TÉCNICO LISBOA

Aprendizagem 2023/24

Homework I – Group XXX

(ist105940, ist, ist1106221)

I. Pen-and-paper

1

$y_{num} = w_0 + w_1 \phi(y_1, y_2)$ fazemos o $\phi(y_1, y_2)$ para cada ponto

$W = (X^T X)^{-1} X^T Z$

$\phi(x_1) = 1 \times 1 = 1$

$\phi(x_2) = 3$

$\phi(x_3) = 6$

$\phi(x_4) = 9$

$\phi(x_5) = 8$

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 9 \\ 1 & 8 \end{bmatrix}$$

$$Z = \begin{bmatrix} 1.25 \\ 7 \\ 2.7 \\ 3.2 \\ 5.5 \end{bmatrix} \quad W = \left(\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{pmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 9 \\ 1 & 8 \end{bmatrix} \right)^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{pmatrix} \times \begin{bmatrix} 1.25 \\ 7 \\ 2.7 \\ 3.2 \\ 5.5 \end{bmatrix}$$

$$= \left\| \begin{pmatrix} 5 & 27 \\ 27 & 191 \end{pmatrix} \right\|^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{pmatrix} \times \begin{bmatrix} 1.25 \\ 7 \\ 2.7 \\ 3.2 \\ 5.5 \end{bmatrix} = \begin{bmatrix} \cancel{\frac{749.4}{226}} \\ \cancel{\frac{25.7}{226}} \end{bmatrix} = \cancel{\begin{bmatrix} 3.316 \\ 0.1137 \end{bmatrix}}$$

$y_{num} = \cancel{3.30447}\ 3.3159 + 0.11370 \times \phi(y_1, y_2)$

$\cancel{y_{num} = 3.370442 + 0.1170\, \phi(y_1, y_2)}$

$$= \cancel{\begin{bmatrix} 3.36942 \\ 0.1137 \end{bmatrix}}$$

$$= \begin{bmatrix} 3.31593 \\ 0.11372 \end{bmatrix} \quad y_{num} = 3.31593 + 0.11372\, \phi(y_1, y_2)$$

2.8

$$w_{rid} = (X^T X + \lambda I)^{-1} X^T y$$

$$= \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 9 \\ 1 & 8 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix} \times \begin{bmatrix} 1.25 \\ 7 \\ 2.7 \\ 3.2 \\ 5.5 \end{bmatrix}$$

$$= \cancel{\begin{bmatrix} 1.813 \\ 0.32695 \end{bmatrix}} = \cancel{\begin{bmatrix} 1.81106 \\ 0.32695 \end{bmatrix}} = \begin{bmatrix} 1.81809 \\ 0.32376 \end{bmatrix}$$

O Ridge model afetou ~~fo~~ bastante a ~~proporção~~ muito mais do peso de cada ~~uma~~ ~~todos~~ dados ~~...~~ $\phi(y_1, y_2)$, ~~dos~~ pesos ~~a~~ à ~~feaso~~ feature transformada em comparação com o modelo OLS ~~que o modelo OLS~~ assim suaviza o impacto de cada uma da ~~feta~~ features no modelo.

$$y_{num} = \cancel{1.81106 + 0.32695\,\phi(y_1, y_2)}$$

$$1.81809 + 0.32376\,\phi(y_1, y_2)$$

3. (teste)

para OLS

$y_{num}(x_6) = 3.31593 + 4 \times 0.11372 = \cancel{3.77077}$ 3.77081

$y_{num}(x_7) = 3.31593 + 2 \times 0.11372 = \cancel{3.54335}$ 3.54337

$y_{num}(x_8) = 3.31593 + 5 \times 0.11372 = \cancel{3.88448}$ 3.88453

para Ridge

$y_{num}(x_6) = 1.81809 + 4 \times 0.32376 = 3.11313$

$y_{num}(x_7) = 1.81809 + 2 \times 0.32376 = 2.46561$

$y_{num}(x_8) = 1.81809 + 5 \times 0.32376 = 3.43689$

Valores reais de $x_{6_{ynum}} = 0.7$ $x_{7_{ynum}} = 1.1$ $x_{8_{ynum}} = 2.2$

~~RMSE teste para OLS~~

RMSE teste para OLS

$$= \sqrt{\frac{(3.77081 - 0.7)^2 + (3.54337 - 1.1)^2 + (3.88453 - 2.2)^2}{3}}$$

$= \cancel{2.46954}$ 2.4656

RMSE teste para Ridge =

$$\sqrt{\frac{(3.11313 - 0.7)^2 + (2.46561 - 1.1)^2 + (3.43689 - 2.2)^2}{3}}$$

$= 1.7529$

para OLS treino

$y_{num}(x_1) = 3.42965$ $\quad y_{num}(x_2) = 3.65709$

$y_{num}(x_3) = 3.97825$ $\quad y_{num}(x_4) = 4.33941$

$y_{num}(x_5) = 4.22569$

para Ridge

$y_{num}(x_1) = 2.14185$ $\quad y_{num}(x_2) = 2.78937$

$y_{num}(x_3) = 3.76065$ $\quad y_{num}(x_4) = \cancel{4.43673}$ 4.73193

$y_{num}(x_5) = 4.40817$

RMSE treino OLS =

$$\sqrt{\frac{(3.42965-1.25)^2+(\cancel{2.23}\ 3.65709-7)^2+(3.97825-2.7)^2+(4.33941-3.2)^2+(4.22569-5.5)^2}{5}}$$

$= 2.02635$

RMSE treino Ridge =

$$\sqrt{\frac{(2.14185-1.25)^2+(2.78937-7)^2+(3.76065-2.7)^2+(\cancel{4.43673}\ 4.7193-3.2)^2+(4.40817-5.5)^2}{5}}$$

$= \cancel{2.1151}$ 2.15354

~~o OLS~~ Nos dados de treino o OLS tem um valor mais baixo (2.01635) comparado com o Ridge (2.15354) isto é esperado porque o OLS é optimizado para os dados de treino podendo levar a overfiting, enquanto o Ridge adiciona uma penalização fazendo o modelo não se adaptar tão bem aos dados de treino. Nos dados de teste o Ridge tem um desempenho muito melhor (1.7529) comparado com (1.46564) OLS, isto é esperado porque o Ridge evita o overfiting, assim sendo podemos dizer que a regularização ajudou o modelo a generalizar melhor os dados de teste.

4

forward Pass

$$z^{(1)} = W^{(1)} x_1 + b^{(1)} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.2 \\ 0.2 & 0.1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0 \\ 0.1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.2 \\ 0.3 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.4 \end{bmatrix}$$

$$z^{(2)} = W^{(2)} z^{(1)} + b^{(2)} = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.3 \\ 0.4 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2.7 \\ 2.3 \\ 2.0 \end{bmatrix}$$

softmax activation

$$\sigma(z^{(2)})_c = \frac{e^{z_c}}{\sum_{l=1}^{|c|} e^{z_l}} \qquad e^{z^{(2)}} = \begin{bmatrix} e^{2.7} \\ e^{2.3} \\ e^{2.0} \end{bmatrix} = \begin{bmatrix} 14.87973 \\ 9.97418 \\ 7.38906 \end{bmatrix}$$

$$\sum_{l=1}^{3} e^{z_l^{(2)}} = 14.87973 + 9.97418 + 7.3806$$
$$= 32.24297$$

$$\cancel{\sigma(z^{(2)})_c} \quad X^{(2)} = \sigma(z^{(2)}) = \begin{bmatrix} 0.46148 \\ 0.30934 \\ 0.22917 \end{bmatrix}$$

$$\frac{\partial E}{\partial W^{(l)}} = \frac{\partial E}{\partial z^{(l)}} \left(x^{(l-1)}\right)^T$$

$$\frac{\partial E}{\partial b^{(l)}} = \frac{\partial E}{\partial z^{(l)}}$$

$$\frac{\partial E}{\partial z^{(l)}} = \frac{\partial E}{\partial x^{(l)}} \circ \frac{\partial x^{l}}{\partial z^{l}} = \delta^{(l)}$$

$$\frac{\partial E}{\partial x_c^{(l)}} = \frac{\partial}{\partial x_c^{(l)}} -t_c^{(l)} \log\left(x_c^{(l)}\right) = \frac{-t_c}{\phi\left(z_c^{(l)}\right)}$$

$$\frac{\partial x_c^{(l)}}{\partial z_c^{(l)}} = \frac{\partial}{\partial z_c^{(l)}} \frac{e^{z_c^{(l)}}}{\sum_{k=1}^{|C|} e^{z_k^{(l)}}} = \phi\left(z_c^{(l)}\right)\left(1-\phi\left(z_c^{(l)}\right)\right)$$

$$\delta_c^{(l)} = \frac{-t_c}{\phi\left(z_c^{(l)}\right)} \times \phi\left(z_c^{l}\right)\left(1-\phi\left(z_c^{(l)}\right)\right) = \phi\left(z_c^{(l)}\right) - t_c$$

$$\frac{\partial E}{\partial x^{(1)}} = \sum_{l=1} t_l \; X_l^{(2)} \; \partial z_c^{(1)}$$

$$\delta^{(2)} = \begin{bmatrix} 0.46149-0 \\ 0.30934-1 \\ 0.22917-0 \end{bmatrix} = \begin{bmatrix} 0.46149 \\ -0.69066 \\ 0.22917 \end{bmatrix}$$

$$\frac{\partial E}{\partial W^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \cdot \delta^{(2)} \cdot \left(z^{(1)}\right)^T = \begin{bmatrix} \cancel{0.46149} & 0.46149 \\ \cancel{0.30934} & -0.69066 \\ & 0.22917 \end{bmatrix} \cdot \begin{bmatrix} 0.3 & 0.3 & 0.4 \end{bmatrix}$$

$$= \begin{bmatrix} 0.13845 & 0.13845 & 0.184596 \\ -0.10720 & -0.20720 & -0.27626 \\ 0.06875 & 0.06875 & 0.09167 \end{bmatrix}$$

$$\frac{\partial E}{\partial b^{(2)}} = \begin{pmatrix} 0.46149 \\ -0.69066 \\ 0.22917 \end{pmatrix} = \delta^{(2)}$$

atualização dos pesos

$$W_{new}^{(2)} = W_{old}^{(2)} - \eta \times \frac{\partial E}{\partial W^{(2)}} = \begin{pmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} - 0.1 \begin{bmatrix} 0.13845 & 0.13845 & 0.184596 \\ -0.10720 & -0.20720 & -0.27626 \\ 0.06875 & 0.06875 & 0.09167 \end{bmatrix}$$

$$= \begin{bmatrix} 0.98616 & 1.98616 & 1.98154 \\ 1.01072 & 2.02072 & 1.02763 \\ 0.99313 & 0.99313 & 0.99083 \end{bmatrix}$$

$$b^{(2)}_{new} = b^{(2)}_{old} - 0.1 \times \delta^{(2)} = \begin{bmatrix}1\\1\\1\end{bmatrix} - 0.1 \times \begin{bmatrix}0.46149\\-0.69066\\0.22917\end{bmatrix}$$

$$= \begin{bmatrix}0.95385\\1.06907\\0.97708\end{bmatrix}$$

$$\delta^{(1)} = \left(W^{(2)}_{old}\right)^T \cdot \delta^{(2)} \circ \cancel{f'(x^{(1)})} \left\{ \frac{\partial x^{(1)}}{\partial z^{(1)}} \right.$$

$$= \left(W^{(2)}_{old}\right)^T \cdot \delta^{(2)} \times 1 = \begin{bmatrix}1&1&1\\2&2&1\\2&1&1\end{bmatrix}\begin{bmatrix}0.46149\\-0.69066\\0.22917\end{bmatrix}$$

$$= \begin{bmatrix}0\\-0.22917\\0.46149\end{bmatrix}$$

$$\frac{\partial E}{\partial W^{(1)}} = \cancel{\frac{\partial E}{\partial z^{(1)}}}\ \delta^{(1)} \cdot (x_1)^T = \begin{bmatrix}0\\-0.22917\\0.46149\end{bmatrix}(1,1) =$$

$$= \begin{bmatrix}0&0\\-0.22917&-0.22917\\0.46149&0.46149\end{bmatrix}$$

$$\frac{\partial E}{\partial b^{(1)}} = \cancel{\partial E}\ \delta^{(1)} = \begin{bmatrix}0\\-0.22917\\0.46149\end{bmatrix}$$

atualizar peso de $W^{(1)}$

$$W^{(1)}_{new} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.2 \\ 0.2 & 0.1 \end{bmatrix} - 0.1 \times \begin{bmatrix} 0 & 0 \\ -0.22917 & -0.22917 \\ 0.46149 & 0.46149 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 & 0.1 \\ 0.122917 & 0.222917 \\ 0.153851 & 0.05385 \end{bmatrix}$$

$$b^{(1)}_{new} = b^{(1)}_{old} - 0.1 \times \delta^{(1)} = \begin{bmatrix} 0.1 \\ 0 \\ 0.1 \end{bmatrix} - 0.1 \times \begin{bmatrix} 0 \\ -0.22917 \\ 0.46149 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 \\ 0.022917 \\ 0.05381 \end{bmatrix}$$

Aprendizagem 2023/24

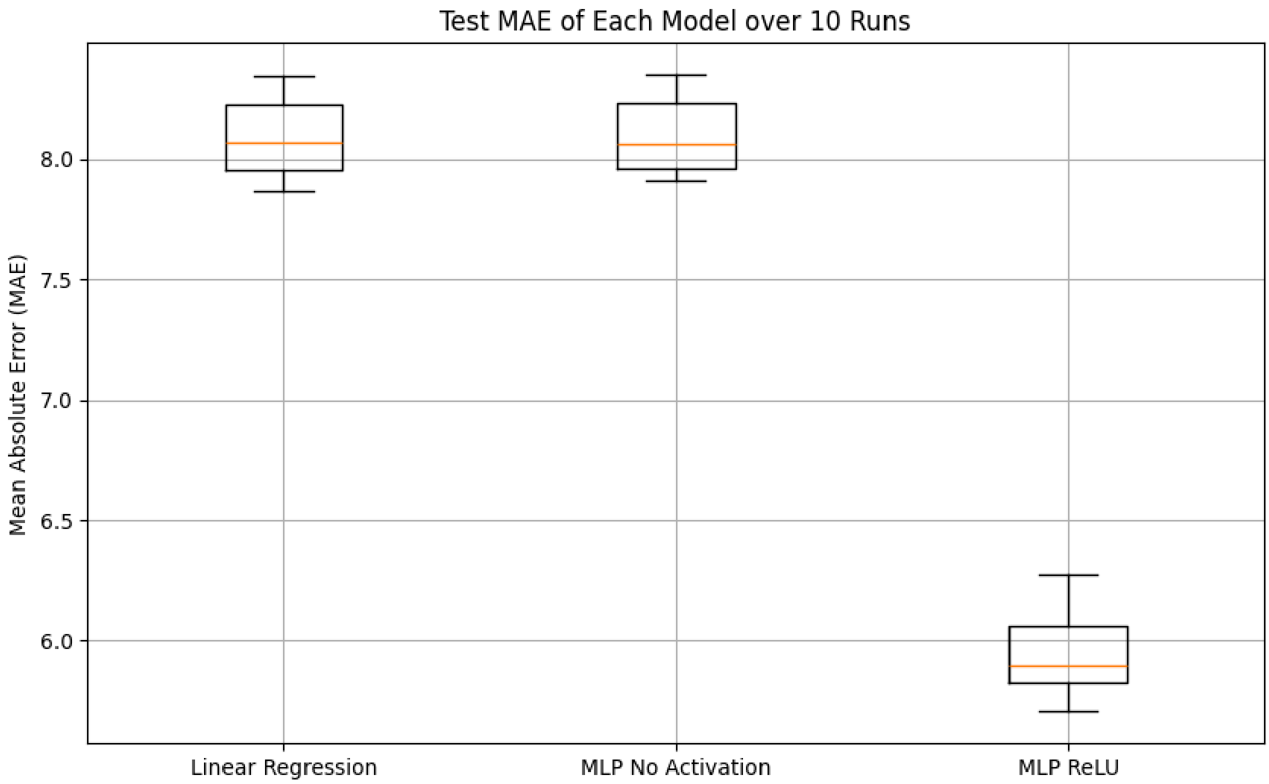
Homework I – Group XXX

(ist105940, ist, ist1106221)

II. Programming and critical analysis

5

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
import matplotlib.pyplot as plt
from warnings import filterwarnings
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

filterwarnings('ignore')

# Load the data
data = pd.read_csv("parkinsons.csv", delimiter=',')
X = data.drop('target', axis=1)
y = np.ravel(data['target'])

# Initialize variables to store MAE results
mae_linear = []
mae_mlp_no_activation = []
mae_mlp_relu = []

# Loop over 10 runs with different random train-test splits
for i in range(1, 11):
    # Split the data (80-20 split with different random states)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=i)

    # 1. Linear Regression Model
    lr_model = LinearRegression()
    lr_model.fit(X_train, y_train)
    y_pred_lr = lr_model.predict(X_test)
    mae_linear.append(mean_absolute_error(y_test, y_pred_lr))

    # 2. MLP Regressor without activation functions (identity)
    mlp_no_activation = MLPRegressor(hidden_layer_sizes=(10, 10), activation='identity', random_state=0)
    mlp_no_activation.fit(X_train, y_train)
    y_pred_mlp_no_act = mlp_no_activation.predict(X_test)
    mae_mlp_no_activation.append(mean_absolute_error(y_test, y_pred_mlp_no_act))

    # 3. MLP Regressor with ReLU activation functions
    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)
    mae_mlp_relu.append(mean_absolute_error(y_test, y_pred_mlp_relu))

# Create a DataFrame for plotting
results = pd.DataFrame({
    'Linear Regression': mae_linear,
    'MLP No Activation': mae_mlp_no_activation,
    'MLP ReLU': mae_mlp_relu
})

# Plot the boxplot using matplotlib
plt.figure(figsize=(10, 6))
plt.boxplot([results['Linear Regression'], results['MLP No Activation'], results['MLP ReLU']],
            labels=['Linear Regression', 'MLP No Activation', 'MLP ReLU'])
plt.title('Test MAE of Each Model over 10 Runs')
plt.ylabel('Mean Absolute Error (MAE)')
plt.grid(True)
plt.show()
```



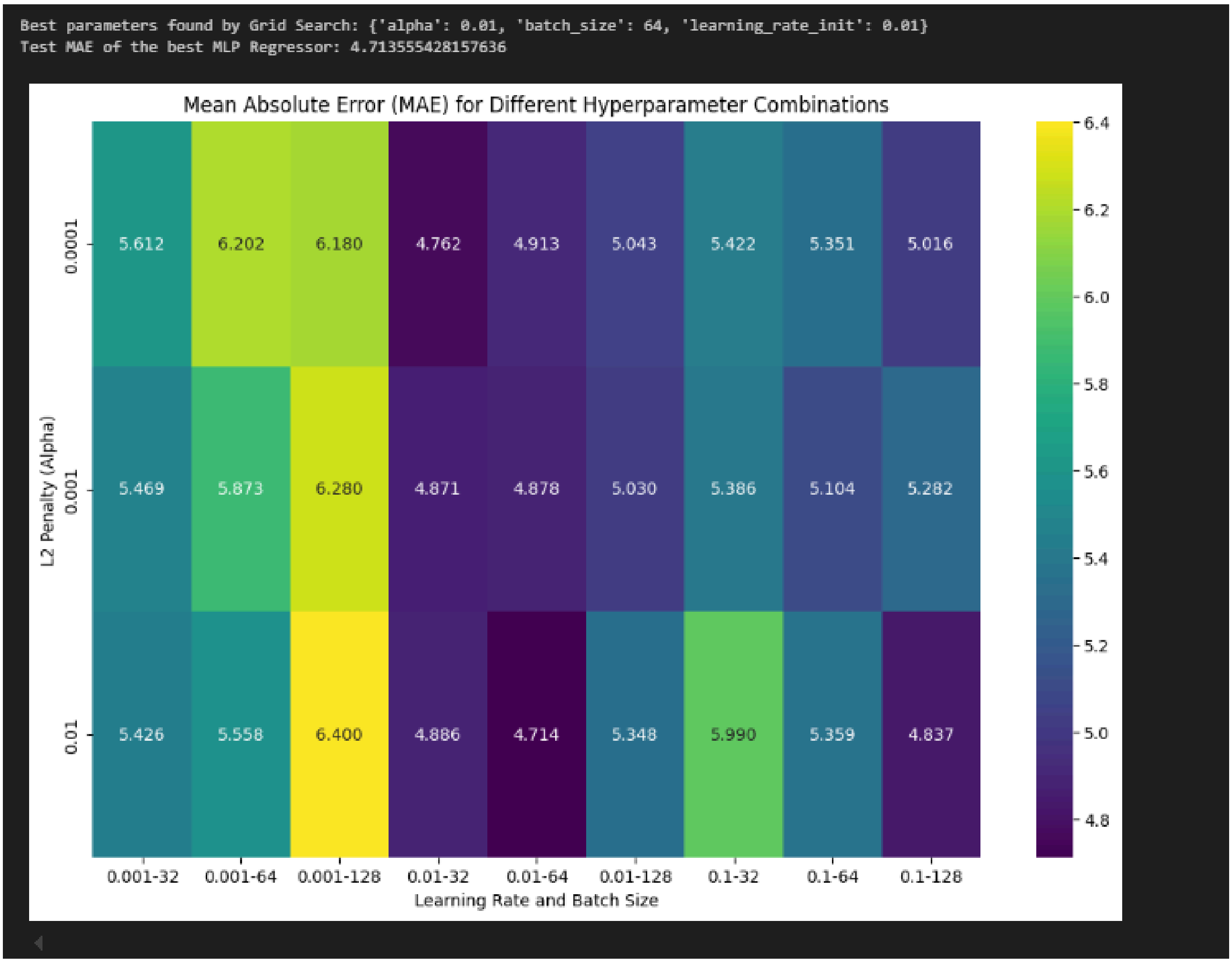
6

Comparando os resultados entre uma Regressão Linear e uma MLP sem funções de ativação, podemos verificar como as funções de ativação impactam e são cruciais nas redes neurais com base nos boxplots apresentados. A Regressão Linear é uma regressão que ajusta uma linha entre os dados, assumindo uma relação linear entre as variáveis. Como pode ser observado no box plot, o erro médio absoluto da Regressão Linear é mais ou menos igual a 8.0. Ou seja, um modelo linear tem baixa capacidade de adequar modelos de alto grau de complexidade ou não lineares, pois só ajusta esse tipo de relação linear entre variáveis. Uma MLP sem uma função de ativação, por outro lado, se assemelha a uma Regressão Linear. Isso acontece porque, sem funções de ativação, cada camada da MLP realiza uma transformação linear dos dados. Isto é, a saída de cada camada é apenas uma combinação linear de seus valores de entrada. Como resultado, uma MLP sem ativação comporta-se muito semelhante à já mencionada (Regressão Linear) e o seu MAE é de aproximadamente 8.0, como mostra o boxplot. Porém, no caso da MLP com a função de ativação ReLU, esta é a responsável pela introdução da não-linearidade no modelo. Especificamente, a não-linearidade permite que a MLP emita padrões no espaço de dados mais complexos do que apenas transformações lineares, que é de fato o caso das simples redes lineares perceptron. O gráfico revela que a MLP com a função de ativação ReLU faz uma diferença significativa para o erro médio absoluto, pois cai drasticamente para cerca de 6.0.

Em resumo, a comparação mostra que uma MLP sem a função de ativação tem desempenho semelhante ao de uma Regressão Linear, enquanto a utilização de funções de ativação, como ReLU, melhora substancialmente o desempenho da rede ao permitir a modelagem de relações não lineares.

7

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import make_scorer, mean_absolute_error
from warnings import filterwarnings
import matplotlib.pyplot as plt
import seaborn as sns

filterwarnings('ignore')

# Load the data
data = pd.read_csv("parkinsons.csv", delimiter=',')
X = data.drop('target', axis=1)
y = np.ravel(data['target'])

# Split the data (80-20 split with random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
param_grid = {
    'alpha': [0.0001, 0.001, 0.01],  # L2 penalty
    'learning_rate_init': [0.001, 0.01, 0.1],  # Learning rate
    'batch_size': [32, 64, 128],  # Batch size
}

# MLP Regressor to be tuned
mlp = MLPRegressor(hidden_layer_sizes=(10, 10))

# GridSearchCV setup
grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring='neg_mean_absolute_error', n_jobs=-1)

# Train-test split with random_state=0
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Perform the grid search
grid_search.fit(X_train, y_train)

# Get the best parameters and the corresponding test MAE

print("Best parameters found by Grid Search:", grid_search.best_params_)
print("Test MAE of the best MLP Regressor:", -1*grid_search.best_score_)

# Reshape the results for plotting
results = grid_search.cv_results_

# Create a DataFrame to organize the results
results_df = pd.DataFrame({
    'alpha': results['param_alpha'],
    'learning_rate_init': results['param_learning_rate_init'],
    'batch_size': results['param_batch_size'],
    'mean_test_score': -results['mean_test_score']  # Convert to MAE
})

# Pivot the DataFrame for heatmap plotting
heatmap_data = results_df.pivot_table(index='alpha', columns=['learning_rate_init', 'batch_size'], values='mean_test_score')

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, fmt=".3f", cmap='viridis')
plt.title('Mean Absolute Error (MAE) for Different Hyperparameter Combinations')
plt.xlabel('Learning Rate and Batch Size')
plt.ylabel('L2 Penalty (Alpha)')
plt.show()
```



Complexidade da modelagem: Aumentar o parâmetro alfa pode ser eficaz em controlar o overfiting, mas isso pode limitar a habilidade do modelo de aprender relações mais intrínsecas.

Ritmo da convergência versus estabilidade: A taxa de aprendizagem tem um papel fundamental na velocidade da adaptação do modelo aos dados. Contudo, é fundamental encontrar um meio-termo para garantir que a convergência ocorra de forma estável. Se a taxa for muito elevada, pode impedir que o modelo converja adequadamente. As mudanças mais rápidas nem sempre resultam na compreensão mais profícua, exigindo um ritmo ponderado entre velocidade e consistência.

Eficiência vs. Estabilidade: A escolha do tamanho do lote impacta tanto a velocidade do treinamento quanto a estabilidade do modelo. Lotes menores podem oferecer uma melhor capacidade de generalização, mas geralmente a um custo computacional mais elevado. Por outro lado, lotes maiores podem acelerar o processamento, mas também podem aumentar o risco de overfitting.

END