CS37300 Homework	— Homework 4—	Last update: November 25, 2025
Due: Nov 24 (Mon), 2025 (11:59 pm)		

Name: _____ PUID: _____

Instructions and Policy: Each student should write up their own solutions independently. You need to indicate the names of the people you discussed a problem with; ideally you should discuss with no more than two other people.

- YOU MUST INCLUDE YOUR NAME IN THE HOMEWORK
- The answers (without the python scripts) MUST be in submitted via Gradescope.
- The python scripts will be submitted separately via Gradescope.
- Please write clearly and concisely - clarity and brevity will be rewarded. Refer to known facts as necessary.
- Theoretical questions MUST include the intermediate steps to the final answer.
- For, the PDF report, mark the corresponding pages for each subproblem on Gradescope. Any incorrectly marked problem/ subproblem will not get any points.
- Zero points in any question where the python code answer doesn't match the answer on Gradescope.
- If the answer is a plot, it should be added to the PDF and, in the code, it should always be saved as an file (image or PDF), and not using plt.show(). You can use plt.savefig and then use plt.close().

1 Theoretical Questions (9+13+5+18=45 pts)

Please submit your answers on Gradescope.

Q0 (9 pts): Maximum Likelihood Estimation

The probability density function of a Gaussian distribution with parameters, mean μ and variance σ^2 is given by:

$$f\left(x;\mu,\sigma^2\right)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

Consider a set $\{X_1, X_2, \ldots., X_N\}$ of N independent and identically distributed Gaussian random variables, with parameters μ, σ.

1. **(5 pts)** Write the expression for the negative log-likelihood $\text{NLL}\left(X_1,\ldots,X_N\right)=-\log P\left(X_1, X_2, \ldots., X_N;\mu,\sigma^2\right)$ using the Gaussian density function given above.

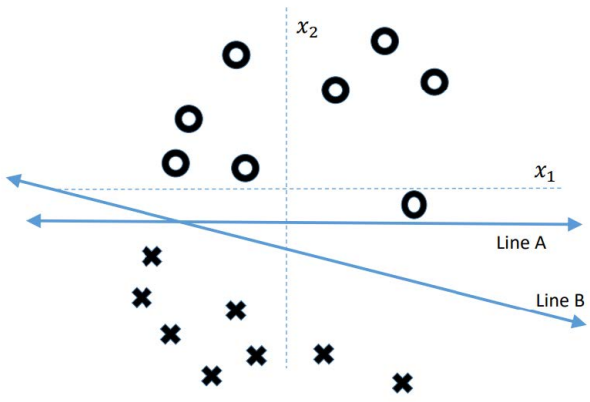
   $P(x_1,...,x_n;\mu,\sigma^2)=\prod f(x_i;\mu,\sigma^2)=(\frac{1}{\sqrt{2\pi\sigma}})^N\cdot(\sum\frac{-(x_i-\mu)^2}{2\sigma^2})$
   $Log((\frac{1}{\sqrt{2\pi\sigma}})^N\cdot(\sum\frac{-(x_i-\mu)^2}{2\sigma^2}))=-\frac{N}{2}Log(2\pi\sigma^2)+\sum\frac{-(x_i-\mu)^2}{2\sigma^2}$
   $=-\frac{N}{2}Log(2\pi\sigma^2)+\frac{1}{2\sigma^2}\sum(x_i-\mu)^2$

2. **(4 pts)** Compute the gradient of the negative log-likelihood with respect to μ and σ (it should be a function of $X_1, X_2, \ldots., X_N$). Include all derivation steps for full credits.
   Hint: There are two gradients that need to be computed: $\frac{\partial}{\partial\mu}\text{NLL}\left(X_1,\ldots,X_N\right)$ and $\frac{\partial}{\partial\sigma}\text{NLL}\left(X_1,\ldots,X_N\right)$.

$$\frac{\partial}{\partial \mu} NLL = \frac{1}{2\sigma^2} \sum 2(\mu - x_i) = \frac{1}{\sigma^2} \sum (\mu - x_i)$$
$$\frac{\partial}{\partial \sigma} NLL = \frac{N}{2} Log(2\pi) + NLog(\sigma) + \frac{1}{2\sigma^2} \sum (x_i - \mu)^2 = \frac{N}{\sigma} - \frac{1}{\sigma^3} \sum (x_i - \mu)^2$$

Q1 (13 pts): Logistic Regression

Let's think about what happens with linearly separable data in logistic regression. Consider the following training set:



Each data point has two features - one shown on the horizontal axis, and the second on the vertical axis. Each data point is in one of the two classes - 'X' is class 1, 'O' is class -1. In this dataset, there are 8 'X' samples and 8 'O' samples. Our model is logistic regression:

$$\Pr(y = 1 \mid x, w) = \frac{1}{1 + \exp(-w_0 - w_1 x[1] - w_2 x[2])}$$

We have shown two lines that might have been the result of our training set.

1. **(6 pts)** What are some possible values of w to produce line A and line B that separate the data

samples? Line A is horizontal with an intercept of -1. Consider also the separating Line B with slope of -1/2 and an intercept of -2 on the vertical axis. **Both line A and line B separate all the data points correctly.**

Line A: $w_0 + w_1 x_1 + w_2 x_2 = 0$
$x_2 = \frac{-w_0}{w_2} \Rightarrow w^T = [1, 0, 1]$
Line B: $\frac{1}{2} x_1 + x_2 + 2 = 0 \Rightarrow w^T = [2, 0.5, 1]$ scaled by 2 $w^T = [4, 1, 2]$

2. **(3 pts)** Show that for the $\mathbf{w} = [w_0, w_1, w_2]^T$ you found in part (a), that $2\mathbf{w}$ still corresponds to the same separating line (i.e., to the same lines A and B).

$w^T x = 0$, for $c > 0$
$\Rightarrow (cw)^T x = cw^T x = 0 \iff w^T x = 0$
scaling w does not change the boundary

3. **(4 pts)** The most common loss (or score) function for logistic regression is the negative log-likelihood (NLL). Using class labels $y \in \{-1, 1\}$ the NLL for the training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}$ is

$$\text{NLL}(\mathcal{D}) = \sum_{i=1}^{n} \log(1 + \exp(-y_i \mathbf{x}_i^T \mathbf{w})),$$

where $\mathbf{w} = [w_0, w_1, w_2]^T$ is a column vector. Considering line A in the figure at the beginning of the question. Show that the $\mathbf{w}$ we gave in part (1) for line A will not be the result that minimizes the NLL. Is this an issue for gradient descent model search? Why?

Hint: Think about part (2), where multiplying $\mathbf{w}$ by a scalar would always give smaller NLLs.

The w from part (1) cannot minimize the NLL becasue scaling w makes every term $Log(1 + exp(-y_i x_i^T w)$ smaller
so the NLL has no finite minimum for separable data
This is an issue for gradient descent because the weights grow toward infinity instead of converging.

Q2 (5 pts): True or False Questions

Answer the following as True or False with *a justification or an example*. Points are uniformly distributed wihtin the questions.

1. **(1 pts)** The perceptron algorithm can only converge if the data is linearly separable.

   True, otherwise it will run in an infinite loop.

2. **(1 pts)** Maximum Likelihood Estimation (MLE) maximizes the probability of observing the given data under the model.

   True what MLE does is that it finds most probable parameters under assummed statistical model.

3. **(1 pts)** In logistic regression, L2 regularization penalizes large weights and helps prevent overfitting.

True, L2 is a technique to prevent overfitting by adding a penalty

4. **(1 pts)** In logistic regression, the choice of the decision threshold for classification (e.g., >0.5 for positive instances) will affect the model's parameters and can impact its precision and recall.

False, changing the decision threshold affects only the predicted labels but it does not change the model learned parameters, which stay put after training.

5. **(1 pts)** If you have a convex loss function, and using the gradient descent algorithm does not result in finding the global minimum value of the loss function

False, for a convex loss function any point with zero gradient is a global minimum, so gradient descent will find it as long as all of the assumptions are valid.

Q3 (18 pts): Deep Learning/Activations

In this question, we will dive deeper into training ReLU-activated neurons (ReLU neurons).

Inner product layer (fully connected layer) As the name suggests, every output neuron of inner product layer is fully connected to the input neurons. The output is a vector resulting from the multiplication of the input with a weight matrix W plus a bias offset, i.e.:

$$\mathbf{z}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}, \tag{1}$$

where the input $\mathbf{x}$ is d dimensional column vector, and $\mathbf{W}$ is a $n \times d$ matrix and $\mathbf{b}$ is n dimensional column vector. This is just a linear transformation of the input. Often, we will concatenate the bias $\mathbf{b}$ into W by concatenating the constant 1 to $\mathbf{x}$. The weight parameter $\mathbf{W}$ and bias parameter $\mathbf{b}$ are learnable in this layer.

Activation. We add nonlinear activation functions after the inner product layers to model non-linearity (non-polynomial, to be more precise). Here are variants of the ReLU activation function, popular choices for non-polynomial activations:

- **ReLU**:
$$\text{relu}(z) = \max(0, z) \tag{2}$$
- **Leaky ReLU**:
$$\text{leaky\_relu}(z) = \begin{cases} z & \text{if } z \geq 0, \\ \alpha z & \text{if } z < 0, \end{cases} \tag{3}$$
- **ELU**:
$$\text{elu}(z) = \begin{cases} z & \text{if } z \geq 0, \\ \alpha(\exp(z) - 1) & \text{if } z < 0, \end{cases} \tag{4}$$
where α is a small constant.

If the input to the above functions is a vector $\mathbf{z}$*, then each function is applied to each element of the vector.*

Consider the negative log-likelihood as the score and gradient descent as the search algorithm used to find the optimal parameters of the neural network.

1. (**6 pts**) Compute the derivatives of the ReLU, leaky ReLU and ELU units with respect to variable z in equations (2), (3) and (4), respectively. (For points where the gradient is undefined, the derivative at that point may be designated as either the left-hand derivative or the right-hand derivative.)

$$relu(z) = max(z, 0), z > 0 \Rightarrow relu(z) = z \Rightarrow \frac{d}{dz}(relu(z)) = 1$$
$$relu(z) = max(z, 0), z < 0 \Rightarrow relu(z) = 0 \Rightarrow \frac{d}{dz}(relu(z)) = 0$$
$$\frac{d}{dz}(1) = \begin{cases} 0 & z < 0 \\ 1 & z > 0 \end{cases}$$
$$\frac{d}{dz}(2) = \begin{cases} 1 & z \geq 0 \\ \alpha & z < 0 \end{cases}$$
$$\frac{d}{dz}(3) = \begin{cases} 1 & z \geq 0 \\ \alpha e^z & z < 0 \end{cases}$$

2. (**6 pts**) Compute $\partial \mathbf{h}(\mathbf{x}) / \partial \mathbf{w}_j$, where $\mathbf{w}_j$ is the element j of the weight vector for

   (a) $\mathbf{h}(\mathbf{x}) = \text{relu}(\mathbf{w}^T \mathbf{x})$

   (b) $\mathbf{h}(\mathbf{x}) = \text{leaky\_relu}(\mathbf{w}^T \mathbf{x})$

   (c) $\mathbf{h}(\mathbf{x}) = \text{elu}(\mathbf{w}^T \mathbf{x})$

   (you will need to use the chain rule; see class notes).

   (For points where the gradient is undefined, the derivative at that point may be designated as either the left-hand derivative or the right-hand derivative.)

$$\frac{\partial h}{\partial w_j}(1) = \begin{cases} 0 & w^T x < 0 \\ x_j & w^T x > 0 \end{cases}$$

$$\frac{\partial h}{\partial w_j}(2) = \begin{cases} x_j & w^T x \geq 0 \\ \alpha x_j & w^T x < 0 \end{cases}$$

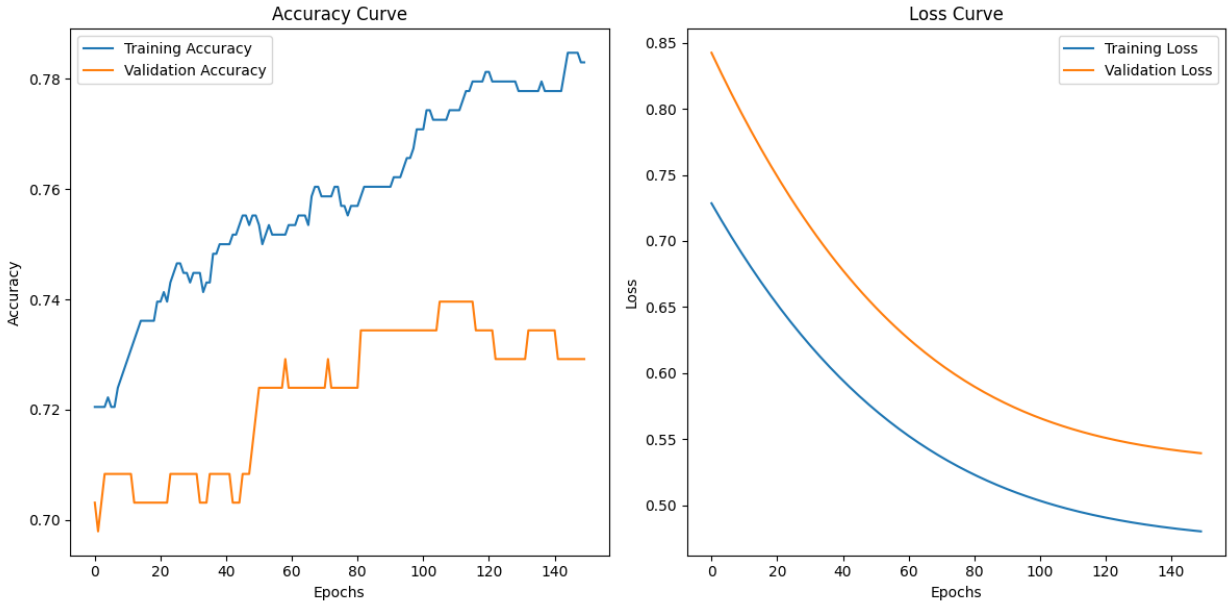
$$\frac{\partial h}{\partial w_j}(3) = \begin{cases} x_j & w^T x \geq 0 \\ x_j \alpha e^z & w^T x < 0 \end{cases}$$

3. (**4 pts**) We will now use ReLU gradient computed above to showcase a significant drawback of using ReLUs on neural networks. Recall that a ReLU **neuron** of a neural network is simply a logistic regression where we replace the sigmoid activation $\sigma()$ of the logistic by the ReLU activation. We have seen how the gradient of a weight vector $\mathbf{w}$ of a single logistic unit is computed by averaging the gradient computed over each training example $\{\mathbf{x}_i, y_i\}_{i=1}^n$. Clearly, if the resulting gradient of a parameter is zero, this parameter will not change in the next gradient step. Define the general mathematical condition relating $\{\mathbf{x}_i\}_{i=1}^n$ and $\mathbf{w}$ and b under which the gradient of a ReLU neuron $\mathbf{h}(\mathbf{x}) = \text{relu}(\mathbf{w}^T\mathbf{x} + \mathbf{b})$ is zero for all the inputs $\{\mathbf{x}_i\}_{i=1}^n$. When this happens, we say that the neuron has died.
Hint: Pay attention to the fact that the gradient of a parameter is the sum of the gradients over all training examples of the data.

Relu has 0 gradient with respect to w and b if it outputs 0 for all x_i
so, $w^T x_i + b \leq 0$ for all $i = 1, ..., n$
so, Relu(z) ≤ 0 for any training point.
so, partials go to zero and neuron "dies"

4. (**2 pts**) Can Leaky Relu and ELU neurons die like ReLU neurons? Explain using answers in 2.1 and 2.2.

Neither leaky relu or elu fully shutts of gradient when the input is nonpositive so they don't die like relu neurons. Look at the functions never zero!


Accuracy Curve
Training Accuracy
Validation Accuracy
Accuracy
Epochs
0.70
0.72
0.74
0.76
0.78
0
20
40
60
80
100
120
140
Loss Curve
Training Loss
Validation Loss
Loss
Epochs
0.50
0.55
0.60
0.65
0.70
0.75
0.80
0.85