AY: 2024-2025
Exam | **AI-ECUE122**
Jan. 2025

M1-S1: Dept. of Electrical Engineering
Teacher: A. Mhamdi
Time Limit: $1\frac{1}{2}$ h

This document contains **8** pages numbered from **1/8** to **8/8**. As soon as it is handed over to you, make sure it is complete. The **2** tasks are independent and can be treated in the order that suits you.

The following rules apply:

1. **A handwritten double-sided A4** sheet is permitted.
2. **Any electronic material**, except basic calculator, is prohibited.
3. **Mysterious or unsupported answers** will not receive full credit.
4. **Round results** to the nearest thousandth *(i.e., third digit after the decimal point)*.
5. **Task Nº2:** Each correct answer will grant a mark with no negative scoring.

Task Nº1 ⌛ 45mn | (7 points)

You are given a neural network with the following structure:

Network Architecture

Input layer: 2 neurons (x_1 and x_2),

Hidden layer: 3 neurons ($a_1^{[1]}$, $a_2^{[1]}$, $a_3^{[1]}$),

Output layer: 1 neuron (y).

Hidden layer: Sigmoid activation,

Output layer: Linear activation.

Training Data

Inputs: $x_1 = 0.5$, $x_2 = 0.1$

Target output: $y = 0.4$

$$\mathcal{W}^{[1]} = \begin{bmatrix} 0.2 & 0.1 \\ -0.3 & 0.5 \\ 0.4 & -0.4 \end{bmatrix} \quad \text{and} \quad \mathcal{W}^{[2]} = \begin{bmatrix} 0.3 & -0.2 & 0.1 \end{bmatrix}$$

$$b^{[1]} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix} \quad \text{and} \quad b^{[2]} = 0.05$$

(a) (1 point) Compute the pre-activation (z) and activation (a) for each hidden neuron, and the output $\hat{y}$.

$$z_1^{[1]} = w_{11}^{[1]}x_1 + w_{12}^{[1]}x_2 + b_1^{[1]} \quad \therefore \quad a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_{21}^{[1]}x_1 + w_{22}^{[1]}x_2 + b_2^{[1]} \quad \therefore \quad a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_{31}^{[1]}x_1 + w_{32}^{[1]}x_2 + b_3^{[1]} \quad \therefore \quad a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z^{[2]} = w_1^{[2]}a_1^{[1]} + w_2^{[2]}a_2^{[1]} + w_3^{[2]}a_3^{[1]} + b^{[2]} \quad \therefore \quad a^{[2]} = z^{[2]}$$

Substitute values:

$$z_1^{[1]} = (0.2)(0.5) + (0.1)(0.1) + 0.1 = 0.21 \quad \therefore \quad a_1^{[1]} = \sigma(0.21) \approx 0.552$$

$$z_2^{[1]} = (-0.3)(0.5) + (0.5)(0.1) + 0.1 \approx 0 \quad \therefore \quad a_2^{[1]} \approx \sigma(0) = 0.5$$

$$z_3^{[1]} = (0.4)(0.5) + (-0.4)(0.1) + 0.1 = 0.26 \quad \therefore \quad a_3^{[1]} = \sigma(0.26) \approx 0.565$$

The output is given by:

$$\hat{y} = w_1^{[2]}a_1^{[1]} + w_2^{[2]}a_2^{[1]} + w_3^{[2]}a_3^{[1]} + b^{[2]}$$

$$\hat{y} = (0.3)(0.552) + (-0.2)(0.5) + (0.1)(0.565) + 0.05$$

$$\hat{y} \approx 0.172$$

(b) (1 point) Compute the loss using the Mean Squared Error (MSE):

$$\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2$$

The loss is given by the Mean Squared Error (MSE):

$$\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2$$

Substitute values:

$$\mathcal{L} = \frac{1}{2}(0.172 - 0.4)^2 = \frac{1}{2}(-0.228)^2 \approx 0.026$$

(c) (3 points) Perform backpropagation to compute gradients of the loss $\mathcal{L}$ with respect to the weights and biases.

Gradients w.r.t. weights and bias at the output layer:

$$\frac{\partial \mathcal{L}}{\partial w_i^{[2]}}, \quad \frac{\partial \mathcal{L}}{\partial b^{[2]}} \qquad \forall i = 1, 2, 3$$

$$\frac{\partial \mathcal{L}}{\partial w_1^{[2]}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_1^{[2]}} = \text{-}0.228 \cdot a_1^{[1]}$$

$$\frac{\partial \mathcal{L}}{\partial w_1^{[2]}} = \text{-}0.228 \cdot 0.552 \approx \text{-}0.126$$

Similarly:

$$\frac{\partial \mathcal{L}}{\partial w_2^{[2]}} = \text{-}0.228 \cdot 0.5 \approx \text{-}0.114$$

$$\frac{\partial \mathcal{L}}{\partial w_3^{[2]}} = \text{-}0.228 \cdot 0.565 \approx \text{-}0.129$$

$$\frac{\partial \mathcal{L}}{\partial b^{[2]}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} = \text{-}0.228$$

Gradients w.r.t. weights and bias at the hidden layer:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{[1]}} \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial b_i^{[1]}} \qquad \forall i = 1, 2, 3, \; j = 1, 2$$

$$\frac{\partial L}{\partial a_1^{[1]}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_1^{[1]}} = \text{-}0.228 \cdot w_1^{[2]}$$

$$\frac{\partial \mathcal{L}}{\partial a_1^{[1]}} = \text{-}0.228 \cdot 0.3 = \text{-}0.0684$$

Similarly:

$$\frac{\partial \mathcal{L}}{\partial a_2^{[1]}} = \text{-}0.228 \cdot (\text{-}0.2) = 0.0456$$

$$\frac{\partial \mathcal{L}}{\partial a_3^{[1]}} = \text{-}0.228 \cdot 0.1 = \text{-}0.0228$$

For $w_{11}^{[1]}$:

$$\frac{\partial \mathcal{L}}{\partial w_{11}^{[1]}} = \frac{\partial \mathcal{L}}{\partial a_1^{[1]}} \cdot \frac{\partial a_1^{[1]}}{\partial z_1^{[1]}} \cdot \frac{\partial z_1^{[1]}}{\partial w_{11}^{[1]}}$$

The derivative of the sigmoid function is:

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

$$\frac{\partial a_1^{[1]}}{\partial z_1^{[1]}} = a_1^{[1]}(1-a_1^{[1]}) = 0.552 \cdot (1-0.552) \approx 0.247$$

$$\frac{\partial \mathcal{L}}{\partial w_{11}^{[1]}} = -0.0684 \cdot 0.247 \cdot x_1 = -0.0684 \cdot 0.247 \cdot 0.5 \approx -0.00845$$

Compute similarly for other weights in $\mathcal{W}^{[1]}$ and biases $b^{[1]}$.

(d) (2 points) Update the weights and biases using gradient descent with $\eta = 0.1$

$$\mathcal{W} \leftarrow \mathcal{W} - \eta \cdot \nabla_{\mathcal{W}} \mathcal{L} \quad \text{and} \quad b \leftarrow b - \eta \cdot \nabla_b \mathcal{L}$$

For $w_1^{[2]}$:

$$w_1^{[2]} = 0.3 - 0.1 \cdot (-0.126) = 0.3126$$

Similarly, update all weights and biases.

AY: 2024-2025 | Full Name: ....
M1-S1: Dept. of Electrical Engineering | ID: ....
Exam | AI-ECUE122 | Class: ....
Jan. 2025 | Room: ....
Teacher: A. Mhamdi | Time Limit: $1\frac{1}{2}$ h

✂- -

Answer Sheet

Task N°2 ⌛ 35mn | (13 points)

(a) ($\frac{1}{2}$ point) In fuzzy logic, what is the term for the process of converting crisp values into fuzzy sets?

√ Fuzzification ◯ Defuzzification ◯ Aggregation ◯ Inference

(b) ($\frac{1}{2}$ point) What is the primary purpose of the membership function in fuzzy logic?

- ◯ To perform defuzzification
- ◯ To combine multiple rules
- √ To indicate the degree of belonging to a fuzzy set
- ◯ To generate random values

(c) ($\frac{1}{2}$ point) Which method is commonly used for defuzzification in fuzzy logic systems?

- ◯ Max-Min composition
- ◯ Union operation
- √ Center of Gravity
- ◯ Fuzzy intersection

(d) ($\frac{1}{2}$ point) Which of these is a characteristic of fuzzy logic systems?

- ◯ They only work with binary values
- ◯ They require exact mathematical models
- ◯ They always produce crisp outputs without defuzzification
- √ They can handle imprecise or vague information

(e) ($\frac{1}{2}$ point) What is the main difference between Mamdani and Sugeno fuzzy models?

- √ The way they define the consequent part
- ◯ The number of rules they can handle
- ◯ The number of inputs they can process
- ◯ The fuzzification process

Do not write anything here

(f) (1/2 point) In the Takagi-Sugeno model, the consequent part is:

- ○ A fuzzy set
- √ A mathematical function of the input variables
- ○ A constant value only
- ○ A linguistic variable

(g) (1/2 point) The Mamdani fuzzy inference system requires:

- √ Both fuzzification and defuzzification
- ○ No defuzzification
- ○ Only fuzzification
- ○ Neither fuzzification nor defuzzification

(h) (1/2 point) The Mamdani method is most suitable for:

- ○ Linear control systems only
- ○ Mathematical analysis
- √ Expert knowledge-based systems
- ○ High-speed computations

(i) (1/2 point) In Takagi-Sugeno systems, the final output is obtained by:

- ○ Center of gravity defuzzification
- ○ Max-min composition
- ○ Mean of maxima method
- √ Weighted average of rule outputs

(j) (1/2 point) The Mamdani model's output is:

- √ A fuzzy set requiring defuzzification
- ○ Always a linear function
- ○ A monotonic function
- ○ A crisp number without defuzzification

(k) (1/2 point) Which of the following is **NOT** a common activation function in neural networks?

○ ReLU ○ Sigmoid √ Cubic Root ○ Tanh

(l) (1/2 point) In a neural network, what does backpropagation primarily do?

- ○ Forward pass calculation

Do not write anything here

✂ -

◯ Input normalization

√ Calculates gradients for weight updates

◯ Data preprocessing

(m) ($^1\!/_2$ point) Which loss function would you use for regression problems?

◯ Cross-Entropy ◯ Hinge Loss √ Mean Squared Error ◯ Binary Cross-Entropy

(n) ($^1\!/_2$ point) During backpropagation, gradients are computed:

◯ In forward direction

◯ Only for the output layer

◯ Only for hidden layers

√ In backward direction from output to input

(o) ($^1\!/_2$ point) Which optimizer uses momentum and adaptive learning rates?

◯ SGD ◯ RMSprop ◯ Adagrad √ Adam

(p) ($^1\!/_2$ point) What is the purpose of the learning rate in gradient descent?

◯ To normalize input data

◯ To initialize network weights

√ To control the size of weight updates

◯ To determine batch size

(q) ($^1\!/_2$ point) What's the advantage of using mini-batch gradient descent?

◯ It uses less memory

√ It balances computation speed and gradient accuracy

◯ It always converges to global minimum

◯ It eliminates the need for epochs

(r) ($^1\!/_2$ point) What's the purpose of the `Softmax` activation function?

√ To convert outputs to probabilities

◯ To introduce non-linearity

◯ To prevent overfitting

◯ To normalize inputs

(s) ($^1\!/_2$ point) How do you create an array of zeros in `Julia`?

√ `zeros(3,3)` ◯ `new_zeros(3,3)` ◯ `array(0,3,3)` ◯ `create_zeros(3,3)`

DO NOT WRITE ANYTHING HERE

✂- -

(t) ($^1/_2$ point) In Julia, what is the correct way to define a custom layer CustomLayer?

○ `class CustomLayer` ○ `def CustomLayer` √ `struct CustomLayer`

(u) ($^1/_2$ point) In Julia, what is the correct way to perform matrix multiplication?

○ `A x B` √ `A * B` ○ `A .* B` ○ `multiply(A,B)`

(v) ($^1/_2$ point) In Julia, what is the correct way to define a function?

○ `void function_name(x)`

○ `def function_name(x)`

√ `function function_name(x)`

○ `func function_name(x)`

(w) ($^1/_2$ point) How do you declare a neural network using Flux.jl?

○ `NeuralNetwork([Dense(10, 5)])`

√ `Chain(Dense(10, 5))`

○ `Network([Dense(10, 5)])`

○ `Sequential(Dense(10, 5))`

ANSWERS

(x) ($^1/_2$ point) In Julia's Flux.jl, how do you define a custom loss function?

○ `new\_loss(x, y) = ...`

○ `function loss = ...`

○ `define_loss(x, y) = ...`

√ `loss(x, y) = sum((model(x) .- y).^2)`

(y) ($^1/_2$ point) In Julia's Flux.jl, how do you compute gradients?

○ `backward(loss)`

○ `compute_gradients(model)`

√ `gradient(() -> loss(x, y), params(model))`

○ `backprop(model, loss)`

(z) ($^1/_2$ point) How do you update model parameters in Flux.jl?

○ `model.update()`

√ `Flux.update!(opt, params(model), gs)`

○ `optimize(model, opt)`

○ `model.step()`