Homework 3 – Deep Learning (CS/DS 541, Murai, Fall 2022)

(With permission from Prof. Whitehill)

You may complete this homework assignment in teams of 2-3 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** [20 points]: As explained in class, let
$$\mathbf{W} = \left[\begin{array}{ccc} \mathbf{w}^{(1)} & \dots & \mathbf{w}^{(c)} \end{array}\right]$$
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 $l = 1, \ldots, c$. (**Hint**: $\sum_k a_k = a_l + \sum_{k \neq l} a_k$. Also, $\sum_k y_k = 1$.)

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

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. **Derivation of Cross-Entropy as Negative Log-Likelihood** [10 points]:

A softmax regression network estimates the probability that the input $\mathbf{x}$ belongs to class k for each $k = 1, \ldots, c$. In particular,

$$\hat{y}_k \doteq P(y_k = 1 \mid \mathbf{x}, \mathbf{W}, \mathbf{b}) \quad \forall k \in \{1, \ldots, c\}$$

During training, we know for each training example $\mathbf{x}$ its ground-truth label $\mathbf{y} = [y_1, \ldots, y_c]^\top$, where $\mathbf{y}$ is a one-hot vector. Suppose the index of the "1" in $\mathbf{y}$ is k (i.e., the ground-truth class of the example is k). Then the *likelihood* of the training example, given fixed $\mathbf{W}, \mathbf{b}$, is $P(y_k = 1 \mid \mathbf{x}, \mathbf{W}, \mathbf{b})$ which, as stated above, is simply $\hat{y}_k$. More generally, we can represent the likelihood of the training example as

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{W}, \mathbf{b}) = P(y_1 = 1 \mid \mathbf{x}, \mathbf{W}, \mathbf{b})^{y_1} \times \ldots \times P(y_c = 1 \mid \mathbf{x}, \mathbf{W}, \mathbf{b})^{y_c}$$

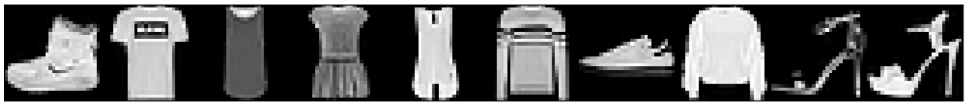
The reason is that exactly one y_k will be 1, and all the other $y_{k' \neq k}$ will be 0; hence, only one factor in the equation above will equal something other than 1. The exponents in the equation above are handy to "pick out" the one term that is relevant. This representation is sometimes called the *1-of-c* notation. Using the definition of $\hat{y}_k$, we can simplify the likelihood to:

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{W}, \mathbf{b}) = \prod_{k=1}^{c} \hat{y}_k^{y_k}$$

Your task: Derive the cross-entropy loss (see slide #74 of Class5.pdf) of a dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$, under a softmax regression network with fixed weights $\mathbf{W}$ and bias $\mathbf{b}$, as the *negative log-likelihood*, i.e., $-\log P(\mathcal{D} \mid \mathbf{W}, \mathbf{b})$. Like in the linear-Gaussian problem from homework 2, you can make use of conditional independence, i.e., given $\mathbf{W}, \mathbf{b}$, the likelihood of $\mathcal{D}$ factorizes into the products of the likelihoods of all the individual training examples.

$$\begin{aligned}
-\log P(\mathcal{D} \mid \mathbf{W}, \mathbf{b}) &= \ldots && (2) \\
&\ldots \text{complete me} \ldots && (3) \\
&= -\sum_{i=1}^{n} \sum_{k=1}^{c} y_k^{(i)} \log \hat{y}_k^{(i)} && (4) \\
&= f_{\text{CE}}(\mathcal{D}; \mathbf{W}, \mathbf{b}) && (5)
\end{aligned}$$

4. **Implementation of softmax regression** [20 points]:



Train a 2-layer softmax neural network to classify images of fashion items (10 different classes, such as shoes, t-shirts, dresses, etc.) from the Fashion 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 class). The cross-entropy loss function[1] that you minimize should be

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

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 $\mathbf{W} = \left[\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(10)}\right]$ and biases $\mathbf{b} = \left[b^{(1)}, \ldots, b^{(10)}\right]$.

To get started, first download the Fashion MNIST dataset from the following web links:

- `https://s3.amazonaws.com/jrwprojects/fashion_mnist_train_images.npy`
- `https://s3.amazonaws.com/jrwprojects/fashion_mnist_train_labels.npy`
- `https://s3.amazonaws.com/jrwprojects/fashion_mnist_test_images.npy`
- `https://s3.amazonaws.com/jrwprojects/fashion_mnist_test_labels.npy`

These files can be loaded into `numpy` using `np.load`. Each "labels" file consists of a 1-d array containing n labels (valued 0-9), and each "images" file contains a 2-d array of size $n \times 784$, where n is the number of images.

Next, implement stochastic gradient descent (SGD) to minimize the cross-entropy loss function on this dataset. Regularize the weights but *not* the biases. Optimize the same hyperparameters as in homework 2 problem 2 (age regression). You should also use the same methodology as for the previous homework, including the splitting of the training files into validation and training portions.

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 this information into the PDF you submit.

Put your code in a Python file called `homework3_WPIUSERNAME1.py` (or `homework3_WPIUSERNAME1_WPIUSERNAME2.py` for teams). For the proof and derivation, as well as the cross-entropy values from the Fashion MNIST problem, please create a PDF called `homework3_WPIUSERNAME1.pdf` (or `homework3_WPIUSERNAME1_WPIUSERNAME2.pdf` for teams). Create a Zip file containing both your Python and PDF files, and then submit on Canvas.

[1] In this equation, the regularization term is not divided by n like in the lecture notes. Either equation is valid since the $1/n$ can be subsumed into α. Here, for simplicity, the $1/n$ is omitted.