

Homework 3 – Deep Learning (CS/DS 541, Whitehill, Spring 2020)

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

1. **Newton's method** [10 points]: Show that, for a 2-layer linear neural network (i.e., $\hat{y} = f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$) and the cost function

$$J(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)})^2$$

Newton's method (see Equation 4.12 in *Deep Learning*) will converge to the optimal solution $\mathbf{w}^* = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}$ in 1 iteration no matter what the starting point $\mathbf{w}^{(0)}$ of the search is.

2. **Derivation of softmax regression gradient updates** [25 points]: As explained in class, let

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}^{(1)} & \dots & \mathbf{w}^{(c)} \end{bmatrix}$$

be an $m \times c$ matrix containing the weight vectors from the c different classes. The output of the softmax regression neural network is a vector with c dimensions such that:

$$\begin{aligned} \hat{y}_k &= \frac{\exp z_k}{\sum_{k'=1}^c \exp z_{k'}} \\ z_k &= \mathbf{x}^\top \mathbf{w}^{(k)} + b_k \end{aligned} \tag{1}$$

for each $k = 1, \dots, c$. Correspondingly, our cost function will sum over all c classes:

$$f_{\text{CE}}(\mathbf{W}, \mathbf{b}) = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^c y_k^{(i)} \log \hat{y}_k^{(i)}$$

Important note: When deriving the gradient expression for each weight vector $\mathbf{w}^{(l)}$, it is crucial to keep in mind that the weight vector for each class $l \in \{1, \dots, c\}$ affects the outputs of the network for *every* class, *not* just for class l . This is due to the normalization in Equation 1 – if changing the weight vector *increases* the value of \hat{y}_l , then it necessarily must *decrease* the values of the other $\hat{y}_{l' \neq l}$.

In this homework problem, please complete the following derivation that is outlined below:

Derivation: For each weight vector $\mathbf{w}^{(l)}$, we can derive the gradient expression as:

$$\begin{aligned} \nabla_{\mathbf{w}^{(l)}} f_{\text{CE}}(\mathbf{W}, \mathbf{b}) &= -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^c y_k^{(i)} \nabla_{\mathbf{w}^{(l)}} \log \hat{y}_k^{(i)} \\ &= -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^c y_k^{(i)} \left(\frac{\nabla_{\mathbf{w}^{(l)}} \hat{y}_k^{(i)}}{\hat{y}_k^{(i)}} \right) \end{aligned}$$

We handle the two cases $l = k$ and $l \neq k$ separately. For $l = k$:

$$\begin{aligned} \nabla_{\mathbf{w}^{(l)}} \hat{y}_k^{(i)} &= \text{complete me...} \\ &= \mathbf{x}^{(i)} \hat{y}_l^{(i)} (1 - \hat{y}_l^{(i)}) \end{aligned}$$

For $l \neq k$:

$$\begin{aligned} \nabla_{\mathbf{w}^{(l)}} \hat{y}_k^{(i)} &= \text{complete me...} \\ &= -\mathbf{x}^{(i)} \hat{y}_k^{(i)} \hat{y}_l^{(i)} \end{aligned}$$

To compute the total gradient of f_{CE} w.r.t. each $\mathbf{w}^{(k)}$, we have to sum over all examples *and* over $l = 1, \dots, c$. (**Hint:** $\sum_k a_k = a_l + \sum_{k \neq l} a_k$.)

$$\begin{aligned}\nabla_{\mathbf{w}^{(l)}} f_{\text{CE}}(\mathbf{W}, \mathbf{b}) &= -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^c y_k^{(i)} \nabla_{\mathbf{w}^{(l)}} \log \hat{y}_k^{(i)} \\ &= \text{complete me...} \\ &= -\frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)} \left(y_l^{(i)} - \hat{y}_l^{(i)} \right)\end{aligned}$$

Finally, show that

$$\nabla_{\mathbf{b}} f_{\text{CE}}(\mathbf{W}, \mathbf{b}) = -\frac{1}{n} \sum_{i=1}^n \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)} \right)$$

3. **Implementation of softmax regression** [20 points]: Train a 2-layer softmax neural network to classify images of hand-written digits from the MNIST dataset. The input to the network 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). The cross-entropy loss function that you minimize should be

$$f_{\text{CE}}(\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(10)}) = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^{10} y_k^{(i)} \log \hat{y}_k^{(i)} + \frac{\alpha}{2} \sum_{k=1}^c \mathbf{w}^{(k)\top} \mathbf{w}^{(k)}$$

where n is the number of examples and α is a regularization constant.. Note that each \hat{y}_k implicitly depends on all the weights $\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(10)}$.

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

- https://s3.amazonaws.com/jrwprojects/mnist_train_images.npy
- https://s3.amazonaws.com/jrwprojects/mnist_train_labels.npy
- https://s3.amazonaws.com/jrwprojects/mnist_validation_images.npy
- https://s3.amazonaws.com/jrwprojects/mnist_validation_labels.npy
- https://s3.amazonaws.com/jrwprojects/mnist_test_images.npy
- https://s3.amazonaws.com/jrwprojects/mnist_test_labels.npy

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

Then implement stochastic gradient descent (SGD) to minimize the cross-entropy loss function. Regularize the weights but *not* the bias \mathbf{b} . Optimize the same hyperparameters as in homework 2 problem 2 (age regression). You should also use the same methodology as for the previous homework, except that the MNIST dataset includes a dedicated validation set that you should use.

Performance evaluation: Once you have tuned the hyperparameters and optimized the weights so as to maximize performance on the validation set, then: (1) **stop** training the network and (2) evaluate the network on the **test** set. Record the performance both in terms of (unregularized) cross-entropy loss (smaller is better) and percent correctly classified examples (larger is better).

Put your code in a Python file called `homework3.WPIUSERNAME1.py` (or `homework3.WPIUSERNAME1.WPIUSERNAME3.py` for teams). For the proof and derivation, please create a PDF called `homework3.WPIUSERNAME1.pdf` (or `homework3.WPIUSERNAME1.WPIUSERNAME3.pdf` for teams). Create a Zip file containing both your Python and PDF files, and then submit on Canvas.