Assigned: 07 October

Homework #4

EE 541: Fall 2024

Due: Sunday, 13 October at 22:00. Submission instructions will follow separately on brightspace.

1. **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
$\delta^{(l)}$:			
$\mathbf{W}^{(l)}$:			
$\mathbf{b}^{(l)}$:			

2. Logistic regression

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 classify input images using a logistic-regression. Use only Python standard library modules, `numpy`, and `mathplotlib` for this problem.

(a) Logistic "2" detector

In this part you will use the provided MNIST handwritten-digit data to build and train a logistic "2" detector:

$$y = \begin{cases} 1 & \mathbf{x} \text{ is a "2"} \\ 0 & \text{else.} \end{cases}$$

A logistic classifier takes learned weight vector $\mathbf{w} = [w_1, w_2, \ldots w_L]^T$ and the unregularized offset bias $b \triangleq w_0$ to estimate a probability that an input vector $\mathbf{x} = [x_1, x_2, \ldots, x_L]^T$ is "2":

$$p(\mathbf{x}) = P[Y = 1|\mathbf{x}, \mathbf{w}] = \frac{1}{1 + \exp\left(-\left(\sum_{i=1}^{L} w_k \cdot x_k + w_0\right)\right)} = \frac{1}{1 + \exp\left(-\left(w^T x + w_0\right)\right)}.$$

Train a logistic classifier to find weights that minimize the binary log-loss (also called the binary cross entropy loss):

$$l(w) = -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \log p(x)\right) + \left(1 - y_i\right) \log\left(1 - p(x)\right)$$

where the sum is over the N samples in the training set. Train your model until convergence according to some metric you choose. Experiment with variations of ℓ_1- and/or ℓ_2-regularization to stabilize training and improve generalization.

Submit answers to the following.

i. How did you determine a learning rate? What values did you try? What was your final value?

ii. Describe the method you used to establish model convergence.

iii. What regularizers did you try? Specifically, how did each impact your model or improve its performance?

iv. Plot log-loss (*i.e.,* learning curve) of the training set and test set on the same figure. On a separate figure plot the accuracy against iteration number of your model on the training set and test set. Plot each as a function of the iteration number.

v. Clasify each input to the binary output "`digit is a 2`" using a 0.5 threshold. Compute the final loss and final accuracy for both your training set and test set.

Submit your trained weights to Autolab. Save your weights and bias to an hdf5 file. Use keys `w` and `b` for the weights and bias, respectively. `w` should be a length-784 numpy vector/array and `b` should be a numpy scalar. Use the following as guidance:

```
with h5py.File(outFile, 'w') as hf:
hf.create_dataset('w', data = np.asarray(weights))
hf.create_dataset('b', data = np.asarray(bias))
```

Note: you will **not** be scored on your models overall accuracy. But a low-score may indicate errors in training or poor optimization.