	2	3
$\mathbf{s}^{(l)}$:			
$\mathbf{a}^{(l)}$:			
$\dot{\mathbf{a}}^{(l)}$:			(not needed)

(b) **Backpropagation Computation:** Apply standard SGD backpropagation for the input assuming a multi-category cross-entropy loss function and one-hot labeled target: $\mathbf{y} = [\ 0\ \ 0\ \ 1\ ]^{\mathrm{T}}$. Follow the notation used in the slides, *i.e.,* $\delta^{(l)} = \nabla_{\mathbf{s}^{(l)}} C$. Enter the delta values in the table below and provide the updated weights and biases assuming a learning rate $\eta = 0.5$.

l:	1	2	3
$\delta^{(l)}$:			
$\mathbf{W}^{(l)}$:			
$\mathbf{b}^{(l)}$:			