Homework 3 – Machine Learning (CS4342, Whitehill, Spring 2020)

You may complete this homework assignment either individually or in teams up to 2 people.

1. **Softmax regression (aka multinomial logistic regression)** [35 points]: In this problem you will train a softmax regressor to classify images of hand-written digits from the MNIST dataset. The input to the machine will be a 28×28-pixel image (converted into a 784-dimensional vector); the output will be a vector of 10 probabilities (one for each digit). Specifically, the machine you create should implement a function $g : \mathbb{R}^{785} \rightarrow \mathbb{R}^{10}$, where the kth component of $g(\tilde{\mathbf{x}})$ (i.e., the probability that input $\tilde{\mathbf{x}}$ belongs to class k) is given by

$$\frac{\exp(\tilde{\mathbf{x}}^\top \tilde{\mathbf{w}}_k)}{\sum_{k'=0}^{9} \exp(\tilde{\mathbf{x}}^\top \tilde{\mathbf{w}}_{k'})}$$

where $\tilde{\mathbf{x}} = [\mathbf{x}^\top, 1]^\top$.

The weights should be trained to minimize the cross-entropy (CE) loss:

$$f_{\text{CE}}(\tilde{\mathbf{w}}_0, \ldots, \tilde{\mathbf{w}}_9) = -\frac{1}{n} \sum_{j=1}^{n} \sum_{k=0}^{9} y_k^{(j)} \log \hat{y}_k^{(j)}$$

where n is the number of training examples. Note that each $\hat{y}_k$ implicitly depends on all the weights $\tilde{\mathbf{w}}_0, \ldots, \tilde{\mathbf{w}}_9$, where each $\tilde{\mathbf{w}}_k = [\mathbf{w}_k^\top, 1]^\top$.

To get started, first download the MNIST dataset (including both the training and testing subsets) from the following web links:

- `https://s3.amazonaws.com/jrwprojects/small_mnist_train_images.npy`
- `https://s3.amazonaws.com/jrwprojects/small_mnist_train_labels.npy`
- `https://s3.amazonaws.com/jrwprojects/small_mnist_test_images.npy`
- `https://s3.amazonaws.com/jrwprojects/small_mnist_test_labels.npy`

These files can be loaded into `numpy` using `np.load`.

Then implement **stochastic gradient descent** (SGD) as described in the lecture notes. I recommend setting $\tilde{n} = 100$ for this project.

Note that, since there are 785 inputs (including the constant 1 term) and 10 outputs, there will be 10 separate weight vectors, each with 785 components. Alternatively, you can conceptualize the weights as a 10×785 matrix.

After optimizing the weights on the training set, compute both (1) the loss and (2) the accuracy (percent correctly classified images) on the **test** set. **Include both the cross-entropy loss values and the "percent-correct" accuracy in the screenshot that you submit.**

Finally, create an image to visualize each of the trained weight vectors $\mathbf{w}_0, \ldots, \mathbf{w}_9$ (similar to what you did in homework 2).

2. **Data augmentation** [25 points]: It is often useful to enlarge your training set by synthesizing new examples from ones you already have. The simplest way to do this is to apply **label-preserving transformations**, i.e., create new "copies" of some original training examples by altering them in subtle ways such that the label of the copy is always the same as the original. For images, this can be achieved through operations such as rotation, scaling, translating, as well as adding random noise to the value of each image pixel (e.g., from a Gaussian or Laplacian distribution). (For *symmetric* classes (e.g., 8, 0), you could use mirroring/flipping, though this is not required for this assignment.)

You are required to implement all of the following transformations: translation, rotation, scaling, random noise. (For rotation, feel free to use the `skimage.transform.rotate` method in the `skimage`

package.) From *each* example i of the original training set $(\mathbf{x}^{(i)}, y^{(i)})$, randomly pick one of the augmentation methods above, and generate a new image whose label is also $y^{(i)}$. Put all n of these new examples into a new Python array called `Xaug` and its associated label into a new array `yaug`. We will manually inspect your code to verify that you completed this correctly. Then, show 1 example (in the PDF file) of an original and augmented training example for *each* of these transformations (i.e., 4 original and 4 augmented images in total).

Note: augmenting the training set in this assignment will likely *not* help much to improve generalization accuracy because of how the softmax regression classifier works (it is a generalized linear model). However, it can make a substantial improvement for other models we will explore later in this course, e.g., non-linear support vector machines and neural networks.

In addition to submitting your Python code in a file called `homework3_WPIUSERNAME1.py` (or `homework3_WPIUSERNAME1_WPIUSERNAME2.py` for teams), please submit a PDF file containing a screenshot of (1) the last 20 iterations of your gradient descent on the training data. Name the file `homework3_WPIUSERNAME1.pdf` (or `homework3_WPIUSERNAME1_WPIUSERNAME2.pdf` for teams).