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 $l = 1, \ldots, c$. (**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

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

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 $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(10)}$.

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.

learning rate = [0.001, 0.005, 0.01, 0.05]

epochs = [50, 100, 200, 400]

Batch size = [0.1, 0.2, 0.3, 0.5]

alpha = [0.001, 0.002, 0.005, 0.01]

} Tuned hyperparameters.

I could get highest accuracy of only 17%.

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.

$$J(w) = \frac{1}{2n} \sum_{i=1}^{n} \left(w^{(i)T} x^{(i)} - y^{(i)} \right)^2$$

$$J(w) = \frac{1}{2n} \left(w^T x - y \right)^T (w^T x - y)$$

Vectorised equation.

According to Newton's method

$$w^{(k+1)} = w^{(k)} - H^{-1} \nabla_w J(w)$$

where

$H \rightarrow$ is the Hessian of $J(w)$

Note :- from result in class

$$H\left[J(w)\right] = \frac{1}{n} X^T X$$

if $J(w)$ is in the form $\frac{1}{2n} (x^T w - y)^T (x^T w - y)$

$$\therefore H^{-1}\left[J(w)\right] = n \left(X^T X \right)^{-1}$$

$$\omega^{(k+1)} = \omega^{(k)} - \eta (X^T X)^{-1} \nabla_\omega J(\omega)$$

$$\nabla_\omega J(\omega) = \nabla_\omega \left[\frac{1}{2n} (X^T\omega - y)^T (X^T\omega - y) \right]$$

Note: from H.W 1

$$\nabla_x \left[(Ax+b)^T (Ax+b) \right] = 2A^T (Ax+b)$$

$$\nabla_\omega J(\omega) = \frac{1}{2n} \left[2X(X^T\omega - y) \right]$$

$$\Rightarrow \quad \omega^{(k+1)} = \omega^{(k)} - \eta (X^T X)^{-1} \cdot \left[\frac{1}{2n} \left[2X(X^T\omega - y) \right] \right]$$

$$= \omega^{(k)} - (X^T X)^{-1} \underbrace{\left[XX^T\omega - Xy \right]}_{\hookrightarrow \text{at } k^{th} \text{ point}}$$

$$= \omega^{(k)} - (X^T X)^{-1} (XX^T) \omega^{(k)} + (X^T X)^{-1} Xy$$

$$= \omega^{(k)} - \omega^{(k)} + (X^T X)^{-1} Xy$$

$\therefore$ $w^{(K+1)} = (X^T X)^{-1} X y$

Hence, no matter what / how many examples are there, w converges to $(X^T X)^{-1} X y$ in the first update itself.

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

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

($m \times 1$ for $\mathbf{w}^{(1)}$; $1 \times c$ for $\mathbf{W}$)

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**: $\sum_k a_k = a_l + \sum_{k \neq l} a_k$.)

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

Finally, show that

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

Sol':

$$\nabla_{\omega^{(l)}} z_k = \frac{d}{d\omega^{(l)}} \left[x^{(i)T} \omega^{(k)} + b_k \right]$$

$$\nabla_{\omega^{(l)}} z_k = \begin{cases} x^{(i)} & l = k \\ 0 & l \neq 0 \end{cases}$$

$$\begin{aligned}
\nabla_{\omega^{(l)}} \left(\sum_{k'=1}^{c} e^{z_k} \right) &= e^{z_k} \cdot \nabla_{\omega^{(1)}} z_1 \\
&+ e^{z_k} \cdot \nabla_{\omega^{(l)}} z_2 \\
&+ \\
&\vdots \\
&+ e^{z_k} \cdot \nabla_{\omega^{(l)}} z_c
\end{aligned}$$

$$\nabla_{\omega^{(l)}} \left(\sum_{k'=1}^{c} e^{z_k} \right) = \underbrace{e^{z_k} x^{(i)}}_{\hookrightarrow \text{ at } k = l} \Big/ e^{z_l} x^{(i)}$$

$$\hat{y}_k^{(i)} = \frac{e^{z_k}}{\sum_{k'=1}^{c} e^{z_k}} \; ; \; z_k = x^T w^{(k)} + b_k$$

$$\frac{d}{dx}\left(\frac{f}{g}\right) = \frac{gf' - fg'}{g^2}$$

Now,

$$\nabla_{w^{(l)}} \hat{y}_k^{(i)}$$

Case 1)

$l = k$

$$\nabla_{w^l} \hat{y}_k^{(i)} = \frac{\sum_{k'=1}^{c} e^{z_k}\left[e^{z_k} \nabla_{w^l} z_k\right] - e^{z_k}\left[\nabla_{w^l}\left(\sum_{k'=1}^{c} e^{z_k}\right)\right]}{\left(\sum_{k'=1}^{c} e^{z_k}\right)^2}$$

$$= \frac{\sum_{k'=1}^{c} e^{z_k}\left[e^{z_l} x^{(i)}\right] - e^{z_k}\left[e^{z_l} x^{(i)}\right]}{\left(\sum_{k'=1}^{c} e^{z_k}\right)^2}$$

$$= \frac{e^{z_l} x^{(i)}}{\sum_{k'=1}^{c} e^{z_k}} - \frac{e^{z_l}}{\sum_{k'=1}^{c} e^{z_k}} \times \frac{e^{z_l}}{\sum_{k'=1}^{c} e^{z_k}} \times x^{(i)}$$

$$= \hat{y}_l^{(i)} x^{(i)} - \left(\hat{y}_l^{(i)}\right)^2 x^{(i)}$$

$$\nabla_{w^l} \hat{y}_k^{(i)} = \hat{y}_l^{(i)} x^{(i)} \left(1 - \hat{y}_l^{(i)}\right) \quad \text{at } l = k$$

Case (ii) :-

$$l \neq k$$

$$\nabla_{\omega^l} \hat{y}_k^{(i)} = \frac{\sum_{k'=1}^{c} e^{z_k} \left[e^{z_k} \nabla_{\omega_k} z_k \right] - e^{z_k} \left[\nabla_{\omega_k} \left(\sum_{k'=1}^{c} e^{z_k} \right) \right]}{\left(\sum_{k'=1}^{c} e^{z_k} \right)^2}$$

$$= \frac{\sum_{k'=1}^{c} e^{z_k} \left[e^{z_k} \times 0 \right] - e^{z_k} \left[e^{z_l} x^{(i)} \right]}{\left(\sum_{k'=1}^{c} e^{z_k} \right)^2}$$

$$= 0 - \frac{e^{z_k}}{\sum_{k'=1}^{c} e^{z_k}} \times \frac{e^{z_l}}{\sum_{k'=1}^{c} e^{z_k}} \times x^{(i)}$$

$$= - \hat{y}_k^{(i)} \hat{y}_l^{(i)} x^{(i)}$$

$$\nabla_{\omega^l} \hat{y}_{ik}^{(i)} = - \hat{y}_k^{(i)} \hat{y}_l^{(i)} x^{(i)} \quad \text{at } l \neq k$$

$$\rightarrow \quad \nabla_{w^{(l)}} J_{CE}(W,b) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{c} y_k^{(i)} \nabla_{w^{(l)}} \log \hat{y}_k^{(i)}$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{c} y_k^{(i)} * \frac{1}{\hat{y}_k^{(i)}} * \nabla_{w^{(l)}} \hat{y}_k^{(i)}$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} * \frac{1}{\hat{y}_l^{(i)}} \nabla_{w^{(l)}} \hat{y}_l^{(i)} + \sum_{k\neq l} y_k^{(i)} * \frac{1}{\hat{y}_k^{(i)}} * \nabla_{w^{(l)}} \hat{y}_k^{(i)} \right]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} \cdot \frac{1}{\hat{y}_l^{(i)}} \cdot \left(x^{(i)} \hat{y}_l^{(i)} (1-\hat{y}_l^{(i)}) \right) + \sum_{k\neq l} y_k^{(i)} \cdot \frac{1}{\hat{y}_k^{(i)}} \cdot \left[-\hat{y}_l^{(i)} \hat{y}_k^{(i)} x^{(i)} \right] \right]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} \left(x^{(i)} (1-\hat{y}_l^{(i)}) + \sum_{k\neq l} y_k^{(i)} (-\hat{y}_l^{(i)} x^{(i)}) \right) \right]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} x^{(i)} - y_l^{(i)} \hat{y}_l^{(i)} x^{(i)} - \sum_{k\neq l} y_k^{(i)} \hat{y}_l^{(i)} x^{(i)} \right]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} x^{(i)} - \sum_{k=1}^{c} y_k^{(i)} \hat{y}_l^{(i)} x^{(i)} \right]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} x^{(i)} - x^{(i)} \hat{y}_l^{(i)} \sum_{k=1}^{c} y_k^{(i)} \right]$$

$\sum_{k=1}^{c} y_k^{(i)} \rightarrow = 1$

Note:-

We know that

$$\sum_{k=1}^{c} y_k^{(i)} = \sum_{k=1}^{c} \hat{y}_k^{(i)} = 1$$

bcoz'

$y_k^{(i)}$ & $\hat{y}_k^{(i)}$ are probabilities

& sum over all the classes will be equal to 1

$$= -\frac{1}{n} \sum_{i=1}^{n} \left[y_l^{(i)} x^{(i)} - x^{(i)} \hat{y}_l^{(i)} \right]$$

$$\nabla_{w^{(l)}} J_{CE}(W,b) = -\frac{1}{n} \sum_{i=1}^{n} x^{(i)} \left[y_l^{(i)} - \hat{y}_l^{(i)} \right]$$

Now,

Lets Proove for ∇_b

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

$$= -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{c} y_k^{(i)} \cdot \frac{1}{\hat{y}_k^{(i)}} \cdot \nabla_b \hat{y}_k^{(i)}$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} \cdot \frac{1}{\hat{y}_l^{(i)}} \nabla_{w^{(l)}} \hat{y}_l^{(i)} + \sum_{k\neq l} y_k^{(i)} \cdot \frac{1}{\hat{y}_k^{(i)}} \cdot \nabla_{w^{(l)}} \hat{y}_k^{(i)} \right]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} \cdot \frac{1}{\hat{y}_l^{(i)}} \cdot \left(x^{(i)} \hat{y}_l^{(i)} (1-\hat{y}_l^{(i)}) \right) + \sum_{k\neq l} y_k^{(i)} \cdot \frac{1}{\hat{y}_k^{(i)}} \cdot \left[-\hat{y}_l^{(i)} \hat{y}_k^{(i)} x^{(i)} \right] \right]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} \left(1-\hat{y}_l^{(i)}\right) + \sum_{k\neq l} y_k^{(i)} \left(-\hat{y}_l^{(i)}\right) \right]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} - y_l^{(i)} \hat{y}_l^{(i)} - \sum_{k\neq l} y_k^{(i)} \hat{y}_l^{(i)} \right]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} - \sum_{k=1}^{c} y_k^{(i)} \hat{y}_l^{(i)} \right]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_l^{(i)} - \hat{y}_l^{(i)} \underbrace{\sum_{k=1}^{c} y_k^{(i)}}_{\hookrightarrow = 1} \right]$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \left[y_k^{(i)} - \hat{y}_k^{(i)} \right]$$

$$\therefore \nabla_b f_{CE}(W, b) = -\frac{1}{n} \sum_{i=1}^{n} \left[y_k^{(i)} - \hat{y}_k^{(i)} \right]$$