# ECE 417 Multimedia Signal Processing
# Solutions to Homework 3

UNIVERSITY OF ILLINOIS
Department of Electrical and Computer Engineering

Assigned: Tuesday, 9/19/2023; Due: Tuesday, 10/3/2023

**Problem 3.1**

Suppose $y = \|\mathbf{AX}\|_F$, the Frobenius norm of $\mathbf{A}$ times $\mathbf{X}$ (note: NOT the squared Frobenius norm!) Find $\partial y/\partial \mathbf{X}$.

**Solution:** There are several approaches you could use. Since $\|\mathbf{AX}\|_F$ and $\|\mathbf{AX}\|_F^2$ are both scalars, the following approach works:

$$\begin{aligned}
\frac{\partial \|\mathbf{AX}\|_F}{\partial \mathbf{X}} &= \frac{\partial (\|\mathbf{AX}\|_F^2)^{1/2}}{\partial \|\mathbf{AX}\|_F^2} \frac{\partial \|\mathbf{AX}\|_F^2}{\partial \mathbf{X}} \\
&= \frac{1}{2} \frac{1}{(\|\mathbf{AX}\|_F^2)^{1/2}} \frac{\partial \operatorname{tr}\left((\mathbf{AX})^T \mathbf{AX}\right)}{\partial \mathbf{X}} \\
&= \frac{1}{2\|\mathbf{AX}\|_F} \frac{\partial \operatorname{tr}\left(\mathbf{X}^T \mathbf{A}^T \mathbf{AX}\right)}{\partial \mathbf{X}} \\
&= \frac{1}{2\partial \|\mathbf{AX}\|_F} \left( \frac{\partial \operatorname{tr}\left(\operatorname{sg}(\mathbf{X}^T) \mathbf{A}^T \mathbf{AX}\right)}{\partial \mathbf{X}} + \left( \frac{\partial \operatorname{tr}\left(\mathbf{X}^T \mathbf{A}^T \mathbf{A} \operatorname{sg}(\mathbf{X})\right)}{\partial \mathbf{X}^T} \right)^T \right) \\
&= \frac{1}{\|\mathbf{AX}\|_F} \mathbf{X}^T \mathbf{A}^T \mathbf{A}
\end{aligned}$$

**Problem 3.2**

You are trying to represent a mapping from a real-valued scalar input, $x \in \Re$, to a real-valued scalar output, $y \in \Re$. Suppose that you have a series of training examples, $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. You want to represent the mapping from $y$ to $x$ using the following two-layer network:

$$\begin{aligned}
\mathbf{z}_1 &= \mathbf{w}_1 x + \mathbf{b}_1 \\
\mathbf{a}_1 &= \mathrm{ReLU}(\mathbf{z}_1) \\
g(x) &= \mathbf{w}_2^T \mathbf{a}_1 + b_2
\end{aligned}$$

Your goal is to find $\mathbf{w}_1$, $\mathbf{b}_1$, $\mathbf{w}_2$ and $b_2$ in order to minimize $\mathcal{L}$ which is defined as

$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^{n} (y_i - g(x_i))^2$$

How many hidden nodes (what dimension of $\mathbf{w}_1$) do you need to guarantee that $\mathcal{L} = 0$? In terms of the training tokens $x_i$ and $y_i$, choose some values of $\mathbf{w}_1$, $\mathbf{b}_1$, $\mathbf{w}_2$ and $b_2$ that will result in $\mathcal{L} = 0$. Hint: you may find it convenient to assume that the data are sorted in order of increasing $x$, i.e., $x_i < x_{i+1}$ for all $i$.

**Solution:** We need at least $n - 1$ hidden nodes to make sure that $\mathcal{L} = 0$. Taking advantage of the hint, we can set the offsets and slopes so that the first training token is represented by $b_2$, and each succeeding training example is represented by the cumulative sum of ReLUs for all previous $x_i$. There are a few different ways in which this can be done; here is one:

$$\begin{aligned} b_2 &= y_1 \\ b_{1,i} &= -x_{i-1}, \quad 2 \le i \le n \\ w_{1,i} &= 1 \\ w_{2,i} &= \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \sum_{j=1}^{i-1} w_{2,j} \end{aligned}$$

**Problem 3.3**

Suppose you have a neural net that is configured as a binary classifier. Given a vector input $\mathbf{x}$, it computes a scalar output $g(\mathbf{x}) \in (0, 1)$ according to

$$\begin{aligned} \mathbf{z}_1 &= \mathbf{W}_1\mathbf{x} + \mathbf{b}_1 \\ \mathbf{a}_1 &= \mathrm{ReLU}(\mathbf{z}_1) \\ z_2 &= \mathbf{w}_2^T\mathbf{a}_1 + b_2 \\ g(\mathbf{x}) &= \sigma(z_2) \end{aligned}$$

where $\sigma(\cdot)$ is the logistic sigmoid, and $\mathrm{ReLU}(\mathbf{z}_1)$ applies rectified linear nonlinearities to each element of $\mathbf{z}_1$. Suppose that the loss function is binary cross entropy, i.e.,

$$\mathcal{L} = -y \ln g(\mathbf{x}) - (1 - y) \ln(1 - g(\mathbf{x}))$$

What is $\partial\mathcal{L}/\partial\mathbf{W}_1$? Express your answer in terms of any of the vectors or matrices specified in the problem, but if you have to define any additional matrices (e.g., if you have to define $\mathrm{ReLU}'(\mathbf{z}_1)$), be sure to specify the values of every element of every such matrix in terms of the elements of the matrices and vectors provided.

**Solution:** First, the derivative w.r.t. the matrix can be done element-by-element:

$$\begin{aligned} z_{1,j} &= \sum_k w_{1,j,k}x_k + b_j \\ \frac{\partial\mathcal{L}}{\partial w_{1,j,k}} &= \frac{\partial\mathcal{L}}{\partial z_{1,j}}x_k \\ \frac{\partial\mathcal{L}}{\partial\mathbf{W}_1} &= \mathbf{x}\frac{\partial\mathcal{L}}{\partial\mathbf{z}_1} \end{aligned}$$

Now we need to find the derivative w.r.t. $\mathbf{z}_1$. For this, we can use the chain rule:

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\mathbf{z}_1} &= \frac{\partial\mathcal{L}}{\partial g(\mathbf{x})}\frac{\partial g(\mathbf{x})}{\partial z_2}\frac{\partial z_2}{\partial\mathbf{a}_1}\frac{\partial\mathbf{a}_1}{\partial\mathbf{z}_1} \\ &= -\left(\frac{y}{g(\mathbf{x})} - \frac{1-y}{1-g(\mathbf{x})}\right)\sigma'(z_2)\mathbf{w}_2^T\,\mathrm{ReL\acute{U}}(\mathbf{z}_1), \end{aligned}$$

where $\sigma'(z_2) = \sigma(z_2)(1-\sigma(z_2))$, and $\text{ReLU}'(\mathbf{z}_1)$ is a diagonal matrix whose $(i,i)^{\text{th}}$ element is 1 if $z_{1,i} > 0$, and 0 otherwise. Putting it all together, we get

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1} = -\mathbf{x}\left(\frac{y}{g(\mathbf{x})} - \frac{1-y}{1-g(\mathbf{x})}\right)\sigma'(z_2)\mathbf{w}_2^T \overset{'}{\text{ReLU}}(\mathbf{z}_1)$$

Note that further simplifications of this answer are possible, though not required by the problem statement. For example, $\sigma'(z_2) = g(\mathbf{x})(1-g(\mathbf{x}))$, and therefore, after some simplification,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1} = (g(\mathbf{x}) - y)\mathbf{x}\mathbf{w}_2^T \overset{'}{\text{ReLU}}(\mathbf{z}_1)$$

**Problem 3.4**

The filters in convolutional neural nets tend to learn common features of the images on which they are trained. To see how this occurs, consider the following CNN, composed of a one-channel convolutional layer followed by one global pooling layer, with a grayscale input image:

$$z_1[m,n] = w_1[m,n] * x[m,n] \tag{3.4-1}$$

$$g(x) = \max_{m=0}^{M-1}\max_{n=0}^{N-1} z_1[m,n] \tag{3.4-2}$$

Suppose that, initially, $w_1[m,n] = \delta[m,n]$, i.e., $w[0,0] = 1$ and $w[m,n] = 0$ for all other $(m,n)$. The maximization in Eq. (3.4-2) therefore picks out the maximum sample in the input signal, which is at position $(m^*, n^*)$:

$$(m^*, n^*) = \operatorname*{argmax}_{m=0}^{M-1}\operatorname*{argmax}_{n=0}^{N-1} z_1[m,n]$$

Suppose that the loss function is MSE, $\mathcal{L} = \frac{1}{2}(y - g(x))^2$, and suppose that the CNN is trained for one step of gradient descent with a learning rate of $\eta = 1$:

$$w_1[m,n] \leftarrow w_1[m,n] - \frac{\partial \mathcal{L}}{\partial w[m,n]}$$

After the one step of training shown above, what is $w_1[m,n]$? Write your answer in terms of $y$, $x$, $g(x)$, $\delta[m,n]$, $m^*$, $n^*$, and any of the other variables that have been introduced in the problem statement.

**Solution:**

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w[m,n]} &= (g(x) - y)\sum_{m'}\sum_{n'}\left(\frac{\partial g(x)}{\partial z_1[m',n']}\right)\left(\frac{\partial z_1[m',n']}{\partial w_1[m,n]}\right) \\ &= (g(x) - y)x[m^* - m, n^* - n]\end{aligned}$$

Therefore

$$w_1[m,n] \leftarrow \delta[m,n] + (y - g(x))x[m^* - m, n^* - n]$$

In other words, it is a copy of $x[m,n]$, flipped and shifted by $(m^*, n^*)$, scaled by the error term $(y - g(x))$, and added to the original filter coefficients $\delta[m,n]$.