Homework 3 – Deep Learning (CS/DS 541, Whitehill, Spring 2021)

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

Answer: The Hessian of the MSE loss for linear regression, as described in the lecture slides, is a constant: $\mathbf{H} = \frac{1}{n}\mathbf{X}\mathbf{X}^\top$, where n is the number of training examples. Let $\mathbf{w}$ be the initial weight vector. Then the updated weight vector is

$$\begin{aligned}
\mathbf{w}^{\text{new}} &= \mathbf{w} - \mathbf{H}^{-1}\nabla_{\mathbf{w}} f_{\text{MSE}} && (1)\\
&= \mathbf{w} - \left(\frac{1}{n}\mathbf{X}\mathbf{X}^\top\right)^{-1} \nabla_{\mathbf{w}} f_{\text{MSE}} && (2)\\
&= \mathbf{w} - n(\mathbf{X}\mathbf{X}^\top)^{-1}\nabla_{\mathbf{w}} f_{\text{MSE}} && (3)\\
&= \mathbf{w} - n(\mathbf{X}\mathbf{X}^\top)^{-1}\frac{1}{n}\mathbf{X}(\mathbf{X}^\top\mathbf{w} - \mathbf{y}) && (4)\\
&= \mathbf{w} - (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{X}^\top\mathbf{w} + (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y} && (5)\\
&= \mathbf{w} - \mathbf{I}\mathbf{w} + (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y} && (6)\\
&= (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y} && (7)\\
&= \mathbf{w}^* && (8)
\end{aligned}$$

2. **Derivation of softmax regression gradient updates** [25 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:

$$\begin{aligned}
\hat{y}_k &= \frac{\exp z_k}{\sum_{k'=1}^{c} \exp z_{k'}} && (9)\\
z_k &= \mathbf{x}^\top \mathbf{w}^{(k)} + b_k
\end{aligned}$$

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 9 – 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:

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

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, $\sum_k y_k = 1$.)

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

Finally, show that

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

Answer: For $l = k$:

$$\begin{aligned}
\nabla_{\mathbf{w}^{(l)}} \hat{y}_k^{(i)} &= \nabla_{\mathbf{w}^{(l)}} \left[\frac{\exp \mathbf{x}^{(i)\top} \mathbf{w}^{(l)}}{\sum_{k'=1}^{c} \exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k')}} \right] \\
&= \mathbf{x}^{(i)} \frac{\exp \mathbf{x}^{(i)\top} \mathbf{w}^{(l)}}{\sum_{k'=1}^{c} \exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k')}} - \mathbf{x}^{(i)} \frac{\exp \mathbf{x}^{(i)\top} \mathbf{w}^{(l)}}{\left(\sum_{k'=1}^{c} \exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k')} \right)^2} \exp(\mathbf{x}^{(i)\top} \mathbf{w}^{(l)}) \\
&= \mathbf{x}^{(i)} \left[\frac{\exp \mathbf{x}^{(i)\top} \mathbf{w}^{(l)}}{\sum_{k'=1}^{c} \exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k')}} - \frac{(\exp \mathbf{x}^{(i)\top} \mathbf{w}^{(l)})^2}{\left(\sum_{k'=1}^{c} \exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k')} \right)^2} \right] \\
&= \mathbf{x}^{(i)} \left[\hat{y}_l^{(i)} - \left(\hat{y}_l^{(i)} \right)^2 \right] \\
&= \mathbf{x}^{(i)} \hat{y}_l^{(i)} (1 - \hat{y}_l^{(i)})
\end{aligned}$$

For $l \neq k$:

$$\begin{aligned}
\nabla_{\mathbf{w}^{(l)}} \hat{y}_k^{(i)} &= \nabla_{\mathbf{w}^{(l)}} \left[\frac{\exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k)}}{\sum_{k'=1}^{c} \exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k')}} \right] \\
&= -\mathbf{x}^{(i)} \frac{\exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k)}}{(\sum_{k'=1}^{c} \exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k')})^2} \exp \mathbf{x}^{(i)\top} \mathbf{w}_l \\
&= -\mathbf{x}^{(i)} \frac{(\exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k)})(\exp \mathbf{x}^{(i)\top} \mathbf{w}_l)}{\left(\sum_{k'=1}^{c} \exp \mathbf{x}^{(i)\top} \mathbf{w}^{(k')} \right)^2} \\
&= -\mathbf{x}^{(i)} \hat{y}_k^{(i)} \hat{y}_l^{(i)}
\end{aligned}$$

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

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

The gradient w.r.t. $\mathbf{b}$ is derived exactly the same way as for $\nabla_{\mathbf{W}}$, except that there is no $\mathbf{x}^{(i)}$ term. Hence, we have

$$\nabla_{b_l} f_{\text{CE}}(\mathbf{W}, \mathbf{b}) = -\frac{1}{n} \sum_{i=1}^{n} \left[y_l^{(i)} - \hat{y}_l^{(i)} \right] \tag{10}$$

Combining all these (scalar) gradients into one vector, we obtain:

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

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

Train a 2-layer softmax neural network to classify images of 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 that you minimize should be

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

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.

Answer: Here are the key methods for performing the softmax, computing cross-entropy, and computing its gradient:

```
def softmax (z):
    denom = np.sum(np.exp(z), axis=1, keepdims=True)
    return np.exp(z) / denom

def fCE (W, X, Y, alpha = 0.):
    Yhat = softmax(X.T.dot(W))
    cost = - 1./X.shape[1] * np.sum(Y * np.log(Yhat))
    return cost

def gradK (W, X, Y, alpha = 0.):
    Yhat = softmax(X.T.dot(W))
    reg = alpha * W
    dfCEdw = 1./X.shape[1] * X.dot(Yhat - Y) + reg
    dfCEdb = 1./X.shape[1] * np.sum(Yhat - Y, axis=0)
    return dfCEdw, dfCEdb

def computeAccuracy (W, X, Y):
    Yhat = softmax(X.T.dot(W))
    return np.mean(np.argmax(Y, axis=1) == np.argmax(Yhat, axis=1))
```

Put your code in a Python file called `homework3_WPIUSERNAME1.py` (or `homework3_WPIUSERNAME1_WPIUSERNAME3.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_WPIUSERNAME3.pdf` for teams). Create a Zip file containing both your Python and PDF files, and then submit on Canvas.