Due Date: October 19th, 2023 at 11:00 pm

Instructions

- *For all questions, show your work!*
- *Use LaTeX and the template we provide when writing your answers. You may reuse most of the notation shorthands, equations and/or tables. See the assignment policy on the course website for more details.*
- *The use of AI tools like Chat-GPT to find answers or parts of answers for any question in this assignment is not allowed. However, you can use these tools to improve the quality of your writing, like fixing grammar or making it more understandable. If you do use these tools, you must clearly explain how you used them and which questions or parts of questions you applied them to. Failing to do so or using these tools to find answers or parts of answers may result in your work being completely rejected, which means you'll receive a score of 0 for the entire theory or practical section.*
- *Submit your answers electronically via Gradescope.*
- *TAs for this assignment are* ***Saba Ahmadi, Sahar Dastani, and Sophie Xhonneux.***

Question 1 (5-2-2-1-3-2)**.** (**Activation Functions and Backpropagation**) *Please note that during marking we will not read more than the question says you should write, e.g. if the question asks for one sentence, only the first sentence will be read.*

1. Given the following 3 layer neural network $\boldsymbol{o} = f(\boldsymbol{x})$

$$\begin{aligned} \boldsymbol{h} &= \boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1 \\ \boldsymbol{a} &= \sigma(\boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2) \\ \boldsymbol{o} &= \boldsymbol{W}_3\boldsymbol{a} + \boldsymbol{b}_3, \end{aligned} \tag{1}$$

   where $\boldsymbol{x} \in \mathrm{R}^n$, $\boldsymbol{h} \in \mathrm{R}^m$, $\boldsymbol{a} \in \mathrm{R}^r$, $\boldsymbol{o} \in \mathrm{R}^s$, $\sigma(x) = \frac{1}{1+e^{-x}}$, and a loss function $\mathrm{L}(\boldsymbol{o}, \boldsymbol{y})$ apply the back-propagation algorithm computing the gradients with respect to the weights $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$ to show which are the necessary intermediate vectors that we need to store during the forward pass such as to not need recomputation. Use the notation provided above.
2. Assuming each floating point number takes two bytes to store, give the amount of memory in bytes, needed to store during the forward pass for the backward pass to not need recomputation.
3. Compute the left and right derivative of the `ReLU` function at $x = 0$. Show your work.
4. Give **one** sentence to explain why we cannot use the `ReLU` function if we want to differentiate our neural network twice.
5. The swish activation function is defined as $g(x) = x\sigma(\beta x)$. Simplify the function in the limit as β approaches 0 and infinity. Show your work.
6. Give 1 reason (one sentence) as to why we might want to use the swish function over the `ReLU`. Give 1 reason (one sentence) as to why we might want to use the `ReLU` over the swish (hint: monotonicity).

Answer 1. Q1.1) Given the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The derivative of the sigmoid function is:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

The partial derivative of the loss function L(o,y) for every parameter,
- Gradient with respect to output o:

$$\frac{\partial L}{\partial o} = l$$

- Gradient with respect to W_3, b_3:

$$\frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial o} \times a$$

$$\frac{\partial L}{\partial b_3} = \frac{\partial L}{\partial o}$$

- Gradient with respect to a:

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial o} \times W_3$$

- Gradient with respect to W_2 and b_2:

$$\frac{\partial L}{\partial (W_2 h + b_2)} = \frac{\partial L}{\partial a} \times \sigma'(W_2 h + b_2)$$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial (W_2 h + b_2)} \times h$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial (W_2 h + b_2)}$$

- Gradient with respect to h:

$$\frac{\partial L}{\partial h} = \frac{\partial L}{\partial (W_2 h + b_2)} \times W_2$$

- Gradient with respect to W_1 and b_1:

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial h} \times x$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial h}$$

For efficient backpropagation, we need to store: 1) h, the output of the first linear layer, which is needed when computing the gradient with respect to W_2 and b_2.
2) a, the output of the hidden layer after activation, which is needed when computing the gradient with respect to W_2 and b_2.
3) o, this value is output layer activation, which is needed (because we need $\frac{\partial L}{\partial o}$) when computing the gradient for other weights.

Q1.2) Given that each floating-point number requires two bytes of storage, we can compute the storage needs for the h, a and o vectors. $\boldsymbol{h}$ requires $\boldsymbol{m}$ floating point numbers. $\boldsymbol{a}$ and $\boldsymbol{o}$ need r and s floating-point values respectively. So, total memory needed = 2m+2r+2s bytes.

Q1.3) As we know, the ReLU function is defined as:

$$\text{ReLU}(x) = \max(0, x)$$

where any input less than 0 is mapped to 0, and any input greater than 0 is mapped to itself (x). Gradient of ReLU with respect to output x, for x<0 is 0 and for x>0 is 1. However, at x=0, the ReLU isn't differentiable because there's a corner point. So, we should compute left and right derivatives.
- Left Derivative:

$$\lim_{x \to 0^-} \frac{\partial ReLU}{\partial X} = 0$$

- Right Derivative:

$$\lim_{x \to 0^+} \frac{\partial ReLU}{\partial X} = 1$$

Q1.4) The ReLU function isn't differentiable at x=0, leading to an undefined second derivative at this location.

Q1.5) We know that the sigmoid function $\sigma(x)$ is given by:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

- The Sigmoid function in the limit as β approaches 0:

$$\lim_{\beta \to 0} x\sigma(\beta x)$$

As β tends toward 0, βx converges to 0 irrespective of the value of x. Consequently, the value of the sigmoid function approaches 0.5 in this limit:

$$\sigma(0) = \frac{1}{1+e^0} = \frac{1}{2}$$

Hence, g(x) can be reduced to: $g(x) \approx 0.5x$
- The Sigmoid function in the limit as β approaches ∞:

$$\lim_{\beta \to \infty} x\sigma(\beta x)$$

Thus, there are three conditions: x>0, x<0 and x=0. When $x > 0$, βx tends toward infinity, leading $\sigma(\beta x)$ to converge to 1. When $x < 0$, βx tends toward negetive infinity, leading $\sigma(\beta x)$ to converge to 0. finally, When $x = 0$, βx remains 0 regardless of β. As β tends towards infinity, the behavior of $g(x)$ approximates that of the ReLU function.

Q1.6) The swish function, which is continuously differentiable throughout, resolves the ReLU's problem of lacking differentiability at zero (X=0).
ReLU consistently behaves across its entire domain due to its strict monotonic nature, in contrast to the swish function which, without this strict monotonicity, might introduce unexpected bends (inflection points) in the function curve which can lead to undesirable properties during learning.

Note: I used Grammerly and ChatGPT to improve the quality of my writing.

Question 2 (4-4-5-3)**.** (**Convolution Basics**)

1. **Short answers:**
   (a) When you apply a filter to feature maps, under what conditions does the size of the feature map decrease ?
   (b) What impact does a larger stride have ?
   (c) Explain what you understand from the term "sparse connections" in terms of connections between two layers in a neural network.
   (d) As we are aware, each *Convolutional* layer contains learnable parameters, including weights and biases. If you decide not to include biases in the *Convolution* layer of the following network, what will be the impact on its performance (compared to the case when the *Convolution* layer includes biases) ?

$$Input \rightarrow Convolution \rightarrow BatchNorm \rightarrow Activation \rightarrow Output$$

2. **True/False:**
   (a) In a convolutional layer, the number of weights is contingent upon the depth of the input data volume.
   (b) In a convolutional layer, the number of biases equals the number of filters.
   (c) The total number of parameters is influenced by the stride and padding.
   (d) Maxpooling operates only on the height and width dimensions of an image.
3. **Convolutional Architecture:** Let's analyze the convolutional neural network (CNN) defined by the layers in the left column. For each layer, determine the output volume's shape and the number of parameters. It's worth noting that in our notation, "Conv4-N-Pvalid-S2" represents a convolutional layer with N 4x4 filters, valid padding, and a stride of 2.

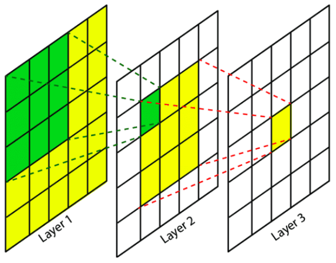

FIGURE 1 – Illustration of Receptive Field.

Layers	Output volume dimensions	Number of parameters
Input	$64 \times 64 \times 1$	
Conv4-5-Pvalid-S2		
Pool3		
Conv3-5-Pvalid-S1		
Pool2		
FC5		

4. **Receptive Fields:** Receptive Field is defined as the size of the region in the input that produces a feature. For example, when using a Conv-3x3 layer, each feature is generated by a 3x3 grid in the input. Having two Conv-3x3 layers in succession leads to a 5x5 receptive field. An example of this is outlined in Figure 1. Note that there is a receptive field associated with each feature (Figure 1 shows the receptive field associated with the yellow-colored feature in Layer 3), where a feature is a cell in the output activation map.

   Consider a network architecture consisting solely of 5 Convolution layers, each with kernel size 5, stride 1, zero-padding of two pixels on all four sides, 8 intermediate channels, and one output channel. What is the receptive field of the final features obtained through this architecture, that is, what is the size of the input that corresponds to a particular pixel position in the output ?

Answer 2. Q2.1.a) The size of the filter exceeds 1×1, and there's either no padding or it's inadequate to maintain the spatial measurements.

Q2.1.b) Reducing the spatial dimensions of the output feature map more drastically which can result in missing finer details from the input feature map.

Q2.1.c) A setup where not every neuron in one layer is connected to every neuron in the subsequent layer (in contrast to dense connections) which can reduce the number of parameters and potentially speed up training.

Q2.1.d) The network might be more sensitive to weight initialization (because we consider only origin) and when batch normalization is employed following the convolution, the lack of biases could have a lessened effect (because we can re-scale and re-center).

Q2.2.a) True

Q2.2.b) True

Q2.2.c) False

Q2.2.d) True

Q2.3) For the first row, because it is input layer the number of parameters equals to zero. For the second row, based on the given information (Conv4-5-Pvalid-S2), there's a convolutional layer equipped with five 4×4 filters, featuring valid padding and a stride of 2 and also the input size is $64 \times 64 \times 1$. For calculating the spatial size with valid padding: $\frac{W-F}{S} + 1$. W is the input width, F is the filter size, and S is the stride. So, output width and height is $\frac{64-4}{2} + 1 = 31$. Also there are 5 filters. So, output dimensions is: $31 \times 31 \times 5$. On the other side for number of parameters, we have five 4×4 filters that each has 1 bias, so in total is $1 \times (5 \times 16) + 5 = 85$. For the Pool3 layer which means 3x3 Pooling layer, the output width and height is: $\lfloor \frac{Pre-widthorheight}{windowsize} \rfloor = \lfloor \frac{31}{3} \rfloor = 10$. So, output dimensions is $10 \times 10 \times 5$. The number of parameters is 0 due to pooling layers don't have trainable parameters. Based on the given information (Conv3-5-Pvalid-S1), there's a convolutional layer equipped with five 3×3 filters, featuring valid padding and a stride of 1. So, output width and height is $\frac{10-3}{1} + 1 = 8$; Also there are 5 filters. So, output dimensions is: $8 \times 8 \times 5$. On the other side for number of parameters, we have five 3×3 filters that each has 1 bias, so in total is $5(5 \times 9) + 5 = 230$. For the Pool2 layer which means 2×2 Pooling layer, the output width and height is: $\lfloor \frac{8}{2} \rfloor = 4$. So, output dimensions is $4 \times 4 \times 5$. The number of parameters is 0 due to pooling layers don't have trainable parameters. For the FC5 layer, the output volume dimensions is 5, because it has no width or height. For the Number of parameters, the previous layer has an output size of $4 \times 4 \times 5 = 80$, and for each neuron, we have 80 weights plus 1 bias. So, in total: $5 \times 80 + 5 = 405$

Layers	Output volume dimensions	Number of parameters
Input	$64 \times 64 \times 1$	0
Conv4-5-Pvalid-S2	$31 \times 31 \times 5$	85
Pool3	$10 \times 10 \times 5$	0
Conv3-5-Pvalid-S1	$8 \times 8 \times 5$	230
Pool2	$4 \times 4 \times 5$	0
FC5	5	405

Q2.4) The expansion of the receptive field across several convolutional layers is described as:

$$receptivefield_{\text{newlayer}} = receptivefield_{\text{prevlayer}} + (k-1) \times s$$

Based on the information provided in the question, Initial receptive field (RFinit)=1, kernel size (K)=5 and Stride (S)=1.
- Receptive Field for the first layer:

$$RF_{\text{1layer}} = RF_{\text{init}} + (k-1) \times s$$

$$RF_{\text{1layer}} = 1 + (5-1) \times 1 = 5$$

- Receptive Field for the second layer:

$$RF_{\text{2layer}} = RF_{\text{1layer}} + (k-1) \times s$$

$$RF_{\text{2layer}} = 5 + (5-1) \times 1 = 9$$

- Receptive Field for the third layer:

$$RF_{\text{3layer}} = RF_{\text{2layer}} + (k-1) \times s$$

$$RF_{\text{3layer}} = 9 + (5-1) \times 1 = 13$$

- Receptive Field for the fourth layer:

$$RF_{\text{4layer}} = RF_{\text{3layer}} + (k-1) \times s$$

$$RF_{\text{4layer}} = 13 + (5-1) \times 1 = 17$$

- Receptive Field for the fifth layer:

$$RF_{\text{5layer}} = RF_{\text{4layer}} + (k-1) \times s$$

$$RF_{\text{5layer}} = 17 + (5-1) \times 1 = 21$$

Thus, the receptive field for the final features produced by this architecture is 21×21.

Note: I used Grammerly and ChatGPT to improve the quality of my writing.

Question 3 (10)**.** (**Mixture Models meet Neural Networks**)

Consider modelling some data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, $\boldsymbol{x}_n \in \mathbb{R}^d$, $y_n \in \{0, 1\}$, using a mixture of logistic regression models, where we model each binary label y_n by first picking one of the K logistic regression models, based on the value of a latent variable $z_n \sim \text{Categorical}(\pi_1, ..., \pi_K)$ and then generating y_n *conditioned* on z_n as $y_n \sim \text{Bernoulli}[\sigma(\boldsymbol{w}_{z_n}^T \boldsymbol{x}_n)]$, where $\sigma(\cdot)$ is the sigmoid activation function.

Now consider the *marginal* probability of the label $y_n = 1$, given $\boldsymbol{x}_n$, i.e., $p(y_n = 1|\boldsymbol{x}_n)$, and show that this quantity can also be thought of as the output of a neural network. Clearly specify what is the input layer, the hidden layer(s), activations, the output layer, and the connection weights of this neural network.

Answer 3. Q3.1) - ***Input Layer is:*** considering the question's assumptions:

$$\{(x_n, y_n)\}_{n=1}^N, \; x_n \in \mathbb{R}^d, \; y_n \in \{0, 1\}$$

the data point x_n acts as the Input Layer. With x_n belonging to $\mathbb{R}^d$, the input layer consists of d nodes.

- ***Hidden Layer is:*** considering the question's assumptions, choosing a latent variable z_n based on a categorical distribution given by $z_n \sim \text{Categorical}(\pi_1, \ldots, \pi_K)$. The latent variable z_n acts as the hidden layer, and being sourced from a categorical distribution with K categories, this layer comprises K nodes. Each node corresponds to a logistic regression model, and selecting a specific z_n is akin to triggering a particular neuron in the layer.

- ***Activation Layer is:*** considering the question's assumptions, y_n conditioned on:

$$z_n \sim \text{Bernoulli}(\sigma(w_{z_n}^T x_n))$$

. The outcomes of the logistic regression models represent the activations in the hidden layer. The activation corresponding to the k^{th} neuron (or the logistic regression model) is denoted as $\sigma(w_k^T x_n)$.

- ***Output Layer is:*** the final layer produces a binary outcome represented by y_n. This is conceptualized as a Bernoulli random variable. The likelihood $p(y_n = 1|x_n)$ can be perceived as an aggregated sum of the hidden layer's activations, with the aggregation influenced by the probabilities $\pi_1, \ldots, \pi_K$ sourced from the categorical distribution.

- ***Connection Weights are:*** for the input and hidden layer, each of the K logistic regression models is characterized by its respective weight, represented by the logistic regression parameter w_k. Also, for the hidden layer and output, the weights are determined by the probabilities $\pi_1, \ldots, \pi_K$ sourced from the categorical distribution. To determine the marginal probability $p(y_n = 1|x_n)$, we aggregate over every possible values of z_n:

$$p(y_n = 1|x_n) = \sum_{k=1}^{K} \pi_k \sigma(w_k^T x_n)$$

This resembles a neural network's mechanism where the resultant output is an accumulated sum of activations originating from the hidden layer.

Note: I used Grammerly and ChatGPT to improve the quality of my writing.

Question 4 (10)**.** (**Cross Entropy**)

Cross-entropy loss function (a popular loss function) is given by:

$$ce(p, x) = -x \log(p) - (1 - x) \log(1 - p)$$

This derivation assumes that x is binary, i.e. $x \in \{0, 1\}$. However, the same loss function is often also used with real-valued $x \in (0, 1)$.

1. Derive the cross-entropy loss function using the maximum likelihood principle for $x \in \{0, 1\}$.
2. Suggest a probabilistic interpretation of the cross-entropy loss function when $x \in (0, 1)$. (HINT: KL divergence between two distributions)

Answer 4. Q4.1) Given the data ($x \in \{0, 1\}$) and assumptions, the likelihood function can be described as:

$$L(p|x) = \begin{cases} p & \text{if } x = 1 \\ 1 - p & \text{if } x = 0 \end{cases}$$

In other words, it can be written as:

$$L(p|x) = p^x (1 - p)^{1-x}$$

.

In order to optimize the likelihood, we target maximizing the logarithm of the likelihood. The negative log likelihood, which is our objective to minimize, is expressed as:

$$-l(p|x) = -x \log(p) - (1 - x) \log(1 - p)$$

This formulation corresponds to the binary classification's cross-entropy loss function:

$$ce(p, x) = -x \log(p) - (1 - x) \log(1 - p)$$

Q4.2) Given the data ($x \in \{0, 1\}$) and Cross-entropy loss function:

$$ce(p, x) = -x \log(p) - (1 - x) \log(1 - p)$$

the main idea comes from the Kullback-Leibler (KL) divergence, which measures the difference between two probability distributions. For distributions p and q, the KL divergence is described as:

$$D_{KL}(p||q) = \sum_i p(i) \log\left(\frac{p(i)}{q(i)}\right)$$

When narrowed down to binary events, it's formulated as:

$$D_{KL}(x||p) = x \log\left(\frac{x}{p}\right) + (1 - x) \log\left(\frac{1 - x}{1 - p}\right)$$

The association between cross-entropy and KL divergence can be articulated as:

$$H(p,x) = D_{KL}(x||p) + H(x)$$

Here, $H(x)$ denotes the entropy of x, and given a specific x, it remains constant. As a consequence, the act of minimizing the cross-entropy in terms of p aligns with the goal of minimizing the KL divergence between x and p.

Note: I used Grammerly and ChatGPT to improve the quality of my writing.

Question 5 (1-4-2)**.** (**Optimization**)

Suppose we want to minimize $f(x,y) = y^2 + (y-x)^2$.

1. Find the true minimum.
2. For each step size given below, compute:
   (a) The value of (x1, y1) using the full gradient descent (not stochastic) at the starting point (x0, y0) = (1, 1).
   (b) The L2 distance of (x1, y1) from the true minimum.

   Please note that the (x1, y1) is (x, y) coordinate after 1 step of gradient descent.

   i. Step size $s = 0.01$.
   ii. Step size $s = 0.1$.
   iii. Step size $s = 10$.
3. What are the implications of using different step sizes in gradient descent, and what strategies can be employed to effectively address these implications ?

Answer 5. Q5.1) To identify the minimum of f, we must evaluate the gradient of f and equate it to zero.

The respective partial derivatives with respect to x and y are:

$$\frac{\partial f}{\partial x} = -2(y-x)$$

$$\frac{\partial f}{\partial y} = 2y + 2(y-x)$$

For f to attain a minimum, its gradient should nullify:

$$-2(y-x) = 0$$

$$2y + 2(y-x) = 0$$

Deducing from the initial equation, we get $y = x$. Incorporating this value into the subsequent equation yields:

$$2x + 2(x-x) = 0$$

consequently,

$$2x = 0$$

and:

$$x = 0$$

given this $y = x$, we also find $y = 0$.

Hence, the minimum of f resides at the coordinate (0,0).

Q5.2) Based on previous question, the gradients are: Given the function:

$$f(x, y) = y^2 + (y - x)^2$$

The gradients are:

$$\frac{\partial f}{\partial x} = -2(y - x)$$
$$\frac{\partial f}{\partial y} = 2y + 2(y - x)$$

Starting at the point $(x_0, y_0) = (1, 1)$:

$$\frac{\partial f}{\partial x} = -2(1 - 1) = 0$$
$$\frac{\partial f}{\partial y} = 2(1) + 2(1 - 1) = 2$$

For the full gradient descent update:

$$x_1 = x_0 - s \times \frac{\partial f}{\partial x}$$
$$y_1 = y_0 - s \times \frac{\partial f}{\partial y}$$

For $s = 0.01$:

$$x_1 = 1 - 0.01(0) = 1$$
$$y_1 = 1 - 0.01(2) = 0.98$$

For $s = 0.1$:

$$x_1 = 1 - 0.1(0) = 1$$
$$y_1 = 1 - 0.1(2) = 0.8$$

For $s = 10$:

$$x_1 = 1 - 10(0) = 1$$

$$y_1 = 1 - 10(2) = -19$$

Now, for the L_2 distance of (x_1, y_1) from the true (Global) minimum $(0, 0)$:

$$L_2 = \sqrt{(x_1 - 0)^2 + (y_1 - 0)^2}$$

For $s = 0.01$:

$$L_2 = \sqrt{(1-0)^2 + (0.98-0)^2} \approx 1.4$$

For $s = 0.1$:

$$L_2 = \sqrt{(1-0)^2 + (0.8-0)^2} \approx 1.28$$

For $s = 10$:

$$L_2 = \sqrt{(1-0)^2 + (-19-0)^2} \approx 19.02$$

Q5.3) The step size in optimization is pivotal. Small step sizes ensure stability but can converge slowly and risk of getting stuck in local minima and saddle points. Large steps might lead to rapid initial convergence but risk overshooting and even divergence. Several strategies modulate the learning rate during optimization. For example, the Step Decay method adjust the learning rate based on epochs. Adaptive algorithms like AdaGrad, RMSprop, and Adam tweak the rate using accumulated gradients. Additionally, to manage large gradients, especially in deep networks, gradient clipping sets a threshold to keep them in check.

Note: I used Grammerly and ChatGPT to improve the quality of my writing.