Natural Language Processing - CC01
(Math Exercise - Lab 8)

Le Thi Phuong Thao - 2252757

March 2025

1 Problem 1

1.1 Forward pass:

$$h_1 = \sigma(w_1 i_1 + w_2 i_2 + b_1)$$

$$h_2 = \sigma(w_3 i_1 + w_4 i_2 + b_1)$$

With

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

and

$$o_1, o2 = [0.01, 0.99]$$

We have: $h_1 = \sigma(0.3775) = 0.59327, h_2 = \sigma(0.3925) = 0.59688$

$$o_1 = \sigma(h_1 w_5 + h_2 w_6 + b_2)$$

$$o_2 = \sigma(h_1 w_7 + h_2 w_8 + b_2)$$

We have: $o_1 = 0.751365, o_2 = 0.772928$

1.2 Backward pass:

Compute loss:

$$L = \frac{1}{2}\sum(o_{pred} - o_{expt})^2 = \frac{1}{2}\left((o_1 - o_{1_{expt}})^2 + (o_2 - o_{2_{expt}})^2\right) = 0.2984$$

Differentiating w.r.t. o_1 and o_2:

$$\frac{\partial L}{\partial o_1} = (o_1 - o_{1expt}) = 0.7414$$

$$\frac{\partial L}{\partial o_2} = (o_2 - o_{2expt}) = -0.2171$$

Using sigmoid activation:

$$\frac{\partial o_1}{\partial z_{o_1}} = o_1(1 - o_1)$$

$$\frac{\partial o_2}{\partial z_{o_2}} = o_2(1 - o_2)$$

Thus:

$$\delta_{o_1} = (o_1 - o_{1expt}) \cdot o_1 \cdot (1 - o_1) = 0.1385$$

$$\delta_{o_2} = (o_2 - o_{2expt}) \cdot o_2 \cdot (1 - o_2) = -0.0381$$

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_5} = \delta_{o1} \cdot h_1 = 0.1385 \times 0.59327 = 0.0822$$

$$\frac{\partial L}{\partial w_6} = \frac{\partial L}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_6} = \delta_{o1} \cdot h_2 = 0.1385 \times 0.59688 = 0.082668$$

$$\frac{\partial L}{\partial w_7} = \frac{\partial L}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_7} = \delta_{o2} \cdot h_2 = -0.0381 \times 0.59327 = -0.0226$$

$$\frac{\partial L}{\partial w_8} = \frac{\partial L}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_8} = \delta_{o2} \cdot h_2 = -0.0381 \times 0.59688 = -0.02274$$

Hidden units:

$$\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial o_1} \cdot \frac{\partial o_1}{\partial h_1} + \frac{\partial L}{\partial o_2} \cdot \frac{\partial o_2}{\partial h_1} = 0.1385 \times 0.4 - 0.0381 \times 0.5 = 0.03635$$

$$\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial o_1} \cdot \frac{\partial o_1}{\partial h_2} + \frac{\partial L}{\partial o_2} \cdot \frac{\partial o_2}{\partial h_2} = 0.1385 \times 0.45 - 0.0381 \times 0.55 = 0.04137$$

We calculate gradient of hidden layer:

$$\delta_{h1} = (\delta_{o1} \cdot w_5 + \delta_{o2} \cdot w_7) \cdot h_1 \cdot (1 - h_1) = 0.03635 \times 0.5933 \times (1 - 0.5933) = 8.77 \times 10^{-3}$$

$$\delta_{h2} = (\delta_{o1} \cdot w_6 + \delta_{o2} \cdot w_8) \cdot h_2 \cdot (1 - h_2) = 0.04137 \times 0.59688 \times (1 - 0.59688) = 9.95 \times 10^{-3}$$

$$\frac{\partial L}{\partial w_1} = \delta_{h1} \cdot \frac{\partial h_1}{\partial w_1} = \delta_{h1} \cdot \delta_{i_1} = 8.77 \times 10^{-3} \times 0.05 = 4.385 \times 10^{-4}$$

$$\frac{\partial L}{\partial w_2} = \delta_{h1} \cdot \frac{\partial h_1}{\partial w_2} = \delta_{h1} \cdot \delta_{i_2} = 8.77 \times 10^{-3} \times 0.1 = 8.77 \times 10^{-4}$$

$$\frac{\partial L}{\partial w_3} = \delta_{h1} \cdot \frac{\partial h_2}{\partial w_1} = \delta_{h1} \cdot \delta_{i_1} = 9.95 \times 10^{-3} \times 0.05 = 4.975 \times 10^{-4}$$

$$\frac{\partial L}{\partial w_4} = \delta_{h_2} \cdot \frac{\partial h_2}{\partial w_4} = \delta_{h_2} \cdot \delta_{i_2} = 9.95 \times 10^{-3} \times 0.1 = 9.95 \times 10^{-4}$$

The update formula:

$$w_{new} = w_{old} - \eta \cdot \frac{\partial L}{\partial w}$$

$$b_{new} = b_{old} - \eta \cdot \frac{\partial L}{\partial b}$$

Therefore, we have:

$$w_1 = w_1 - \eta \cdot \frac{\partial L}{\partial w_1} = 0.15 - 0.5 \times (4.385 \times 10^{-4}) = 0.14978$$

$$w_2 = w_2 - \eta \cdot \frac{\partial L}{\partial w_2} = 0.20 - 0.5 \times (8.77 \times 10^{-4}) = 0.19956$$

$$w_3 = w_3 - \eta \cdot \frac{\partial L}{\partial w_3} = 0.25 - 0.5 \times 4.975 \times 10^{-4} = 0.24975$$

$$w_4 = w_4 - \eta \cdot \frac{\partial L}{\partial w_4} = 0.3 - 0.5 \times 9.95 \times 10^{-4} = 0.2995$$

$$w_5 = w_5 - \eta \cdot \frac{\partial L}{\partial w_5} = 0.4 - 0.5 \times (0.0822) = 0.3589$$

$$w_6 = w_6 - \eta \cdot \frac{\partial L}{\partial w_6} = 0.45 - 0.5 \times (0.082668) = 0.408666$$

$$w_7 = w_7 - \eta \cdot \frac{\partial L}{\partial w_7} = 0.5 - 0.5 \times -0.0226 = 0.5113$$

$$w_8 = w_8 - \eta \cdot \frac{\partial L}{\partial w_8} = 0.55 - 0.5 \times -0.02274 = 0.56137$$

$$b_1 = b_1 - \eta \cdot (\delta_{h1} + \delta_{h2}) = 0.35 - 0.5 \times (8.77 \times 10^{-3} + 9.95 \times 10^{-3}) = 0.34064$$

$$b_2 = b_2 - \eta \cdot (\delta_{o1} + \delta_{o2}) = 0.6 - 0.5 \times (0.1385 - 0.0381) = 0.5498$$

2 Problem 2

2.1 Forward Pass:

The given parameter values are:

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

$$h_1 = i_1 w_1 + i_2 w_2 = 2 \cdot 0.11 + 3 \cdot 0.21 = 0.85$$

$$h_2 = i_1 w_3 + i_2 w_4 = 2 \cdot 0.12 + 3 \cdot 0.08 = 0.48$$

$$out = w_5 h_1 + w_6 h_2 = 0.14 \cdot 0.85 + 0.15 \cdot 0.48 = 0.191$$

2.2 Backward Pass:

Compute loss function:

$$SE = \frac{1}{2}(y - out)^2 = 0.3272$$

Because output layer without activation:

$$\frac{\partial L}{\partial out} = out - y = 0.191 - 1 = -0.809$$

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial out}\frac{\partial out}{\partial w_5} = -0.809 \times 0.85 = -0.68765$$

$$\frac{\partial L}{\partial w_6} = \frac{\partial L}{\partial out}\frac{\partial out}{\partial w_6} = -0.809 \times 0.48 = -0.38832$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial out} \cdot \frac{\partial out}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1} = -0.809 \times 0.14 \times 2 = -0.22652$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial out} \cdot \frac{\partial out}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_2} = -0.809 \times 0.14 \times 3 = -0.33978$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial out} \cdot \frac{\partial out}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_3} = -0.809 \times 0.15 \times 2 = -0.2427$$

$$\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial out} \cdot \frac{\partial out}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_4} = -0.809 \times 0.15 \times 3 = -0.36405$$

Update weights:

$$w_1 = w_1 - \eta \cdot \frac{\partial L}{\partial w_1} = 0.11 - 0.05 \times -0.22652 = 0.121326$$

$$w_2 = w_2 - \eta \cdot \frac{\partial L}{\partial w_2} = 0.21 - 0.05 \times -0.33978 = 0.226989$$

$$w_3 = w_3 - \eta \cdot \frac{\partial L}{\partial w_3} = 0.12 - 0.05 \times -0.2427 = 0.132135$$

$$w_4 = w_4 - \eta \cdot \frac{\partial L}{\partial w_4} = 0.08 - 0.05 \times -0.36405 = 0.09820$$
$$w_5 = w_5 - \eta \cdot \frac{\partial L}{\partial w_5} = 0.14 - 0.05 \times -0.68765 = 0.17438$$
$$w_6 = w_6 - \eta \cdot \frac{\partial L}{\partial w_1} = 0.15 - 0.05 \times -0.38832 = 0.169416$$

3 Problem 3

3.1 Derive an expression for δ_{ii}

$$\delta_{ii} = \frac{\partial y_i}{\partial x_i} = \frac{e^{x_i} \sum_k e^{x_k} - e^{2x_i}}{(\sum_k e^{x_k})^2}$$

$$= \frac{e^{x_i}(\sum_k e^{x_k} - e^{x_i})}{\left(\sum_k e^{x_k}\right)^2}$$

$$= \frac{e^{x_i}}{(\sum_k e^{x_k})} \cdot \frac{(\sum_k e^{x_k} - e^{x_i})}{(\sum_k e^{x_k})}$$

Suppose:

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

We have:

$$= y_i(1 - y_i)$$

Thus,

$$\delta_{ii} = y_i(1 - y_i)$$

3.2 Derive an expression for δ_{ii}, where $i = j$

$$\delta_{ij} = \frac{\partial y_i}{\partial x_j} = \frac{-e^{x_i} e^{x_j}}{(\sum_k e^{x_k})^2}$$

Suppose:

$$\frac{e^{x_n}}{(\sum_k e^{x_k})} = y_n$$

We have:

$$\delta_{ij} = -y_i y_j$$

3.3 Write a general expression for δ_{ii}

From 3.1 and 3.2, a general expression for δ_{ii}:

$$\delta_{ij} = \begin{cases} y_i(1 - y_i), & if \quad i = j \\ -y_i y_j, & if \quad i \neq j \end{cases}$$

4 Problem 4

4.1

If we used the function $h(x) = c \cdot x$ for some $c \in R$, the model can only learn linearly separable decision boundaries. If the dataset is complex and non-linearly separable, a linear activation function will fail to capture the patterns, leading to poor accuracy.
Therefore, it is not possible for this model to achieve perfect accuracy on this dataset.

4.2

Yes, it is possible for this model to achieve perfect accuracy on the dataset. The function $h(x)$ can separate data points into 2 classes (0 and 1), each hidden layer node define a boundary of a square decision region.
The output layer checks if all conditions are satisfied or not to make the final classification. For example:
We need to create four boundary lines to define a rectangle using four neurons in the hidden layer:

$$w_1 = (0, -1), \quad b_1 = 2 \quad \Rightarrow \quad y \geq 2$$

$$w_2 = (-1, 0), \quad b_2 = 2 \quad \Rightarrow \quad x \geq 2$$

$$w_3 = (0, 1), \quad b_3 = -1 \quad \Rightarrow \quad y \leq 1$$

$$w_4 = (1, 0), \quad b_4 = -1 \quad \Rightarrow \quad x \leq 1$$

The output neuron needs to check if all four conditions are met simultaneously. The weights for the output neuron are:

$$w_{output} = (1, 1, 1, 1), \quad b_{output} = -4$$

This means the output neuron computes:

$$\sum w_{output} \cdot h_{hidden} + b_{output} = 0$$

Therefore, $h(x) = 1$, correctly classifying the point as inside the rectangle.

4.3 The gradient for ELU

Consider an alternative to ReLU called Exponential Linear Unit (ELU):

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

The gradient for ELU:

$$\begin{cases} 1, & x \geq 0 \\ \alpha e^x, & x < 0 \end{cases}$$

4.4 Name one advantage of using ELU over ReLU

ELU (Exponential Linear Unit) has an advantage over ReLU because it handles negative values more smoothly, reducing the dying neurons problem and improving convergence speed.

5 Problem 5

5.1 a)

n_x is 300-dimensional feature vector.

5.2 b)

The logistic regression model have 300 weights. The dimension of b is 1 $\times$ 1.

5.3 c)

The dimension of $\hat{y}^{(i)}$ is 1 $\times$ 1.
The output $y(i)$ in terms of the input $x(i)$ and the parameters w and b:

$$y^{(i)} = wx^{(i)} + b$$

5.4 d)

Weight Parameters: 300 $\times$ 80 + 80 $\times$ 1 = 24080
Bias Parameters: 80 + 1 = 81
Therefore, this neural network need 24080 weights and 81 biases.

5.5 e)

Using ReLU before the sigmoid activation in the output layer can cause a vanishing gradient issue because ReLU(z) sets all negative values to zero. If z is negative, the gradient through the ReLU will be zero, preventing weight updates during backpropagation and potentially leading to dead neurons where the model stops learning for certain inputs.

6 Problem 6

6.1

Since the data for each class is normally distributed and both distributions have overlapping tails (N(2,2) and N(0,3)), there will always be some probability of misclassification. Because logistic regression assumes a linear decision boundary, it is not possible to achieve 100% training accuracy.

6.2

6.2.1 a)

No, regularization typically reduces overfitting rather than improve training accuracy, as it penalizes large weights and can lead to a slightly lower training accuracy.

6.2.2 b)

No, standardization helps with optimization but does not directly increase training accuracy.

6.2.3 c)

No, without non-linear activation functions, the deep network is still a linear model, which is equivalent to logistic regression and thus does not improve accuracy.

6.2.4 d)

Yes, a 2-hidden layer network with ReLU can improve accuracy over logistic regression because it can learn non-linear decision boundaries, capture more complex patterns in the data and avoids the limitations of a purely linear model like logistic regression.

7 Problem 7

7.1

$W^{[2]}$ transforms the hidden layer output $a^{[1]}$ (which has dimension $D_a \times 1$) into the output logits $z^{[2]}$ (which has dimension $K \times 1$).

Since matrix multiplication follows the rule:

$$(K \times D_a) \times (D_a \times 1) = (K \times 1)$$

the weight matrix must have shape

$$K \times D_a$$

Shape of $b^{[2]}$ is added to each output neuron, so it must match the dimensions of $z^{[2]}$, which is $K \times 1$ Thus:

$$W^{[2]} : K \times D_a, \quad b^{[2]} : K \times 1$$

If we were vectorizing across m examples, i.e., using a batch of samples $X \in R$ Dxm as input, the shape of hidden layer would be $D_a \times m$

7.2

To find the derivative of $\frac{\partial \hat{y}_k}{\partial z_k^{[2]}}$, we differentiate the softmax function:

$$\frac{\partial \hat{y}_k}{\partial z_k^{[2]}} = \frac{\partial}{\partial z_k^{[2]}} \left(\frac{e^{z_k^{[2]}}}{\sum_{j=1}^K e^{z_j^{[2]}}} \right)$$

Using the quotient rule:

$$\frac{\partial \hat{y}_k}{\partial z_k^{[2]}} = \frac{e^{z_k^{[2]}} \sum_{j=1}^K e^{z_j^{[2]}} - e^{z_k^{[2]}} e^{z_k^{[2]}}}{\left(\sum_{j=1}^K e^{z_j^{[2]}} \right)^2}$$

Suppose: $\hat{y}_k = \frac{e^{z_k^{[2]}}}{\sum_{j=1}^K e^{z_j^{[2]}}}$, we simplify:

$$\frac{\partial \hat{y}_k}{\partial z_k^{[2]}} = \hat{y}_k(1 - \hat{y}_k)$$

Therefore, we have:

$$\frac{\partial \hat{y}_k}{\partial z_k^{[2]}} = \hat{y}_k(1 - \hat{y}_k)$$

7.3

Similar to question 3.3, using the quotient rule, differentiating $\hat{y}_k$ w.r.t. $z_i^{[2]}$ $(i \neq k)$:

$$\frac{\partial \hat{y}_k}{\partial z_i^{[2]}} = \frac{e^{z_k^{[2]}} \cdot (-e^{z_i^{[2]}})}{\left(\sum_{j=1}^K e^{z_j^{[2]}} \right)^2}$$

Suppose that:

$$\hat{y}_k = \frac{e^{z_k^{[2]}}}{\sum_{j=1}^K e^{z_j^{[2]}}}, \quad \hat{y}_i = \frac{e^{z_i^{[2]}}}{\sum_{j=1}^K e^{z_j^{[2]}}}$$

Therefore, we have:

$$\frac{\partial \hat{y}_k}{\partial z_i^{[2]}} = -\hat{y}_k \hat{y}_i, \quad fori \neq k$$

7.4

$$\frac{\partial L}{\partial z_i^{[2]}} = \begin{array}{ll} \hat{y}_i - 1, & for i = k \\ \hat{y}_i, & for i \neq k \end{array}$$

7.5

Since $z^{[2]}$ is defined as:

$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$

taking the derivative with respect to $a^{[1]}$:

$$\frac{\partial z^{[2]}}{\partial a^{[1]}} = W^{[2]}$$

7.6

We are given that the activation function for the hidden layer is Leaky ReLU:

$$a^{[1]} = LeakyReLU(z^{[1]}, \alpha = 0.01)$$

where Leaky ReLU is defined as:

$$LeakyReLU(z) = \begin{array}{ll} z, & if z \geq 0 \\ \alpha z, & if z < 0 \end{array}$$

where $\alpha = 0.01$ is the negative slope.
Since Leaky ReLU is applied element-wise, the derivative is:

$$\frac{\partial a_j^{[1]}}{\partial z_j^{[1]}} = \begin{array}{lll} 1, & if & z_j^{[1]} \geq 0 \\ \alpha, & if & z_j^{[1]} < 0 \end{array}$$

- When $z_j^{[1]}$ is non-negative, the gradient is 1.
- When $z_j^{[1]}$ is negative, the gradient is $\alpha = 0.01$.

7.7

Using the chain rule, the gradients propagate as:

$$\frac{\partial L}{\partial W^{[1]}} = \frac{\partial L}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}} \cdot \frac{\partial z^{[1]}}{\partial W^{[1]}}$$

$$\frac{\partial L}{\partial b^{[1]}} = \frac{\partial L}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}}$$

We have:

$$\frac{\partial L}{\partial W^{[1]}} = (W^{[2]})^T (\hat{y} - y) \cdot D^{[1]} \cdot X^T$$

$$\frac{\partial L}{\partial b^{[1]}} = (W^{[2]})^T (\hat{y} - y) \cdot D^{[1]}$$

7.8

The problem with the initial softmax function is that it can lead to numerical overflow (for large logits) or underflow (for very small logits).
The modified formula subtracts the maximum logit m, keeping the exponentiated terms numerically stable while preserving the correct probability distribution.

8 Problem 8

8.1

No, shuffling the dataset will not impact the gradients in Batch Gradient Descent (BGD). Because, in Batch Gradient Descent (BGD), the entire dataset is used for each update step. Since BGD computes the gradient using all training samples in every iteration, the order of data points does not affect the computed gradients.

8.2

MBGD provides a good balance between computational efficiency and gradient noise reduction, because it updates parameters using a small batch of samples rather than a single sample, leading to faster convergence and more stable updates compared to the high variance of SGD.

8.3

Mini-Batch Gradient Descent (MBGD) is preferred over Batch Gradient Descent (BGD) because it reduces memory usage by processing smaller batches instead of the entire dataset at once, making it feasible for large datasets

8.4

Batch (A), minibatch (B), stochastic (C).

8.5

Cost function does not converge to an optimal solution and can even diverge.

8.6

Cost function may take a very long time to converge. The cost function decreases very slowly.

8.7

$$V = \beta V + (1 - \beta)\frac{\partial L}{\partial W}$$

$$W = W - \alpha V$$

where β is the momentum coefficient and α is the learning rate.

8.8

Momentum speeds up learning by accumulating past gradients to smooth updates, reducing oscillations, and accelerating convergence in the correct direction.