1. (15 points) **Backpropagation for autoencoders.** In an autoencoder, we seek to reconstruct the original data after some operation that reduces the data's dimensionality. We may be interested in reducing the data's dimensionality to gain a more compact representation of the data.

For example, consider $\mathbf{x} \in \mathbb{R}^n$. Further, consider $\mathbf{W} \in \mathbb{R}^{m \times n}$ where $m < n$. Then $\mathbf{Wx}$ is of lower dimensionality than $\mathbf{x}$. One way to design $\mathbf{W}$ so that $\mathbf{Wx}$ still contains key features of $\mathbf{x}$ is to minimize the following expression

$\mathbf{Wx} \in \mathbb{R}^{(m \times n)(n \times 1) = (m \times 1)}$

$$\mathcal{L} = \frac{1}{2} \left\| \mathbf{W}^T \mathbf{W} \mathbf{x} - \mathbf{x} \right\|^2$$

with respect to $\mathbf{W}$. (To be complete, autoencoders also have a nonlinearity in each layer, i.e., the loss is $\frac{1}{2} \left\| f(\mathbf{W}^T f(\mathbf{Wx})) - x \right\|^2$. However, we'll work with the linear example.)

(a) (3 points) In words, describe why this minimization finds a $\mathbf{W}$ that ought to preserve information about $\mathbf{x}$.

(b) (3 points) Draw the computational graph for $\mathcal{L}$. **Hint:** You can set up the computational graph to this problem in a way that will allow you to solve for part (d) without taking 4D tensor derivative.

(c) (3 points) In the computational graph, there should be two paths to $\mathbf{W}$. How do we account for these two paths when calculating $\nabla_{\mathbf{W}} \mathcal{L}$? Your answer should include a mathematical argument.

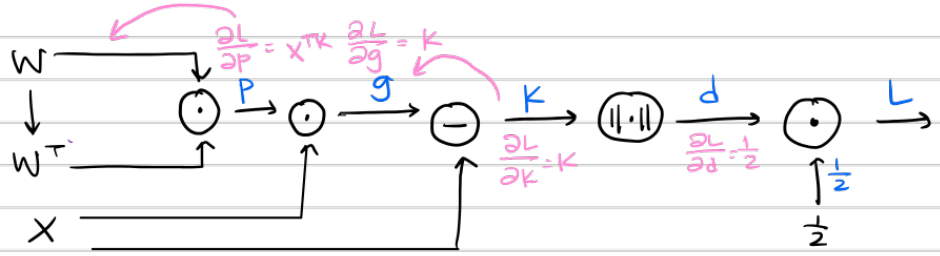
(d) (6 points) Calculate the gradient: $\nabla_{\mathbf{W}} \mathcal{L}$.

a) reconstructing back to x; reduction into lower dimensionality; squared loss

$$\text{In} \quad L = \frac{1}{2} \| W^T W x - x \|^2$$

The objective of the minimization function is to find a matrix W that when reconstructed via W^TWx, preserves the most information of x through minimizing the loss $\mathcal{L}$.

b)



c) To account for the 2 paths leading to W, we can use the law of total derivatives and sum the gradients of the 2 paths.

$$\nabla_W L = \frac{\partial L}{\partial W} + \left(\frac{\partial L}{\partial W^T} \right)^T$$

d) $L = \frac{1}{2} \| K \|^2 \rightarrow \frac{\partial L}{\partial K} = K$

$K = g - x \rightarrow \frac{\partial L}{\partial g} = \frac{\partial K}{\partial g} \cdot \frac{\partial L}{\partial K} = 1 \cdot K$

$g = Px \rightarrow \frac{\partial L}{\partial P} = \frac{\partial g}{\partial P} \cdot \frac{\partial L}{\partial g} = x^T \cdot K$

$P = W^T W \rightarrow \frac{\partial L}{\partial W} = \frac{\partial P}{\partial W} \cdot \frac{\partial L}{\partial P} = W^T x^T K = \underline{WxK^T}$

$\rightarrow \frac{\partial L}{\partial W^T} = \frac{\partial P}{\partial W^T} \cdot \frac{\partial L}{\partial P} = (W^T)^T x^T K = Wx^T K = \underline{Wkx^T}$

$\boxed{\frac{\partial L}{\partial W} = WxK^T + Wkx^T}$

In the architecture shown, D represents the number of neurons in input layer, H represents the number of neurons in hidden layer, and C represents the number of neurons in the output layer (in our design $C = 7$). The output is then passed through a softmax classifier. Although we learned about the ReLu activation in class, we decided to use the Swish activation function (introduced by Google Brain) for the hidden layer. The Swish activation function for any scalar input k is defined as,

$$\text{swish}(k) = \frac{k}{1+e^{-k}} = k\sigma(k),$$

where $\sigma(k)$is the sigmoid activation function you have seen in lecture.

D = # input neurons
H = # Hidden neurons
C = # output neurons = 7

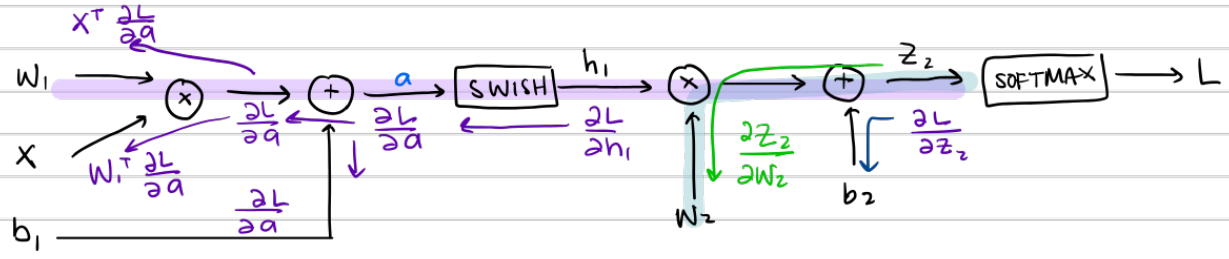
You will train the 2-layer FC net using gradient descent and for that you will need to compute the gradients. For the gradient computations, you are allowed to keep your final answer in terms of $\frac{\partial L}{\partial z_2}$.

(a) (3 points) Draw the computational graph for the 2-layer FC net.

(b) (5 points) Compute $\nabla_{W_2} L$, $\nabla_{b_2} L$.

(c) (7 points) Compute $\nabla_{W_1} L$, $\nabla_{b_1} L$.

a)



b) $z_2 = W_2 h_1 + b_2$ $\quad h_1 = a \cdot \sigma(a) = \frac{a}{1+e^{-a}}$ $\quad a = W_1 X + b_1$

upstream grad $= \frac{\partial L}{\partial z_2}$

$\nabla_{W_2} L = \frac{\partial z_2}{\partial W_2} \cdot \frac{\partial L}{\partial z_2} = \boxed{h_1 \cdot \frac{\partial L}{\partial z_2}}$ (switcharoo?)

↳ $\frac{\partial z_2}{\partial W_2}(W_1 h_1 + b_2) = h_1$

$\frac{\partial L}{\partial a} = \frac{\partial h_1}{\partial a} \cdot \frac{\partial L}{\partial h_1}$

$\nabla_{b_2} L = \frac{\partial z_2}{\partial b_2} \cdot \frac{\partial L}{\partial z_2} = \boxed{\frac{\partial L}{\partial z_2}}$

↳ $= 1$

$\frac{\partial h_1}{\partial a}\left(\frac{a}{1+e^{-a}}\right)$

c) $\nabla_{W_1} L = \frac{\partial L}{\partial a} \cdot X^T = \boxed{\sigma(a) + a\left[\sigma(a)(1-\sigma(a))\right] X^T}$

↳ $\frac{\partial L}{\partial a} = \frac{\partial h_1}{\partial a} \cdot \frac{\partial L}{\partial h_1}$

↳ $h_1 = \frac{a}{1+e^{-a}} = a \cdot \sigma(a)$

$\frac{\partial h_1}{\partial a} = \sigma(a) + a\sigma'(a)$

$\sigma'(a) = \frac{\partial}{\partial a}\frac{1}{1+e^{-a}} = \sigma(a)(1-\sigma(a))$

$= \sigma(a) + a\left[\sigma(a)(1-\sigma(a))\right]$

$\nabla_{b_2} L = \frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial a} =$ ↗

2. I am a C147 student ;)