Aprendizagem 2024
Homework III – Group 44
(ist1106827, ist1107245)

Part I: Pen and paper

For questions in this group, show your numerical results with 5 decimals or scientific notation. Hint: we highly recommend the use of numpy (e.g., linalg.pinv for inverse) or other programmatic facilities to support the calculus involved in both questions (1) and (2). Below is a training dataset D composed by two input variables and two output variables, one of which is numerical (ynum) and the other categorical (yclass). Consider a polynomial basis function $\phi(y_1, y_2) = y_1 \times y_2$ that transforms the original space into a new one-dimensional space.

D	y_1	y_2	y_{num}	y_{class}
x_1	1	1	1.25	B
x_2	1	3	7.0	A
x_3	3	2	2.7	C
x_4	3	3	3.2	A
x_5	2	4	5.5	B

1. Learn a regression model on the transformed feature space using the OLS closed form solution to predict the continuous output ynum.

Regression model

$\omega = (X^T X)^{-1} X^T z$

$\phi(y_1, y_2) = y_1 \times y_2$

$x_1 = (1,1) \rightarrow \phi(x_1) = 1 \times 1 = 1$
$x_2 = (1,3) \rightarrow \phi(x_2) = 1 \times 3 = 3$
$x_3 = (3,2) \rightarrow \phi(x_3) = 3 \times 2 = 6$
$x_4 = (3,3) \rightarrow \phi(x_4) = 3 \times 3 = 9$
$x_5 = (2,4) \rightarrow \phi(x_5) = 2 \times 4 = 8$

$$ynum = \begin{bmatrix} 1,25 \\ 7,0 \\ 2,7 \\ 3,2 \\ 5,5 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 9 \\ 1 & 8 \end{bmatrix}$$

$$X^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix} \qquad X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 9 \\ 1 & 8 \end{bmatrix} = \begin{bmatrix} 5 & 27 \\ 27 & 191 \end{bmatrix}$$

$$(X^T X)^{-1} = \begin{bmatrix} \frac{191}{226} & -\frac{27}{226} \\ -\frac{27}{226} & \frac{5}{226} \end{bmatrix} \qquad (X^T X^{-1}) X^T = \begin{bmatrix} \frac{80}{113} & \frac{55}{113} & \frac{29}{226} & -\frac{26}{113} & -\frac{25}{226} \\ -\frac{11}{113} & -\frac{6}{113} & \frac{3}{226} & \frac{9}{113} & \frac{13}{226} \end{bmatrix}$$

$$\omega = (X^T X)^{-1} \cdot X^T z = (X^T X)^{-1} \cdot X^T \cdot \begin{bmatrix} 1,25 \\ 7,0 \\ 2,7 \\ 3,2 \\ 5,5 \end{bmatrix} = \begin{bmatrix} 3,3159 \\ 0,1137 \end{bmatrix}$$

(output)

$\hat{z} = 3,3159 + 0,1137y$

2. Repeat the previous exercise, but this time learn a Ridge regression with penalty factor $\lambda = 1$. Compare the learnt coefficients with the ones from the previous exercise and discuss how regularization affects them.

Ridge Regression

$$\omega = (X^T \cdot X + \lambda \cdot I)^{-1} \cdot X^T \cdot z$$

(I: matriz identidade; λ: fator de regularização / penalidade, $\lambda = 1$)

$$X^T X = \begin{bmatrix} 5 & 27 \\ 27 & 191 \end{bmatrix} \qquad X^T z = \begin{bmatrix} 19,65 \\ 111,25 \end{bmatrix}$$

$$X^T X + \lambda I = \begin{bmatrix} 5 & 27 \\ 27 & 191 \end{bmatrix} + 1 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 27 \\ 27 & 192 \end{bmatrix}$$

$$(X^T X + \lambda I)^{-1} = \begin{bmatrix} 6 & 27 \\ 27 & 192 \end{bmatrix}^{-1} = \begin{bmatrix} 64/41 & -3/47 \\ -3/47 & 2/41 \end{bmatrix}$$

$$\omega = \begin{bmatrix} 64/41 & -3/47 \\ -3/47 & 2/41 \end{bmatrix} \cdot \begin{bmatrix} 19,65 \\ 111,27 \end{bmatrix} = \begin{bmatrix} 1,8168 \\ 0,3240 \end{bmatrix}$$

$$\hat{z} = 1,8168 + 0,3240\, y$$

comparar:

- $\hat{z} = 3,3159 + 0,1137\, y$ (regression model) ↳ without regularization
- $\hat{z} = 1,8168 + 0,3240\, y$ (Ridge regression) ↳ with regularization

ω_0 (bias): $\dfrac{\omega_0 \text{ Ridge Regression}}{\omega_0 \text{ regression model}} = \dfrac{1,8168}{3,3159} = 0,5479$

ω_1 (peso): $\dfrac{\omega_1 \text{ Ridge Regression}}{\omega_1 \text{ regression model}} = \dfrac{0,3240}{0,1137} = 2,8496$

In Ridge Regression, the bias term significantly decreased from 2.3159 in the standard regression model to 1.8168. This reduction stems from regularization, which shrinks coefficients to mitigate overfitting and reduces reliance on the intercept, resulting in a more stable and generalizable model.

Conversely, the weight for the feature y increased substantially from 0.1137 to 0.3240. This change underscores how regularization redistributes feature importance —diminishing the bias's influence while enhancing the feature's contribution.

As a result, Ridge Regression achieves improved generalization by more effectively balancing the relationships between features and the target variable.

3. Given three new test observations and their corresponding *ynum* output x6 = (2, 2, 0. 7), x7 = (1, 2, 1. 1) and x8 = (5, 1, 2. 2), compare the train and test RMSE of the two models obtained in 1) and 2). Explain if the results go according to what is expected.

$$RMSE(\hat{z}, z) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_i - \hat{z}_i)^2}$$

medida de erro — ROOT MEAN SQUARE ERROR; z_i real, $\hat{z}_i$ previsto

$x_6 = (2, 2, 0.7)$, $x_7 = (1, 2, 1.1)$, $x_8 = (5, 1, 2.2)$

$x_6 = (2, 2) \rightarrow \phi(x_6) = 2 \times 2 = 4$
$x_7 = (1, 2) \rightarrow \phi(x_7) = 1 \times 2 = 2$
$x_8 = (5, 1) \rightarrow \phi(x_8) = 5 \times 1 = 5$

1) Regression Model: $\hat{z} = 3,3159 + 0,1137\,y$

train:

$\hat{z}_1 = w_0 + w_1 \times \phi(1) = 3,3159 + 0,1137 \times 1 = 3,4296$
$\hat{z}_2 = w_0 + w_1 \times \phi(2) = 3,3159 + 0,1137 \times 3 = 3,6570$
$\hat{z}_3 = w_0 + w_1 \times \phi(3) = 3,3159 + 0,1137 \times 6 = 3,9981$
$\hat{z}_4 = w_0 + w_1 \times \phi(4) = 3,3159 + 0,1137 \times 9 = 4,3392$
$\hat{z}_5 = w_0 + w_1 \times \phi(5) = 3,3159 + 0,1137 \times 8 = 4,2255$
$\hat{z}_6 = w_0 + w_1 \times \phi(6) = 3,3159 + 0,1137 \times 4 = 3,7707$
$\hat{z}_7 = w_0 + w_1 \times \phi(7) = 3,3159 + 0,1137 \times 2 = 3,5433$
$\hat{z}_8 = w_0 + w_1 \times \phi(8) = 3,3159 + 0,1137 \times 5 = 3,8844$

$$RMSE = \sqrt{\frac{(1,25 - 3,4296)^2 + (7,0 - 3,6570)^2 + (2,7 - 3,9981)^2 + (3,2 - 4,3392)^2 + (5,5 - 4,2255)^2}{5}} =$$

$= 2,0265$ train

$$RMSE = \sqrt{\frac{(0,7 - 3,7707)^2 + (1,1 - 3,5433)^2 + (2,2 - 3,8844)^2}{3}} = 2,4655$$ test

2) Ridge Regression: $\hat{z} = 1,8168 + 0,3240\,y$

train:

$\hat{z}_1 = w_0 + w_1 \times \phi(1) = 1,8168 + 0,3240 \times 1 = 2,1408$

$\hat{z}_2 = w_0 + w_1 \times \phi(2) = 1,8168 + 0,3240 \times 3 = 2,7888$

$\hat{z}_3 = w_0 + w_1 \times \phi(3) = 1,8168 + 0,3240 \times 6 = 3,7608$

$\hat{z}_4 = w_0 + w_1 \times \phi(4) = 1,8168 + 0,3240 \times 9 = 4,7328$

$\hat{z}_5 = w_0 + w_1 \times \phi(5) = 1,8168 + 0,3240 \times 8 = 4,4088$

$\hat{z}_6 = w_0 + w_1 \times \phi(6) = 1,8168 + 0,3240 \times 4 = 3,1128$

$\hat{z}_7 = w_0 + w_1 \times \phi(7) = 1,8168 + 0,3240 \times 2 = 2,4648$

$\hat{z}_8 = w_0 + w_1 \times \phi(8) = 1,8168 + 0,3240 \times 5 = 3,4368$

$$RMSE = \sqrt{\frac{(1,25 - 2,1408)^2 + (7,0 - 2,7888)^2 + (2,7 - 3,7608)^2 + (3,2 - 4,7328)^2 + (5,5 - 4,4088)^2}{5}} =$$

$$= 2,1536 \quad \text{train}$$

$$RMSE = \sqrt{\frac{(0,7 - 3,1128)^2 + (1,1 - 2,4648)^2 + (2,2 - 3,4368)^2}{3}} = \text{test } 1,7525$$

Conclusão:

1) train: RMSE = 2,0265 ; test: RMSE = 2,4655

2) train: RMSE = 2,1536 ; test: RMSE = 1,7525

Em comparação:

$\text{train}: \frac{\text{train 2)}}{\text{train 1)}} = \frac{2,1536}{2,0265} = 1,0627$

$\text{test}: \frac{\text{test 2)}}{\text{test 1)}} = \frac{1,7525}{2,4655} = 0,7108$

When comparing the train and test RMSE (Root Mean Squared Error) of the two models, we observe a slightly higher value in the training set for the second model (Ridge Regression). However, in the test set, the difference is more noticeable, with the first model (Linear Regression) having a higher RMSE. This indicates that the Ridge model performed better at predicting the *ynum* values for the test observations.

These results are consistent with the expected behavior of the models, where regularization in Ridge improves generalization and prevents overfitting.

4. Consider an MLP to predict the output y_{class}

$$W^{[1]} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.2 \\ 0.2 & 0.1 \end{bmatrix} \quad b^{[1]} = \begin{bmatrix} 0.1 \\ 0 \\ 0.1 \end{bmatrix} \quad W^{[2]} = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad b^{[2]} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

characterized by the weights the output activation function $softmax(Z_c^{[out]}) = \frac{e^{Z_c^{[out]}}}{\sum_{l=1}^{|C|} e^{Z_l^{[out]}}}$, no activations on the hidden layer(s) and the cross-entropy loss: $-\sum_{i=1}^{N}\sum_{l=1}^{|C|} t_l^{(i)} log(Z_l^{[out](i)})$. Consider that the output layer of the MLP gives the predictions for the classes A, B and C in this order. Perform one stochastic gradient descent update to all the weights and biases with learning rate $\eta = 0.1$ using the training observation x_1.

MLP (rede neuronal multicamada)

$\text{softmax}(z_c^{[out]}) = \frac{e^{z_c^{[out]}}}{\sum_{l=1}^{C} e^{z_l^{[out]}}}$

CROSS-entropy loss: $-\sum_{i=1}^{N}\sum_{l=1}^{[C]} t_l^{(i)} \log(X_l^{[out](i)})$

$\eta = 0,1$ usar x_1

ID	y_1	y_2	y_{num}	y_{out}
x_1	1	1	1,25	B

camada 1 (entrada → oculta)
camada 2 (oculta → output)

A rede tem: camada oculta (sem funções de ativação)
camada de saída com função softmax

Foward Prop

$z^{[1]} = w^{[1]} \cdot X^{[0]} + b^{[1]} = \begin{bmatrix} 0,1 & 0,1 \\ 0,1 & 0,2 \\ 0,2 & 0,1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0,1 \\ 0 \\ 0,1 \end{bmatrix} = \begin{bmatrix} 0,3 \\ 0,3 \\ 0,4 \end{bmatrix}$ (3×2, 2×1, 3×1)

$X^{[1]} = z^{[1]}$

$z^{[2]} = w^{[2]} \cdot X^{[1]} + b^{[2]} = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0,3 \\ 0,3 \\ 0,4 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2,7 \\ 2,3 \\ 2 \end{bmatrix}$ (3×3, 3×1, 3×1)

$X^{[2]} = \text{softmax}(z^{[2]}) = \begin{bmatrix} 0,462 \\ 0,309 \\ 0,229 \end{bmatrix}$

CA

$e^{2,7} = 14,879$
$e^{2,3} = 9,974$
$e^{2} = 7,389$
$\sum_{i=1}^{C} e^{z_i^{[out]}} = 32,233$

Back Prop $w^{[2]} = w^{[2]} - \eta \cdot \frac{\partial E}{\partial w^{[2]}}$

$\frac{\partial E}{\partial w^{[2]}} = \underbrace{\frac{\partial E}{\partial X^{[2]}} \circ \frac{\partial X^{[2]}}{\partial z^{[2]}}}_{\delta^{[2]}} \cdot \left(\frac{\partial z^{[2]}}{\partial w^{[2]}}\right)^T$ ($\frac{\partial z^{[2]}}{\partial w^{[2]}} = (X^{[1]})^T$)

$\frac{\partial E}{\partial z^{[2]}} = X^{[2]} - t = \begin{bmatrix} 0,462 \\ 0,309 \\ 0,229 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0,462 \\ -0,691 \\ 0,229 \end{bmatrix}$

($X^{[2]}$: vetor dos outputs reais; t: output previsto)

quando erro = CROSS-entropy e ativação na última layer é softmax $\delta^{[out]} = X^{[out]} - t$

$\delta^{[2]} = (X^{[2]} - t) = \begin{bmatrix} 0,462 \\ -0,691 \\ 0,229 \end{bmatrix}$

$$W^{[2]} = W^{[2]} - 0{,}1 \cdot (\delta^{[2]} \cdot (X^{[2]})^T) = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} - 0{,}1 \cdot \left(\begin{bmatrix} 0{,}462 \\ -0{,}691 \\ 0{,}229 \end{bmatrix} \cdot \begin{bmatrix} 0{,}3 \\ 0{,}3 \\ 0{,}4 \end{bmatrix}^T \right) =$$

$$= \begin{bmatrix} 0{,}98614 & 1{,}98614 & 1{,}98152 \\ 1{,}02073 & 2{,}02073 & 1{,}02764 \\ 0{,}99313 & 0{,}99313 & 0{,}99084 \end{bmatrix}$$

$$b^{[2]} = b^{[2]} - \eta \cdot \frac{\partial E}{\partial b^{[2]}} = b^{[2]} - 0{,}1 \cdot \delta^{[2]} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - 0{,}1 \cdot \begin{bmatrix} 0{,}462 \\ -0{,}691 \\ 0{,}229 \end{bmatrix} = \begin{bmatrix} 0{,}9538 \\ 1{,}0691 \\ 0{,}9771 \end{bmatrix}$$

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

$$\frac{\partial E}{\partial W^{[1]}} = \underbrace{\frac{\partial E}{\partial X^{[1]}} \circ \frac{\partial X^{[1]}}{\partial Z^{[1]}}}_{\delta^{[1]}} \cdot \underbrace{\left(\frac{\partial Z^{[1]}}{\partial W^{[1]}} \right)^T}_{(X^{[0]})^T} \qquad \frac{\partial X^{[1]}}{\partial Z^{[1]}} = \frac{\partial Z^{[1]}}{\partial Z^{[1]}} = 1$$

$$\delta^{[1]} = \frac{\partial E}{\partial Z^{[1]}} \circ \frac{\partial X^{[1]}}{\partial Z^{[1]}} = \left(\left(\underbrace{\frac{\partial Z^{[2]}}{\partial X^{[1]}}}_{W^{[2]}} \right)^T \cdot \delta^{[2]} \right) \circ 1 =$$

(efeito de $X^{[1]}$ em $Z^{[2]}$; regra da cadeia)

$$= \begin{bmatrix} 0{,}98614 & 1{,}98614 & 1{,}98152 \\ 1{,}02073 & 2{,}02073 & 1{,}02764 \\ 0{,}99313 & 0{,}99313 & 0{,}99084 \end{bmatrix}^T \begin{bmatrix} 0{,}462 \\ -0{,}691 \\ 0{,}229 \end{bmatrix} = \begin{bmatrix} -0{,}0230 \\ -0{,}2510 \\ 0{,}4326 \end{bmatrix}$$

$$W^{[1]} = W^{[1]} - 0{,}1 \cdot (\delta^{[1]} \cdot (X^{[1]})^T) = \begin{bmatrix} 0{,}1 & 0{,}1 \\ 0{,}1 & 0{,}2 \\ 0{,}2 & 0{,}1 \end{bmatrix} - 0{,}1 \cdot \left(\begin{bmatrix} -0{,}0230 \\ -0{,}2510 \\ 0{,}4326 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T \right) =$$

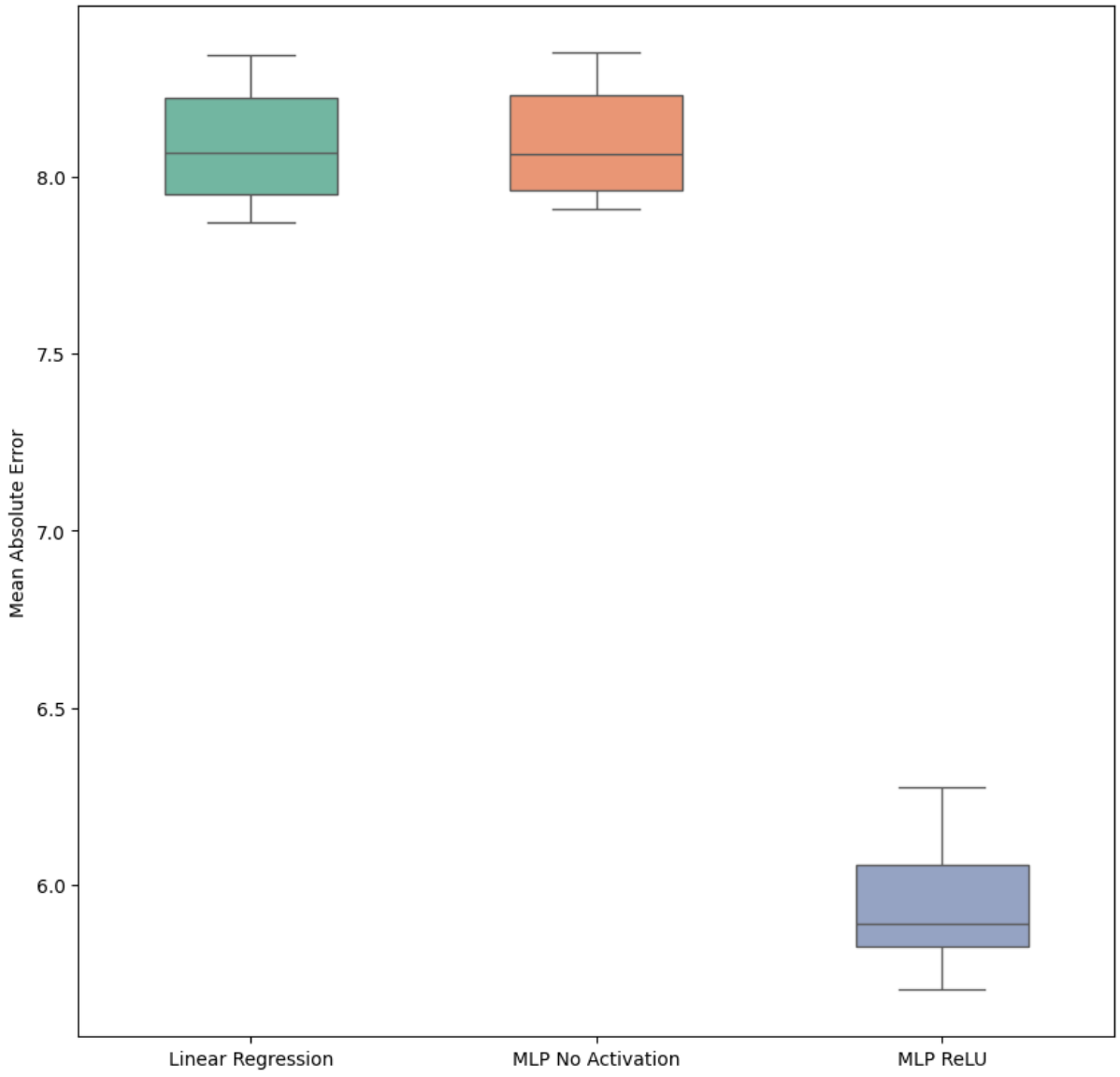
$$= \begin{bmatrix} 0{,}1023 & 0{,}1023 \\ 0{,}1251 & 0{,}2251 \\ 0{,}1567 & 0{,}0567 \end{bmatrix}$$

$$b^{[1]} = b^{[1]} - \eta \cdot \frac{\partial E}{\partial b^{[1]}} = b^{[1]} - 0{,}1 \cdot \delta^{[1]} = \begin{bmatrix} 0{,}1 \\ 0 \\ 0{,}1 \end{bmatrix} - 0{,}1 \cdot \begin{bmatrix} -0{,}0230 \\ -0{,}2510 \\ 0{,}4326 \end{bmatrix} = \begin{bmatrix} 0{,}1023 \\ 0{,}0251 \\ 0{,}0567 \end{bmatrix}$$

Part II: Programming

Consider the parkinsons.csv dataset (available at the course's webpage), where the goal is to predict a patient's score on the Unified Parkinson's Disease Rating Scale based on various biomedical measurements. To answer question 5), average the performance of the models over 10 separate runs. In each run, use a different 80-20 train-test split by setting a random_state = i, with i=1..10.

(a) Train a Linear Regression model, an MLP Regressor with 2 hidden layers of 10 neurons each and no activation functions, and another MLP Regressor with 2 hidden layers of 10 neurons each using ReLU activation functions. (Use random_state = 0 on the MLPs, regardless of the run). Plot a boxplot of the test MAE of each model.



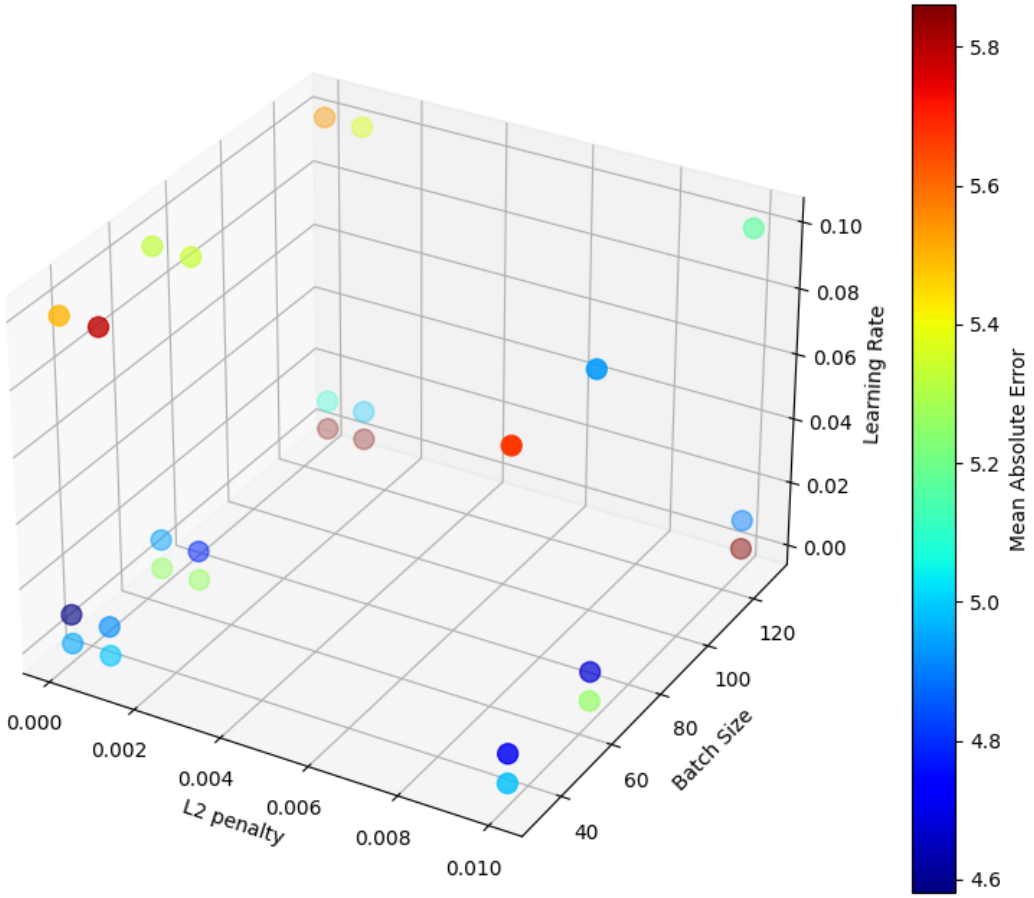
(b) Compare a Linear Regression with a MLP with no activations, and explain the impact and the importance of using activation functions in a MLP . Support your reasoning with the results from the boxplots.

Linear Regression and the MLP without activation functions show comparable Mean Absolute Error values around 8.0. This is because an MLP without activation functions behaves like a linear model. It lacks the ability to model non-linear relationships in the data, making it functionally similar to Linear Regression, where the output is simply a linear combination of the inputs.

The MLP with ReLU activation, on the other hand, has a substantially lower Mean Absolute Erro around 6.0, indicating a significant improvement in performance. The ReLU activation function introduces non-linearity to the model, enabling it to capture complex patterns in the data. Without non-linearity, the model is limited in its expressiveness and can't effectively handle real-world data that often contains non-linear relationships.

The results emphasize that activation functions are essential in neural networks because they allow the network to learn and model complex, non-linear patterns that a purely linear model like Linear Regression or an MLP without activations cannot capture.

(c) Using a 80-20 train-test split with random_state = 0, use a Grid Search to tune the hyperparameters of an MLP regressor with two hidden layers (size 10 each). The parameters to search over are: (i) L2 penalty, with the values {0.0001, 0.001, 0.01}; (ii) learning rate, with the values {0.001, 0.01, 0.1}; and (iii) batch size, with the values {32, 64, 128}. Plot the test MAE for each combination of hyperparameters, report the best combination, and discuss the trade-offs between the combinations.



The best combination hyperparameters found by the Grid Search are:

- L2 penalty: 0.0001
- Learning rate: 0.01
- Batch size: 32

The plot shows that smaller learning rates, like 0.01, generally produce lower errors. This is because a smaller learning rate allows the model to make finer, more controlled updates to the weights, helping it avoid local minimums. Larger learning rates, on the other hand, lead to larger jumps, which often result in suboptimal solutions, as reflected in higher error values.

Similarly, smaller batch sizes tend to perform better. This is consistent with the fact that smaller batches allow for more frequent updates, aiding the model in fine-tuning its understanding. However, small batch sizes demand more computational resources, so there's a trade-off between performance and efficiency.

Interestingly, the L2 penalty doesn't seem to significantly influence the model's performance. Across different L2 values, the error remains relatively constant. This could be due to the regularization having a minimal effect on overfitting in this context, especially since the training and testing errors are quite similar.

In conclusion, the most effective combinations involve small learning rates and batch sizes. L2 penalties don't seem to have a very expressive impact on the MEA, rendering its fine tuning relatively unnecessary. These combinations offer the right balance by allowing the model to learn effectively without overfitting or making drastic updates. High learning rates paired with smaller batch sizes yield worse results than when combined with larger batches. This result is rather surprising and suggests that balancing these parameters is key to minimizing error and improving prediction accuracy.

Appendix

```python
from sklearn.exceptions import ConvergenceWarning
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import StratifiedKFold
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import warnings

df = pd.read_csv("./data/parkinsons.csv")

X = df.drop("target", axis = 1)
y = df["target"]

warnings.filterwarnings(action = "ignore", category = ConvergenceWarning)

random_seeds = range(1, 11)

results = {"Linear Regression": [], "MLP No Activation": [], "MLP ReLU": []}

for i in random_seeds:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = i)

    lr = LinearRegression()
    lr.fit(X_train, y_train)
    y_pred_lr = lr.predict(X_test)
    results["Linear Regression"].append(mean_absolute_error(y_test, y_pred_lr))

    mlp_no_act = MLPRegressor(hidden_layer_sizes = (10, 10), activation ="identity", random_state = 0)
    mlp_no_act.fit(X_train, y_train)
    y_pred_mlp_no_act = mlp_no_act.predict(X_test)
    results["MLP No Activation"].append(mean_absolute_error(y_test, y_pred_mlp_no_act))

    mlp_relu = MLPRegressor(hidden_layer_sizes = (10, 10), activation = "relu", random_state = 0)
    mlp_relu.fit(X_train, y_train)
    y_pred_mlp_relu = mlp_relu.predict(X_test)
    results["MLP ReLU"].append(mean_absolute_error(y_test, y_pred_mlp_relu))

results_df = pd.DataFrame(results)

plt.figure(figsize = (10, 10))

sns.boxplot(data = results_df, palette = "Set2", width = 0.5)
plt.ylabel('Mean Absolute Error')

plt.show()
```

Listing 1: Boxplots for Linear Regression model, MLP with No Activation, and MLP with ReLU

Activation

```
import matplotlib.cm as cm
import matplotlib.colors as mcolors

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
    random_state = 0)

mlp = MLPRegressor(hidden_layer_sizes = (10, 10), random_state = 0)

param_grid = {
    "alpha": [0.0001, 0.001, 0.01],
    "learning_rate_init": [0.001, 0.01, 0.1],
    "batch_size": [32, 64, 128]
}

grid_search = GridSearchCV(mlp, param_grid, scoring = "neg_mean_absolute_error")
grid_search.fit(X_train, y_train)

results = grid_search.cv_results_

alphas = results["param_alpha"].data
learning_rates = results["param_learning_rate_init"].data
batch_sizes = results["param_batch_size"].data
mean_absolute_errors = -results["mean_test_score"]

print("Best parameters found: ", grid_search.best_params_)

fig = plt.figure(figsize = (12, 8))
ax = fig.add_subplot(111, projection = "3d")

sc = ax.scatter(alphas, batch_sizes, learning_rates, c = mean_absolute_errors,
    cmap = cm.jet, s = 100)

ax.set_xlabel("L2 penalty")
ax.set_ylabel("Batch Size")
ax.set_zlabel("Learning Rate")

cbar = plt.colorbar(sc, ax=ax, label = "Mean Absolute Error")

plt.show()
```

Listing 2: Test MAE for hyperparameter combinations of MLP Regressor with two hidden layers