# CSCI-GA 2572 Deep Learning - Homework 1

Due: September 17 @ 11:59pm

**Name**: Stephen Spivack (ss7726)

# 1 Theory

## 1.1 Two-layer neural nets

You are given the following neural net architecture:

$Linear_1 \rightarrow f \rightarrow Linear_2 \rightarrow g$

Where $Linear_i(x) = W^{(i)}x + b^{(i)}$ is the $i$th affine transformation, and $f, g$ are element-wise nonlinear activation function's. When an input $x \in \mathbb{R}^n$ is fed to the network, $\tilde{y} \in \mathbb{R}^K$ is obtained as the output.

### 1.1.1 Regression Task

We would like to perform regression task. We choose $f(\cdot) = 5(\cdot)^+ = 5ReLU(\cdot)$ and $g$ to be the identity function. To train this network, we want to minimize the energy loss $L$ and this is computed via the squared euclidean distance cost $C$, such that $L(w, x, y) = F(x, y) = C(y, \tilde{y}) = ||\tilde{y} - y||^2$, where y is the output target.

(a) Name and mathematically describe the 5 programming steps you would take to train this model with PyTorch using SGD on a single batch of data.

Step 1: forward pass $\tilde{y}^{(i)} = F(x_i)$

Step 2: compute loss $L(\tilde{y}^{(i)}, y^{(i)})$

Step 3: compute gradient of loss $\nabla_w = \sum_{\nabla_w} \frac{\partial L}{\partial w_{(i)}}$

Step 4: Update parameters $w \leftarrow w - \eta \nabla_w L(\tilde{y}^{(i)}, y^{(i)})$

Step 5: zero gradient $\nabla_w L(\tilde{y}^{(i)}, y^{(i)}) \leftarrow 0$

(b) For a single data point $(x, y)$, write down all inputs and outputs for forward pass of each layer. You can only use variable $x, y, W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}$ in your answer. Note that $Linear_i(x) = W^{(i)}x + b^{(i)}$.

$Linear_1$ - perform affine transformation $W^{(1)}x + b^{(1)}$

$f$ - apply ReLU activation $5\max(0, W^{(1)}x + b^{(1)})$

$Linear_2$ - perform affine transformation $W^{(2)}(5\max(0, W^{(1)}x + b^{(1)})) + b^{(2)}$

$g$ - apply identity activation $W^{(2)}(5\max(0, W^{(1)}x + b^{(1)})) + b^{(2)}$ to compute the final output of the model for a single input $(x, y)$.

Final output: $W^{(2)}(5\max(0, W^{(1)}x + b^{(1)})) + b^{(2)}$

(c) Write down the gradients calculated from the backward pass.

You can only use the following variables $x, y, W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, \frac{\partial C}{\partial \tilde{y}}, \frac{\partial a_1}{\partial s_1}, \frac{\partial \tilde{y}}{\partial s_2}$ in your answer, where $s_1, a_1, s_2, \tilde{y}$ are the outputs of $Linear_1, f, Linear_2, g$

Gradients for output layer ($W^{(2)}$ and $b^{(2)}$):

$\frac{\partial C}{\partial W^{(2)}} = \frac{\partial C}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial s_2} \frac{\partial s_2}{\partial W^{(2)}} = 2(\tilde{y} - y)a_1$

$\frac{\partial C}{\partial b^{(2)}} = \frac{\partial C}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial s_2} \frac{\partial s_2}{\partial b^{(2)}} = 2(\tilde{y} - y)$

Gradients for hidden layer ($W^{(1)}$ and $b^{(1)}$):

$\frac{\partial C}{\partial W^{(1)}} = \frac{\partial C}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial s_2} \frac{\partial s_2}{\partial a_1} \frac{\partial a_1}{\partial s_1} \frac{\partial s_1}{\partial W^{(1)}} = 2(\tilde{y} - y)W^{(2)} \frac{\partial a_1}{\partial s_1} x$

$\frac{\partial C}{\partial b^{(1)}} = \frac{\partial C}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial s_2} \frac{\partial s_2}{\partial a_1} \frac{\partial a_1}{\partial s_1} \frac{\partial s_1}{\partial b^{(1)}} = 2(\tilde{y} - y)W^{(2)} \frac{\partial a_1}{\partial s_1}$

(d) Show us the elements of $\frac{\partial a_1}{\partial s_1}$, $\frac{\partial \tilde{y}}{\partial s_2}$, $\frac{\partial C}{\partial \tilde{y}}$ (be careful about the dimensionality).

$\frac{\partial a_1}{\partial s_1} = 5$ if $s_1 > 0$ or 0 if $s_1 \leq 0$

$\frac{\partial \tilde{y}}{\partial s_2}$ is simply a vector of ones with dimensionality $n$

$\frac{\partial C}{\partial \tilde{y}} = 2(\tilde{y} - y)$

### 1.1.2 Classification Task

We would like to perform multi-class classification, so we set $f = tanh$ and $g = \sigma$, the logistic sigmoid function $\sigma(x) = (1 + exp(-x))^{-1}$

(a) If you want to train this network, what do you need to change in the equations of (b), (c) and (d), assuming we are using the same squared Euclidean distance loss function.

For the equations in (b):

$Linear_1$ - perform affine transformation $W^{(1)}x + b^{(1)}$

$f$ - apply tanh activation $\tanh(W^{(1)}x + b^{(1)})$

$Linear_2$ - perform affine transformation $W^{(2)} \tanh(W^{(1)}x + b^{(1)}) + b^{(2)}$

$g$ - apply logistic sigmoid activation $\sigma(W^{(2)} \tanh(W^{(1)}x + b^{(1)}) + b^{(2)})$ to compute the final output of the model for a single input $(x, y)$.

For the equations in (c):

$\frac{\partial C}{\partial W^{(2)}} = \frac{\partial C}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial s_2} \frac{\partial s_2}{\partial W^{(2)}} = 2(\tilde{y} - y)\sigma(s_2)(1 - \sigma(s_2))a_1$

$\frac{\partial C}{\partial b^{(2)}} = \frac{\partial C}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial s_2} \frac{\partial s_2}{\partial b^{(2)}} = 2(\tilde{y} - y)\sigma(s_2)(1 - \sigma(s_2))$

$\frac{\partial C}{\partial W^{(1)}} = \frac{\partial C}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial s_2} \frac{\partial s_2}{\partial a_1} \frac{\partial a_1}{\partial s_1} \frac{\partial s_1}{\partial W^{(1)}} = 2(\tilde{y} - y)\sigma(s_2)(1 - \sigma(s_2))W^{(2)}(1 - \tanh^2(s_1))x$

$\frac{\partial C}{\partial b^{(1)}} = \frac{\partial C}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial s_2} \frac{\partial s_2}{\partial a_1} \frac{\partial a_1}{\partial s_1} \frac{\partial s_1}{\partial b^{(1)}} = 2(\tilde{y} - y)\sigma(s_2)(1 - \sigma(s_2))W^{(2)}(1 - \tanh^2(s_1))$

For the equations in (d):

$\frac{\partial a_1}{\partial s_1} = 1$ if $s_1 > 0$ or 0 if $s_1 \leq 0$

$\frac{\partial \tilde{y}}{\partial s_2} = 1$

$\frac{\partial C}{\partial \tilde{y}} = 2(\tilde{y} - y)$

(b) Now you think you can do a better job by using a Binary Cross Entropy (BCE) loss functions $D_{BCE}(i, \tilde{y}) = \frac{1}{K} \sum_{i=1}^{K} -[y_i log(\tilde{y}_i) + (1 - y_i) log(1 - \tilde{y}_i)]$. What do you need to change in the equations of (b), (c) and (d)?

For the equations in (b):

$Linear_1$ - perform affine transformation $W^{(1)}x + b^{(1)}$

$f$ - apply tanh activation $\tanh(W^{(1)}x + b^{(1)})$

$Linear_2$ - perform affine transformation $W^{(2)} \tanh(W^{(1)}x + b^{(1)}) + b^{(2)}$

$g$ - apply logistic sigmoid activation $\sigma(W^{(2)} \tanh(W^{(1)}x + b^{(1)}) + b^{(2)})$ to compute the final output of the model for a single input $(x, y)$.

For the equations in (c):

$\frac{\partial C}{\partial W^{(2)}} = -\frac{2}{K} \sum_{i=1}^{K} \left( \frac{y_i}{\tilde{y}_i} - \frac{1 - y_i}{1 - \tilde{y}_i} \right) \sigma(s_2) \tanh(W^{(1)}x + b^{(1)})x$

$\frac{\partial C}{\partial b^{(2)}} = -\frac{2}{K}\sum_{i=1}^{K}\left(\frac{y_i}{\tilde{y}_i} - \frac{1-y_i}{1-\tilde{y}_i}\right)\sigma(s_2)$

$\frac{\partial C}{\partial W^{(1)}} = -\frac{2}{K}\sum_{i=1}^{K}\left(\frac{y_i}{\tilde{y}_i} - \frac{1-y_i}{1-\tilde{y}_i}\right)\sigma(s_2)W^{(2)}(1-\tanh^2(s_1))x$

$\frac{\partial C}{\partial b^{(1)}} = -\frac{2}{K}\sum_{i=1}^{K}\left(\frac{y_i}{\tilde{y}_i} - \frac{1-y_i}{1-\tilde{y}_i}\right)\sigma(s_2)W^{(2)}(1-\tanh^2(s_1))$

For the equations in (d):

$\frac{\partial a_1}{\partial s_1} = 1$ if $s_1 > 0$ or 0 if $s_1 \leq 0$.

$\frac{\partial \tilde{y}_i}{\partial s_2} = \sigma(s_2)(1-\sigma(s_2))$

$\frac{\partial C}{\partial \tilde{y}_i} = -\left(\frac{y_i}{\tilde{y}_i} - \frac{1-y_i}{1-\tilde{y}_i}\right)$

(c) Things are getting better. You realize that not all intermediate hidden activations need to be binary (or soft version of binary). You decide to use $f(\cdot) = (\cdot)^+$ but keep $g$ as $\sigma$. Explain why this choice of $f$ can be beneficial for training a (deeper) network.

This is the ReLU activation function. The main reason it is ideal for deeper networks is that it mitigates against the vanishing gradient problem, which occurs once values get small enough.

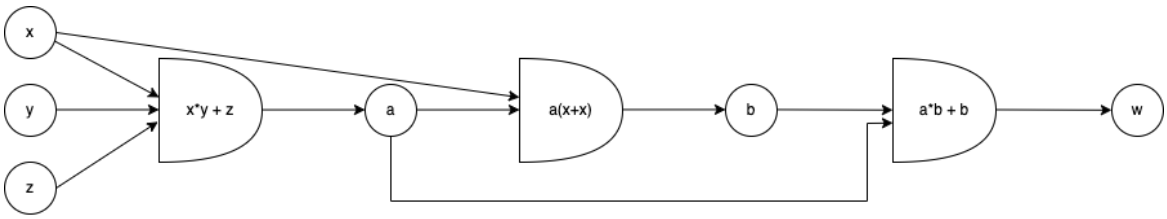
## 1.2 Conceptual Questions

(a) Why is softmax actually softargmax?

Because it preserves the index of each element in the array and returns a normalized probability distribution.

(b) Draw the computational graph defined by this function, with inputs $x, y, z \in \mathbb{R}$ and output $w \in \mathbb{R}$. You make use symbols $x, y, z, o$, and operators $*, +$ in your solution. Be sure to use the correct shape for symbols and operators as shown in class.

$$a = xy + z$$
$$b = a(x + x)$$
$$w = ab + b$$



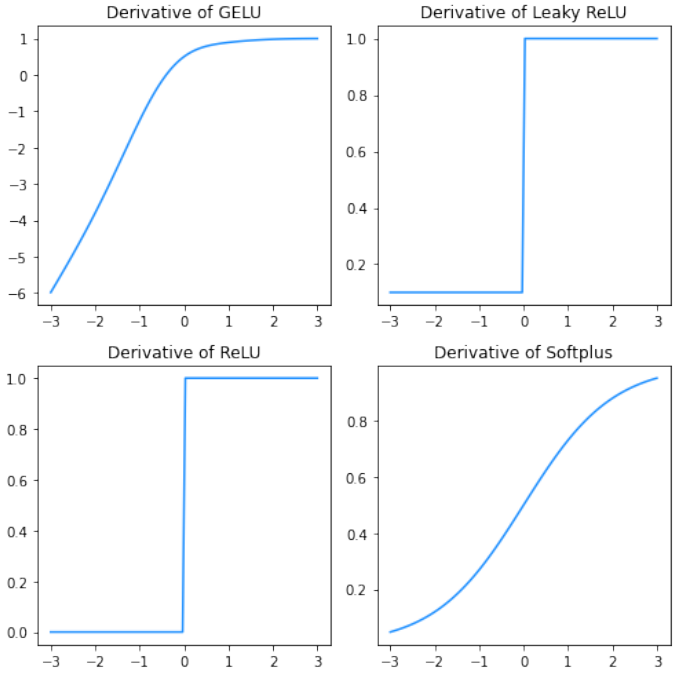

(c) Draw the graph of the derivative for the following functions:

$GELU()$

$LeakyReLU(negativeslope = 0.1)$

$RELU()$

$Softplus(beta = 1)$




(d) Given function $f(x) = W_1 x$ with $W_1 \in \mathbb{R}^{b \times a}$ and $g(x) = W_2 x$ with $W_2 \in \mathbb{R}^{b \times a}$:

a. What is the Jacobian matrix of $f$ and $g$?

$J_f(x) = \frac{\partial f}{\partial x} = W_1$

$J_g(x) = \frac{\partial g}{\partial x} = W_2$

b. What is the Jacobian matrix of $h(x) = f(x) + g(x)$?

$J_h(x) = J_f(x) + J_g(x) = W_1 + W_2$

c. What is the Jacobian matrix of $h(x) = f(x) + g(x)$ if $W_1 = W_2$

$J_h(x) = W_1 + W_1 = 2W_1$

(e) Given function $f(x) = W_1 x$ with $W_1 \in \mathbb{R}^{b \times a}$ and $g(x) = W_2 x$ with $W_2 \in \mathbb{R}^{c \times b}$:

a. What is the Jacobian matrix of $f$ and $g$?

$J_f(x) = \frac{\partial f}{\partial x} = W_1$

$J_g(x) = \frac{\partial g}{\partial x} = W_2$

b. What is the Jacobian matrix of $h(x) = g(f(x)) = (g \circ f)(x)$?

$J_h(x) = J_f(x) \times J_g(x) = W_1 \times W_2$ matrix product

c. What is the Jacobian matrix of $h(x)$ if $W_1 = W_2$ (so $a = b = c$)

$J_h(x) = J_f(x) \times J_g(x) = W_1 \times W_1 = W_1^2$

## 1.3 Deriving Loss Functions

Derive the loss function for the following algorithms based on their common update rule $w_i \leftarrow w_i + \eta(y - \tilde{y})x_i$. Show the steps of the derivation given the following inference rules (simply stating the final loss function will receive no points).

1. Perceptron:

$\tilde{y} = sign(b + \sum_{i=1}^{d} w_i x_i)$

$w_i \leftarrow w_i + \eta(y - \tilde{y})x_i$

$w_i \leftarrow w_i + \eta(y - sign(b + \sum_{i=1}^{d} w_i x_i))x_i$

$L_{perc}(x, y, w) = -(y - sign(b + \sum_{i=1}^{d} w_i x_i)) \sum_{i=1}^{d} w_i x_i$

$L_{perc}(x, y, w) = -(y - \tilde{y}) \sum_{i=1}^{d} w_i x_i$

2. Adaline / Least Mean Squares:

$\tilde{y} = b + \sum_{i=1}^{d} w_i x_i$

$w_i \leftarrow w_i + \eta(y - \tilde{y})x_i$

$w_i \leftarrow w_i + \eta(y - (b + \sum_{i=1}^{d} w_i x_i))x_i$

$L_{ada}(x, y, w) = \frac{1}{2}(y - (b + \sum_{i=1}^{d} w_i x_i))^2$

$L_{ada}(x, y, w) = \frac{1}{2}(y - \tilde{y})^2$

3. Logistic Regression:

$\tilde{y} = tanh(b + \sum_{i=1}^{d} w_i x_i)$

$w_i \leftarrow w_i + \eta(y - \tilde{y})x_i$

$w_i \leftarrow w_i + 2\eta(\frac{1}{2}(y - \tilde{y}))x_i$

$w_i \leftarrow w_i + 2\eta log(\frac{1}{1+exp(-(y-\tilde{y}))})x_i$

$w_i \leftarrow w_i + 2\eta log(1 + exp(-(y - \tilde{y})))x_i$

$L_{logreg}(x, y, w) = -2log(1 + exp(-y \sum_{i=1}^{d} w_i x_i))$