Aprendizagem 2024/2025
Homework III – Group 115
(ist106485, ist107219)

Part I: Pen and paper

Below is training dataset D composed by two input variables and two output variables, one of which is numerical (y_{num}) and the other categorical (y_{class}). Consider a polynomial basis function $\phi(y_1, y_2) = y_1 \times y_2$ that transforms the original space in to 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 continous output y_{num}.

   (a) Let us setup the regression model

   $$y_{num} = w_0 + w_1 \cdot \phi(y_1, y_2)$$

   where w_0 is the intercept, and w_1 is the slope parameter for the $\phi(y_1, y_2)$ feature

   (b) Next, we construct the *design* matrix $\mathbf{X}$ and we calculate its *pseudo-inverse* counterpart.

   $$\mathbf{X} = \begin{bmatrix} 1 & y_1^{(1)} \cdot y_2^{(1)} \\ \vdots & \vdots \\ 1 & y_1^{(j)} \cdot y_2^{(j)} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 9 \\ 1 & 8 \end{bmatrix}$$

   Now for the *pseudo-inverse* matrix $\mathbf{X}^+$,

   $$\mathbf{X}^+ = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 9 \\ 1 & 8 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix}$$

   $$= \left(\begin{bmatrix} 5 & 27 \\ 27 & 191 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix}$$

   $$= \begin{bmatrix} \frac{191}{226} & \frac{-27}{226} \\ \frac{-27}{226} & \frac{5}{226} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix}$$

   $$= \begin{bmatrix} \frac{82}{113} & \frac{55}{113} & \frac{29}{226} & \frac{-26}{113} & \frac{-25}{226} \\ \frac{-11}{113} & \frac{-6}{113} & \frac{3}{226} & \frac{9}{113} & \frac{13}{226} \end{bmatrix}$$

(c) We can now calculate the weight vector $\mathbf{w}$ using the target vector $\mathbf{y}_{num}$ and complete the regression model

$$\mathbf{y}_{num} = \begin{bmatrix} 1.25 \\ 7.0 \\ 2.7 \\ 3.2 \\ 5.5 \end{bmatrix}$$

$$\begin{aligned} \mathbf{w} &= \mathbf{X}^{+} \cdot \mathbf{y}_{num} \\ &= \begin{bmatrix} \frac{3747}{1130} & \frac{257}{2260} \end{bmatrix}^T \\ \hat{y}_{num} &= \mathbf{X} \cdot \mathbf{w} \\ &= \frac{3747}{1130} + \frac{257}{2260} \cdot \phi(y1_{new}, y2_{new}) \\ &\approxeq \boxed{3.31593 + 0.11372(y1_{new} \times y2_{new})} \end{aligned}$$

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.

(a) To learn a Ridge regression we can take our previously calculated *design* matrix and calculate the new weight vector according to the RLS closed form solution.

$$\begin{aligned} \mathbf{w} &= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T \cdot \mathbf{y}_{num} \\ &= \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 9 \\ 1 & 8 \end{bmatrix} + \mathbf{I} \right)^{-1} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix} \cdot \begin{bmatrix} 1.25 \\ 7.0 \\ 2.7 \\ 3.2 \\ 5.5 \end{bmatrix} \\ &= \left(\begin{bmatrix} 6 & 27 \\ 27 & 192 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix} \cdot \begin{bmatrix} 1.25 \\ 7.0 \\ 2.7 \\ 3.2 \\ 5.5 \end{bmatrix} \\ &= \begin{bmatrix} \frac{64}{141} & \frac{-3}{47} \\ \frac{-3}{47} & \frac{2}{141} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 9 & 8 \end{bmatrix} \cdot \begin{bmatrix} 1.25 \\ 7.0 \\ 2.7 \\ 3.2 \\ 5.5 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1709}{940} \\ \frac{913}{2820} \end{bmatrix} \end{aligned}$$

(b) The ridge regression model to predict y_{num} is as follows:

$$\begin{aligned}\hat{y}_{num} &= \mathbf{X} \cdot \mathbf{w} \\ &= \frac{1709}{940} + \frac{913}{2820} \cdot \phi(y_{1_{new}}, y_{2_{new}}) \\ &\approxeq \boxed{1.81809 + 0.32376(y_{1_{new}} \times y_{2_{new}})}\end{aligned}$$

(c) How does regularization affect the learn coefficients, when comparing with the OLS method?

Answer:

We know that Ridge uses L2 regularization which tries to shrink the coefficients for a more stable model while trying to minimize predictions errors.

Observing the weights for each model, it is clearer w_0 has decreased in the Ridge model. However we also notice that w_1 has slightly increased.

We conclude that the regularization method, in general, decreases the weight coefficients, although since the model tries to fit the data as best as possible without overfitting, a slight increase in the weight coefficients might be necessary to avoid underfitting.

3. Given three new test observations and their corresponding y_{num} output $x_6 = (2, 2, 0.7)$, $x_7 = (1, 2, 1.1)$ and $x_8 = (5, 1, 2.2)$, 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.

(a) The root mean squared error can be calculated with this formula

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)}$$

(b) Let us calculate the expected values $\hat{Y}$ according to the both models first where $\hat{Y}_{train} = \{\hat{y}_{num}(y_1^{(i)}, y_2^{(i)}) \mid i \in \{1..5\}\}$ and $\hat{Y}_{test} = \{\hat{y}_{num}(y_1^{(i)}, y_2^{(i)}) \mid i \in \{6..8\}\}$:

$$\begin{aligned}\hat{y}_{num_{OLS}} &= 3.31593 + 0.11372(y_1 \times y_2) \\ \hat{y}_{num_{RLS}} &= 1.81809 + 0.32376(y_1 \times y_2)\end{aligned}$$

$$\begin{aligned}\hat{Y}_{train_{OLS}} &= \{3.42965, 3.65709, 3.99825, 4.33941, 4.22569\} \\ \hat{Y}_{train_{RLS}} &= \{2.14185, 2.78937, 3.76065, 4.73193, 4.40817\}\end{aligned}$$

$$\begin{aligned}\hat{Y}_{test_{OLS}} &= \{3.77081, 3.54337, 3.88453\} \\ \hat{Y}_{test_{RLS}} &= \{3.11313, 2.46561, 3.43689\}\end{aligned}$$

(c) Computing the train and test RMSE for the OLS model

$$\begin{aligned} Y_{train} &= \{1.25, 7.0, 2.7, 3.2, 5.5\} \\ Y_{test} &= \{0.7, 1.1, 2.2\} \end{aligned}$$

$$\begin{aligned} RMSE_{train} &= \sqrt{\frac{1}{5}\left((3.42965 - 1.25) + (3.65709 - 7.0) + (3.99825 - 2.7) + (4.33941 - 3.2) + (4.22569 - 5.5)\right)} \\ &= \sqrt{\frac{0.00009}{5}} \\ &\approxeq \boxed{0.00424} \end{aligned}$$

$$\begin{aligned} RMSE_{test} &= \sqrt{\frac{1}{3}\left((3.77081 - 0.7) + (3.54337 - 1.1) + (3.88453 - 2.2)\right)} \\ &= \sqrt{\frac{7.19871}{3}} \\ &\approxeq \boxed{1.54905} \end{aligned}$$

(d) Computing the train and test RMSE for the Ridge model

$$\begin{aligned} Y_{train} &= \{1.25, 7.0, 2.7, 3.2, 5.5\} \\ Y_{test} &= \{0.7, 1.1, 2.2\} \end{aligned}$$

$$\begin{aligned} RMSE_{train} &= \sqrt{\frac{1}{5}\left((2.14185 - 1.25) + (2.78937 - 7.0) + (3.76065 - 2.7) + (4.73193 - 3.2) + (4.40817 - 5.5)\right)} \\ &= \sqrt{\frac{-1.81803}{5}} \\ &\approxeq \boxed{0.60300} \end{aligned}$$

$$\begin{aligned} RMSE_{test} &= \sqrt{\frac{1}{3}\left((3.11313 - 0.7) + (2.46561 - 1.1) + (3.43689 - 2.2)\right)} \\ &= \sqrt{\frac{5.01563}{3}} \\ &\approxeq \boxed{1.29301} \end{aligned}$$

(e) Do the results go according to what is expected?

Answer:

OLS minimizes the sum of squared errors without introducing a penalty, which tends to fit the training data extremely well leading to a lower training RMSE, although increasing the risk of overfitting that could compromise the generalization ability of the model potentially increasing the test RMSE.

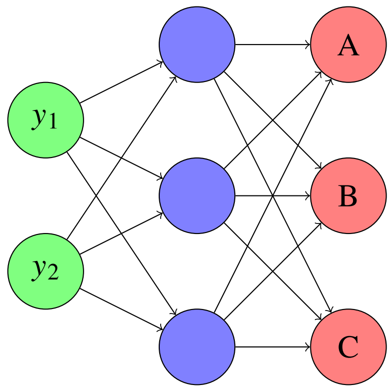
On the other hand the Ridge regression model, as we've seen before introduces penalties which improves the generalization ability, leading to a lower test RMSE but higher training RMSE in comparison with OLS.

Observing the RMSE results of our models we can affirm that the results are to be expected.

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

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

the output activation function $softmax\left(Z_c^{[out]}\right) = \frac{e^{Z_c^{[out]}}}{\sum_{l=1}^{|C|} e^{Z_l^{[out]}}}$, no activations on the hidden layers(s) and the cross-entropy loss: $-\sum_{i=1}^{N}\sum_{l=1}^{|C|} t_l^{(i)} log\left(X_l^{[out](i)}\right)$. Consider also 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.



(a) Firstly, given the training observation $x_1 = (1,1)^T$ using $\mathbf{t} = (0,1,0)^T$ as the true label, since x_1 is classified as class **B**, let us compute the forward pass:

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x}^{[0]} + \mathbf{b}^{[1]} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.2 \\ 0.2 & 0.1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.4 \end{bmatrix}$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{x}^{[1]} + \mathbf{b}^{[2]} = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2.7 \\ 2.3 \\ 2 \end{bmatrix}$$

$$\mathbf{x}^{[1]} = \mathbf{z}^{[1]} \quad \text{(no activation on hidden layers)}$$

$$\mathbf{x}^{[2]} = softmax(\mathbf{z}^{[2]}) = \frac{e^{\mathbf{z}^{[2]}}}{\sum_{l=1}^{|C|} e^{z_l^{[2]}}} = \begin{bmatrix} 0.46149 \\ 0.30934 \\ 0.22917 \end{bmatrix}$$

$$L = -\sum_{l=1}^{|C|} t_l log\left(x_l^{[2]}\right) \quad \text{(Cross-Entropy Loss with } N = 1 \text{ observation)}$$
$$= -1 \cdot log(0.30934) = 1.17330$$

(b) Now we can proceed to the backpropagation

i. Let us setup the equations to calculate the delta values to later update the weights and biases

$$\begin{aligned}\delta^{[2]} &= \frac{\partial L}{\partial \mathbf{x}^{[2]}} \cdot \frac{\partial \mathbf{x}^{[2]}}{\partial \mathbf{z}^{[2]}} \\ &= \text{(After some simplification...)} \\ &= \mathbf{x}^{[2]} - \mathbf{t} \\ &= \begin{bmatrix} 0.46149 \\ 0.30934 \\ 0.22917 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.46149 \\ -0.69066 \\ 0.22917 \end{bmatrix}\end{aligned}$$

$$\begin{aligned}\delta^{[1]} &= \mathbf{W}^{[2]^T} \cdot \delta^{[2]} \\ &= \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.46149 \\ -0.69066 \\ 0.22917 \end{bmatrix} = \begin{bmatrix} 0 \\ -0.22917 \\ 0.46149 \end{bmatrix}\end{aligned}$$

ii. Finally we can update the weights and biases

$$\begin{aligned}\mathbf{W}^{[p]} &= \mathbf{W}^{[p]} - \eta\delta^{[p]} \times \mathbf{x}^{[p-1]^T} \\ \mathbf{b}^{[p]} &= \mathbf{b}^{[p]} - \eta\delta^{[p]}\end{aligned}$$

$$\mathbf{W}^{[2]} = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} - 0.1 \begin{bmatrix} 0.46149 \\ -0.69066 \\ 0.22917 \end{bmatrix} \times \begin{bmatrix} 0.3 & 0.3 & 0.4 \end{bmatrix} = \boxed{\begin{bmatrix} 0.98616 & 1.98616 & 1.98154 \\ 1.02072 & 2.02072 & 1.02763 \\ 0.99312 & 0.99312 & 0.99083 \end{bmatrix}}$$

$$\mathbf{W}^{[1]} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.2 \\ 0.2 & 0.1 \end{bmatrix} - 0.1 \begin{bmatrix} 0 \\ -0.22917 \\ 0.46149 \end{bmatrix} \times \begin{bmatrix} 2.7 & 2.3 & 2 \end{bmatrix} = \boxed{\begin{bmatrix} 0.1 & 0.1 \\ 0.12292 & 0.22292 \\ 0.15385 & 0.05385 \end{bmatrix}}$$

$$\mathbf{b}^{[2]} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - 0.1 \begin{bmatrix} 0.46149 \\ -0.69066 \\ 0.22917 \end{bmatrix} = \boxed{\begin{bmatrix} 0.95385 \\ 1.06907 \\ 0.97708 \end{bmatrix}}$$

$$\mathbf{b}^{[1]} = \begin{bmatrix} 0.1 \\ 0 \\ 0.1 \end{bmatrix} - 0.1 \begin{bmatrix} 0 \\ -0.22917 \\ 0.46149 \end{bmatrix} = \boxed{\begin{bmatrix} 0.1 \\ 0.02292 \\ 0.05385 \end{bmatrix}}$$

Part II: Programming

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

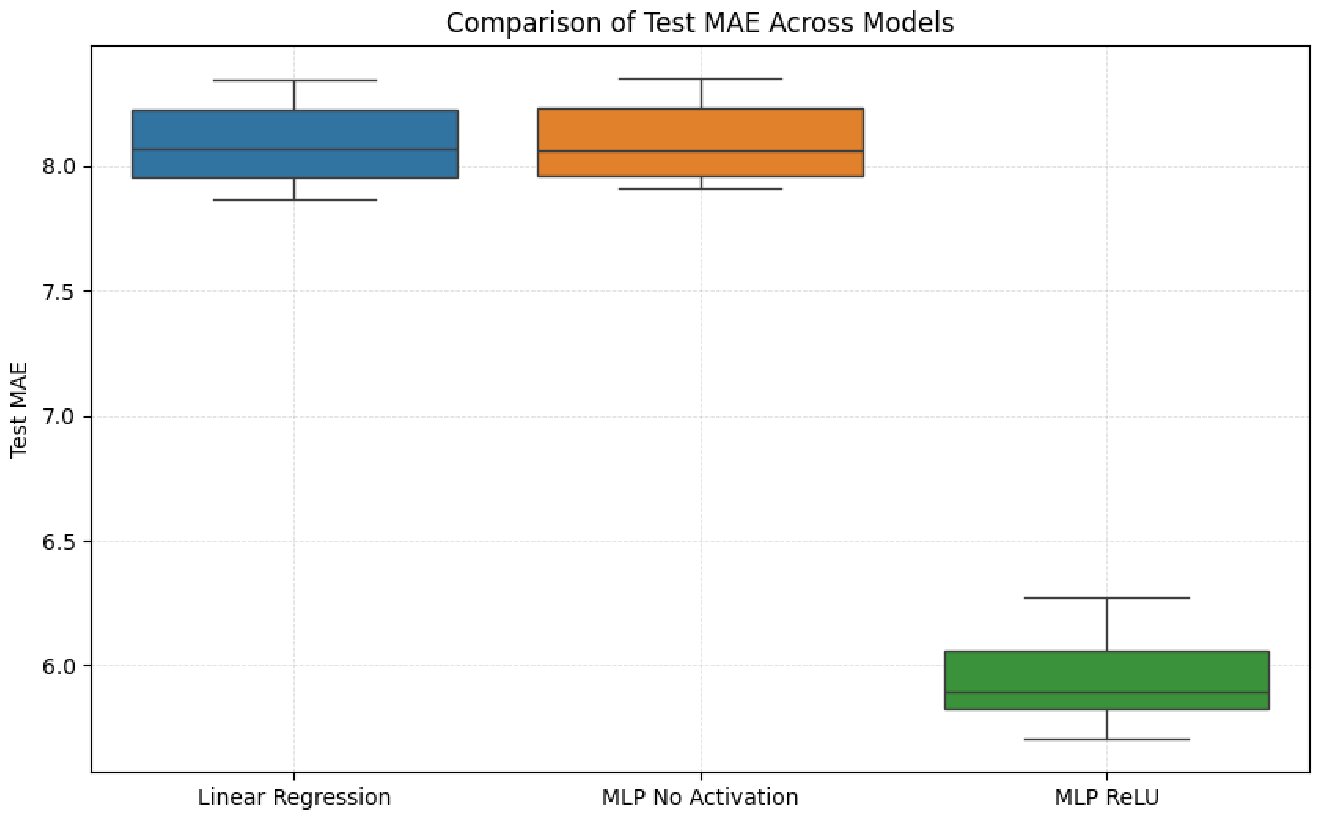


Figure 1: Comparison of Test MAE across different Models

6. 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 results from the boxplots.

Answer:

We know that a linear regressor models a linear relationship between the input variables and the target variable by minimizing a certain loss function. It is also known that in an MLP regressor with identity activation each neuron computes a linear transformation, so it works very similarly to a linear regressor.

Observing the boxplots from the previous exercise, we can indeed confirm that the performance of these two models are quite similar, with almost equal Test MAE values, indicating the linearity of the identity activation function. Furthermore, when we use a non-linear function like *ReLU* we can immediately see lower MAE values, indicating less error predicting new observations.

Therefore, we conclude that non-linear activation functions are very important in an MLP regressor, since they can help us capture complex data patterns more effectively.

7. 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.

   (a) Plot the Test MAE for each combination of hyperparameters

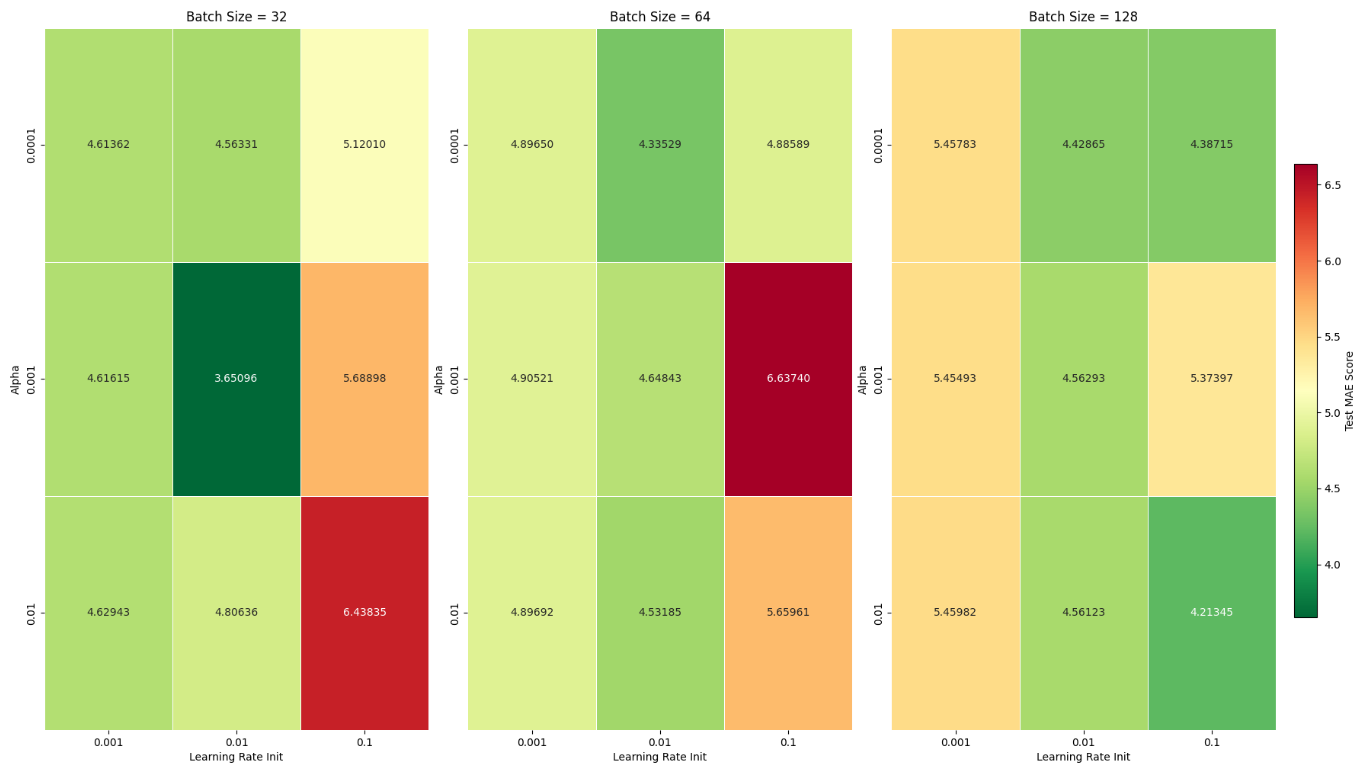


Figure 2: Test MAE Score comparison given different hyperparameters

   (b) Report the best combination
   Answer:
   The heatmap results show that the best MLP Regressor performance was obtained with alpha=0.001, learning_rate_init=0.01 and batch_size=32, with an MAE of 3.65.

   (c) Discuss the trade-offs between the combinations
   Answer:
   L2 regularization with alpha=0.001 offers a moderate level of control over the model's weights, preventing overfitting without impairing the model's ability to learn complex patterns. Higher alpha values, such as 0.01, lead to excessive penalization, resulting in lower performance, especially in combination with higher learning rates, which suggests an excessive limitation of the model's flexibility.

   On the other hand, a learning_rate_init of 0.01 effectively balances convergence speed with training stability. Higher learning rates (0.1) show a consistent tendency to increase the error, suggesting that the model becomes unstable, possibly jumping over local minima without reaching an optimal solution. Lower rates, such as 0.001, reduce this risk, but lengthen the time needed to achieve satisfactory convergence.

The batch_size also plays a key role. A value of 32 is more effective, allowing more frequent weight updates and thus better adaptation to the underlying patterns in the data. As the batch size increases to 64 and 128, a slight degradation in accuracy is observed, possibly due to the lower frequency of updates and the increased inertia in adjusting the weights, which compromises the model's ability to adapt quickly to variations in the data.