Natural Language Processing (CO3086)
NLP 242 - Lab 8: Neural Network

HO CHI MINH UNIVERSITY OF TECHNOLOGY
Vietnam National University Ho Chi Minh

Problem 1

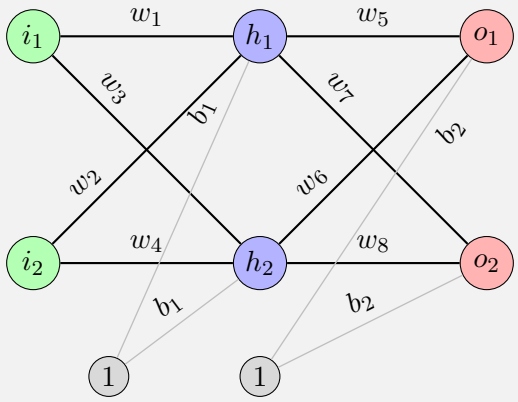
Consider a fully connected neural network with the following architecture:

- **Input Layer:** 2 neurons (i_1, i_2)
- **Hidden Layer:** 2 neurons (h_1, h_2) with sigmoid activation
- **Output Layer:** 2 neurons (o_1, o_2) with sigmoid activation
- **Bias terms:** b_1 for the hidden layer, b_2 for the output layer
- **Loss function:** Mean Squared Error (MSE), given by:

$$\text{MSE} = \frac{1}{2}\sum(o_{\text{expt}} - o_{\text{pred}})^2$$

- **Learning rate:** $\eta = 0.5$

The neural network is structured as follows: Given the following neural network



The given parameter values are:

$$[w_1, w_2, w_3, w_4, b_1, w_5, w_6, w_7, w_8, b_2] = [0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60]$$

The input and expected output values are:

$$[i_1, i_2] = [0.05, 0.10], \quad [o_1, o_2] = [0.01, 0.99]$$

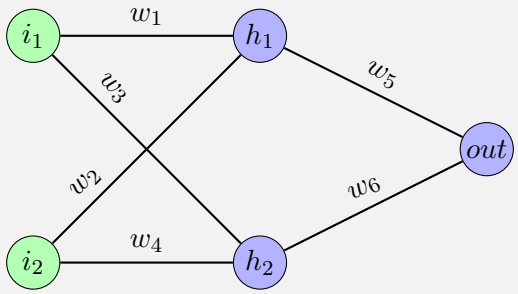
Perform one forward pass and one backward pass through the network to compute the updated weights and biases with detailing each step of the calculations.

Problem 2

Consider a fully connected neural network with the following architecture:

- **Input Layer:** 2 neurons (i_1, i_2)
- **Hidden Layer:** 2 neurons (h_1, h_2) without activation
- **Output Layer:** 1 neurons (out) without activation
- **Loss function:** Squared Error (SE), given by $\text{SE} = \frac{1}{2}(y - out)^2$.
- **Learning rate:** $\eta = 0.05$

The neural network is structured as follows: Given the following neural network



The given parameter values are:

$$[w_1, w_2, w_3, w_4, w_5, w_6] = [0.11, 0.21, 0.12, 0.08, 0.14, 0.15]$$

The input and expected output values are:

$$[i_1, i_2] = [2, 3], \quad [out] = [1]$$

Perform one forward pass and one backward pass through the network to compute the updated weights and biases with detailing each step of the calculations.

Problem 3

Softmax takes in an n-dimensional vector x and outputs another n-dimensional vector y:

$$y_i = \frac{e^{x_i}}{\sum_k e^{x_k}}$$

In this question we're going to compute the gradient of y with respect to x. Let

$$\delta_{ij} = \frac{\partial y_i}{\partial x_j}.$$

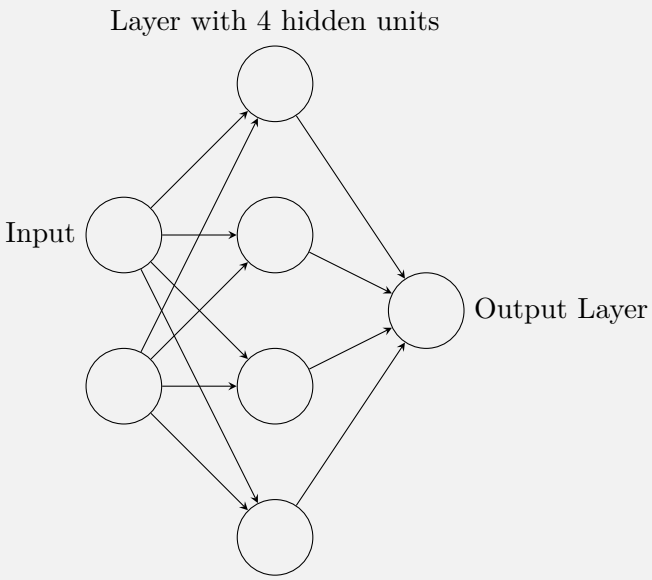
1. Derive an expression for δ_{ii}.
2. Now derive an expression for δ_{ij}, where $i \neq j$.

3. Write a general expression for δ_{ij}. Hint: Feel free to use curly brace notation to distinguish between the two cases where $i = j$ and $i \neq j$. For a concrete example, see how ELU was defined in the previous question.

Problem 4

You have a dataset where each example contains two features, x_1 and x_2, and a binary label. Here's a plot of the dataset:

You want to develop a model to perform binary classification. Suppose you're using a small neural network with the architecture shown below.



Below is a precise definition of the network. Note h is the activation function, $w_{i,j}$ denotes the jth weight in the ith hidden unit in the network, and $w_{output,j}$ denotes the jth weight in the output layer. The biases follow similar rules.

$$a_i = h(w_{i,1} * x_1 + w_{i,2} * x_2 + b_i)$$

$$\text{output} = f\left(\sum_{j=1}^{4}(a_j * w_{\text{output},j}) + b_{\text{output}}\right)$$

Note that h is the activation function for the hidden units and f is the activation function for the output layer. For all of these questions, f is defined as:

$$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

1. If you used the function $h(x) = c * x$ for some $c \in \mathbb{R}$, is it possible for this model to achieve perfect accuracy on this dataset? If so, provide a set of weights that achieves perfect accuracy. If not, briefly explain why.

2. Now assume h is a modified version of the sign function:

$$h(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

   Is it possible for this model to achieve perfect accuracy? If so, provide a set of weights that achieves perfect accuracy. If not, briefly explain why.

3. Recall the activation functions $\text{ReLU}(x) = \max(0, x)$. Consider an alternative to ReLU called Exponential Linear Unit (ELU).

$$\text{ELU}(x) = \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

   What is the gradient for ELU?

4. Name one advantage of using ELU over ReLU.

Problem 5

You want to build a model to predict whether a startup will raise funding (label = 1) or not (label = 0) in its first year. You have access to a dataset of examples with 300 input features, including the number of founders, the number of employees, the variance in the age of the employees, etc. As a baseline model, you decide to train a logistic regression that outputs a probability $\hat{y}^{(i)} \in [0, 1]$ that a startup described by a vector of features $x^{(i)}$ raises funding in its first year, where the predicted label of an input is chosen to be 1 when $\hat{y}^{(i)} \geq 0.5$ and 0 otherwise.

a) If the shape of $x^{(i)}$ is $(n_x, 1)$, what should n_x be?

b) A logistic regression has trainable weights w and bias b. How many weights does the logistic regression model have? What is the dimension of b?

c) Consider the prediction $\hat{y}^{(i)} \in [0, 1]$. What is the dimension of $\hat{y}^{(i)}$? Write the output $\hat{y}^{(i)}$ in terms of the input $x^{(i)}$ and the parameters w and b.

d) After training your baseline logistic regression model, you decide to train a single hidden layer neural network with 80 hidden neurons on the same dataset. How many weights and biases does this neural network have?

e) You decide to use ReLU as your hidden layer activation, and also insert a ReLU before the sigmoid activation such that

$$\hat{y} = \sigma(\text{ReLU}(z)),$$

where z is the pre-activation value for the output layer. What problem are you going to encounter?

Problem 6

You are building a classification model to distinguish between labels from a *synthetically* generated dataset. More specifically, you are given a dataset,

$$\{x^{(i)}, y^{(i)}\}_{i=1}^{m}, \quad \text{where } x^{(i)} \in \mathbb{R}^2, \quad y^{(i)} \in \{0, 1\}$$

The data is generated with the scheme:

$$X \mid Y = 0 \sim \mathcal{N}(2, 2)$$
$$X \mid Y = 1 \sim \mathcal{N}(0, 3)$$

You can assume the dataset is perfectly balanced between the two classes.

1. As a baseline, you decide to use a logistic regression model to fit the data. Since the data is synthesized easily, you can assume you have infinitely many samples (i.e., $m \to \infty$). Can your logistic regression model achieve 100% training accuracy? Explain your answer.

2. After training on a large training set of size M, your logistic regressor achieves a training accuracy of T. Can the following techniques, *applied individually*, improve over this *training accuracy*? Please justify your answer in a single sentence.

   (a) Adding a regularizing term to the binary cross-entropy loss function for the logistic regressor.

   (b) Standardizing all training samples to have *mean zero* and *unit variance*.

   (c) Using a 5-hidden layer feedforward network *without* non-linearities in place of logistic regression.

   (d) Using a 2-hidden layer feedforward network *with* ReLU in place of logistic regression.

Problem 7

The softmax function has the desirable property that it outputs a probability distribution, and is often used as an activation function in many classification neural networks. Consider a 2-layer neural network for K-class classification using softmax activation and cross-entropy

loss, as defined below:

$$\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$
$$\mathbf{a}^{[1]} = \text{LeakyReLU}(\mathbf{z}^{[1]}, \alpha = 0.01)$$
$$\mathbf{z}^{[2]} = W^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]}$$
$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z}^{[2]})$$
$$L = -\sum_{i=1}^{K} y_i \log(\hat{y}_i)$$

where the model is given input $\mathbf{x}$ of shape $D_x \times 1$, and one-hot encoded label $\mathbf{y} \in \{0,1\}^K$. Assume that the hidden layer has D_a nodes, i.e., $\mathbf{z}^{[1]}$ is a vector of size $D_a \times 1$. Recall the softmax function is computed as follows:

$$\hat{\mathbf{y}} = \left[\frac{\exp\left(z_1^{[2]}\right)}{Z}, \ldots, \frac{\exp\left(z_K^{[2]}\right)}{Z}\right]$$

where

$$Z = \sum_{j=1}^{K} \exp\left(z_j^{[2]}\right)$$

1) What are the shapes of $W^{[2]}, b^{[2]}$? If we were vectorizing across m examples, i.e., using a batch of samples $X \in \mathbb{R}^{D_x \times m}$ as input, what would be the shape of the output of the hidden layer?

2) What is $\partial \hat{y}_k / \partial z_k^{[2]}$? Simplify your answer in terms of element(s) of $\hat{\mathbf{y}}$.

3) What is $\frac{\partial \hat{y}_k}{\partial z_i^{[2]}}$, for $i \neq k$? Simplify your answer in terms of element(s) of $\hat{\mathbf{y}}$.

4) Assume that the label $\mathbf{y}$ has 1 at its k^{th} entry, and 0 elsewhere. What is $\frac{\partial L}{\partial z_i^{[2]}}$? Simplify your answer in terms of $\hat{\mathbf{y}}$. *Hint:* Consider both cases where $i = k$ and $i \neq k$.

5) What is $\frac{\partial z^{[2]}}{\partial a^{[1]}}$? Refer to this result as δ_1.

6) What is $\frac{\partial a^{[1]}}{\partial z^{[1]}}$? Refer to this result as δ_2. Feel free to use curly brace notation.

7) Denote $\frac{\partial L}{\partial z^{[2]}}$ with δ_0. What is $\frac{\partial L}{\partial W^{[1]}}$ and $\frac{\partial L}{\partial b^{[1]}}$? You can reuse notations from previous parts. *Hint:* Be careful with the shapes.

8) To avoid running into issues with numerical stability, one trick may be used when implementing the softmax function. Let $m = \max_{i=1}^{K} z_i$ be the max of z_i, then

$$\hat{y}_i = \frac{\exp\left(z_i^{[2]} - m\right)}{\sum_{j=1}^{K} \exp\left(z_j^{[2]} - m\right)}$$

What is the numerical problem with the initial softmax computation? Why the modified formula would help resolving that problem?

Problem 8

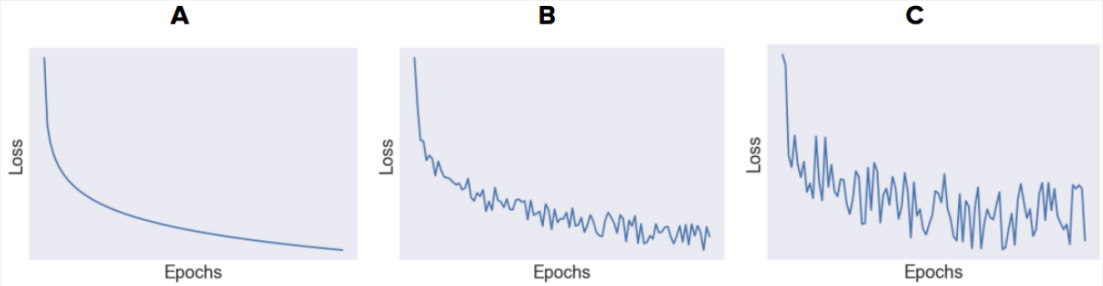
You want to build a classifier that predicts the musical instrument given an audio clip. There are 50 unique types of instruments, and you've collected a dataset of sounds of each, first collecting audio clips of guitars, then clips of violins, and so on for all the different instruments.

1. You use batch gradient descent (BGD) to optimize your loss function, but you have been getting poor training loss. You search your code for potential bugs and realize that you're not shuffling the training data. Would shuffling the training fix this problem? Explain your reasoning. (1-2 sentences)

2. You are deciding whether you should optimize your network parameters using mini-batch gradient descent (MBGD) or stochastic gradient descent (SGD) (i.e. batch size of 1). Give one reason to choose MBGD over SGD. (1-2 sentences)

3. Give one reason to use MBGD over BGD. (1-2 sentences)

4. Label the training loss curves A, B, and C with whether they were likely generated with SGD, MBGD, or BGD *(select one for each)*.



5. You decide to tune your model's learning rate. What is one typical sign of a learning rate being too large?

6. What is one typical sign of a learning rate being too small?

7. You now decide to use gradient descent with momentum. For gradient descent with momentum, write down the update rule for a particular trainable weight W. Use learning rate α and momentum hyperparameter β, and let J denote the cost function. (2 equations)

8. Explain how momentum speeds up learning compared to standard gradient descent. (1 sentence)