LEIC-T 2024/2025
Aprendizagem - Machine Learning
Homework III

Group 36
Francisco Uva – 106340
Pedro Pais - 107482

I) Polynomial Regression

a)

$$\Phi = \begin{pmatrix} 1. & -1 & 1 & -1 \\ 1. & 1 & 1 & 1 \\ 1. & -1.2 & 1.44 & -1.728 \\ 1. & 1.4 & 1.96 & 2.744 \\ 1. & 1.9 & 3.61 & 6.859 \end{pmatrix}$$

b)

$$\Phi^T \Phi = \begin{pmatrix} 5 & 2.1 & 9.01 & 7.875 \\ 2.1 & 9.01 & 7.875 & 20.9473 \\ 9.01 & 7.875 & 20.9473 & 27.65091 \\ 7.875 & 20.9473 & 27.65091 & 59.561401 \end{pmatrix}$$

$w = (\Phi^T \Phi)^{-1} \Phi^T T = \{10.03942366, -1.93661531, -7.49664634, 2.59232475\}^T$

$y = 10.03942366 + -1.93661531*x + -7.49664634*x^2 + 2.59232475*x^3$

```python
import numpy as np

if __name__ == "__main__":
    A = [[1, -1, 1, -1],
         [1, 1, 1, 1],
         [1, -1.2, 1.44, -1.728],
         [1, 1.4, 1.96, 2.744],
         [1, 1.9, 3.61, 6.859]]

    A = np.array(A)
    AT = A.T

    result = np.dot(AT, A)

    print("A^T * A is:")
    print(result)
    print('\n')

    try:
        result_inv = np.linalg.inv(result)
        print("The inverse of A^T * A is:")
        print(result_inv)
    except np.linalg.LinAlgError:
        print("The matrix A^T * A is not invertible.")
    print('\n')

    result_inv_AT = np.dot(result_inv, AT)
    print("The inverse of A^T * A multiplied by A^T is:")
    print(result_inv_AT)
    print('\n')

    T = [[2, 3, -3, 0, -3],]
    T = np.array(T)

    result_T = np.dot(result_inv_AT, T.T)
    print("The result of the multiplication of the inverse of A^T * A with A^T and T is:")
    print(result_T)
```

Group 36
Francisco Uva – 106340
Pedro Pais - 107482

c)

A closed-form solution exists for Bayesian regression with L_2 regularization (Ridge regression) because the loss function, which combines the sum of squared errors and the smooth L_2 penalty, is quadratic and differentiable, allowing for direct computation using linear algebra. In contrast, LASSO regression, which employs L_1 regularization, does not have a closed-form solution due to the non-differentiable nature of the L_1 norm at zero, introducing sharp corners in the objective function that complicate optimization. As a result, LASSO requires specialized iterative algorithms, such as coordinate descent or proximal gradient descent, to approximate the solution, rather than being solvable through straightforward analytical methods.

II) Neural Network NN

$$z^{[1]} = W^{[1]} \cdot x + b^{[1]}$$

$$z^{[1]} = \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.5 \\ -5 \\ 5 \end{pmatrix}$$

Applying the ReLU function f(x) = max(0,x):

$$a^{[1]} = \begin{pmatrix} \max(0, 0.5) \\ \max(0, -5) \\ \max(0, 5) \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0 \\ 5 \end{pmatrix}$$

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

$$z^{[2]} = \begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0.5 \\ 0 \\ 5 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Applying SoftMax to obtain the probabilities:

LEIC-T 2024/2025
Aprendizagem - Machine Learning
Homework III

Group 36
Francisco Uva – 106340
Pedro Pais - 107482

SoftMax(z) = $\frac{e^z}{\sum e^z}$

SoftMax($z^{[2]}$) = $\frac{\binom{e^0}{e^1}}{e^0+e^1}$ = $\frac{\binom{1}{e}}{1+e} \approx \binom{0.2689}{0.7311}$

Calculation of $\delta^{[2]}$:

$$\delta^{[2]} = \frac{\partial Loss}{\partial z^{[2]}} = y - t = \binom{0.2689}{0.7311} - \binom{1}{0} = \binom{-0.7311}{0.7311}$$

$$\frac{\partial Loss}{\partial W^{[2]}} = \delta^{[2]} \cdot \left(a^{[1]}\right)^T = \binom{-0.7311}{0.7311} \cdot \begin{pmatrix} 0.5 & 0 & 5 \end{pmatrix} = \begin{pmatrix} -0.36555 & 0 & -3.6555 \\ 0.36555 & 0 & 3.6555 \end{pmatrix}$$

$$\frac{\partial Loss}{\partial b^{[2]}} = \delta^{[2]} = \binom{-0.7311}{0.7311}$$

$$\delta^{[1]} = \left(W^{[2]}\right)^T \cdot \delta^{[2]} \circ sgn_0\left(z^{[1]}\right) = \left(W^{[2]}\right)^T \cdot \delta^{[2]} \circ 1 = \begin{pmatrix} 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \binom{-0.7311}{0.7311} \circ \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

$$\delta^{[1]} = \begin{pmatrix} 1.4622 \\ 0 \\ 0 \end{pmatrix}$$

We need to update $W^{[1]}$, $b^{[1]}$, $W^{[2]}$, and $b^{[2]}$ because each of these parameters contributes to the overall loss during the forward pass. Gradients calculated via backpropagation show how changes in these weights and biases affect the loss. Since the gradients for all these parameters are non-zero, updating them helps the model learn and minimize the error for future predictions.

Weights and Bias Update:

Group 36
Francisco Uva – 106340
Pedro Pais - 107482

$\eta = 0.1$

$$new\ W^{[1]} = W^{[1]} - \eta \cdot \frac{\partial Loss}{\partial W^{[1]}} =$$

$$= \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} - 0.1 \begin{pmatrix} 1.4622 & 1.4622 & 1.4622 & 1.4622 & 1.4622 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} -0.04622 & -0.04622 & -0.04622 & -0.04622 & -0.04622 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$new\ b^{[1]} = b^{[1]} - \eta \cdot \delta^{[1]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} - 0.1 \begin{pmatrix} 1.4622 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.14622 \\ 0 \\ 0 \end{pmatrix}$$

$$new\ W^{[2]} = W^{[2]} - \eta \cdot \frac{\partial Loss}{\partial W^{[2]}} = \begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix} - 0.1 \begin{pmatrix} -0.36555 & 0 & -3.6555 \\ 0.36555 & 0 & 3.6555 \end{pmatrix}$$

$$new\ W^{[2]} = \begin{pmatrix} 0.0366 & 0 & 0.3656 \\ 1.9634 & 0 & -0.3656 \end{pmatrix}$$

$$new\ b^{[2]} = b^{[2]} - \eta \cdot \frac{\partial Loss}{\partial b^{[2]}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 0.1 \begin{pmatrix} -0.7311 \\ 0.7311 \end{pmatrix}$$

$$new\ b^{[2]} = \begin{pmatrix} 0.0731 \\ -0.0731 \end{pmatrix}$$

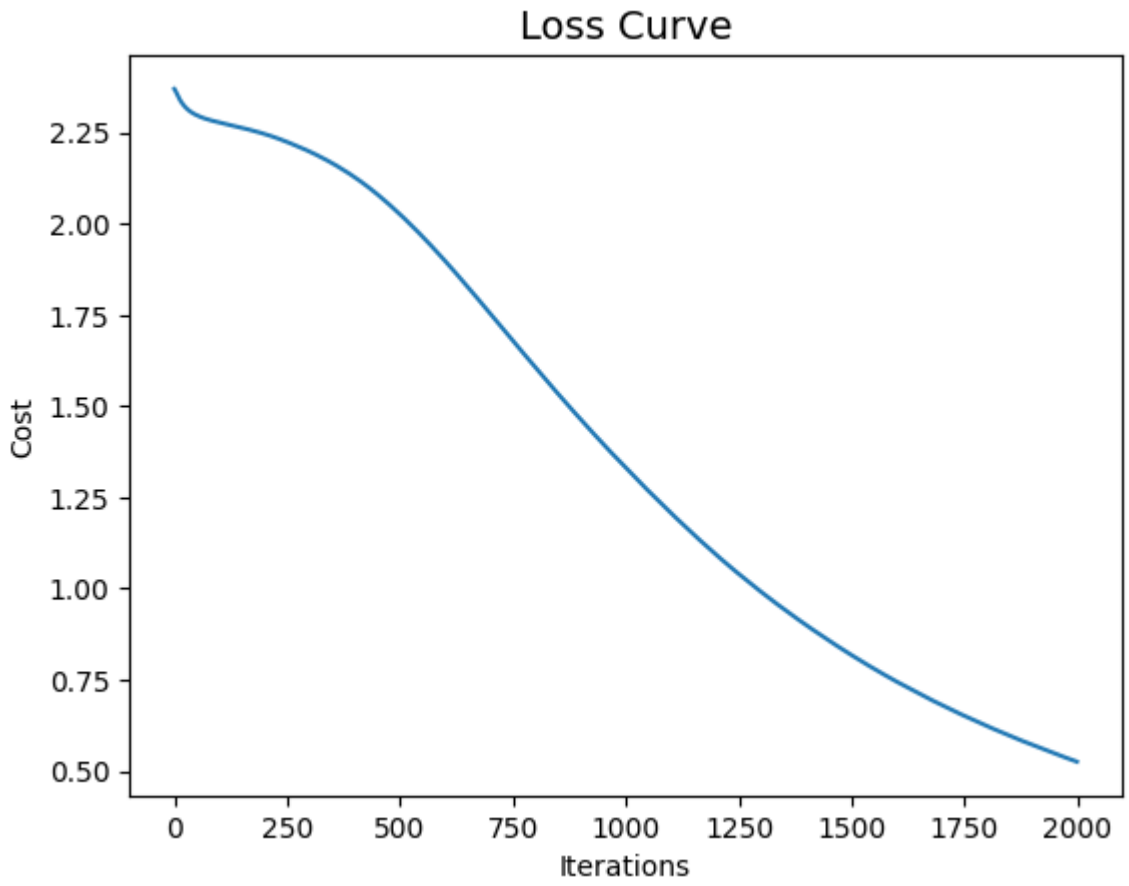
III) Software Experiments

a)

Using the optimized hidden layer size (20, 10) improved the test accuracy, indicating that a more complex network better captured the data patterns.

LEIC-T 2024/2025
Aprendizagem - Machine Learning
Homework III

Group 36
Francisco Uva – 106340
Pedro Pais - 107482



b)

No, the result is not better when using the 'identity' activation function because it does not introduce non-linearity, making the network behave like a linear model, which limits its ability to learn complex patterns in the data.