Aprendizagem 2024
Homework II – Group 053
(ist106930, ist107059)

Part I: Pen and paper

1. OLS Regression

The input variables are:

$$y_1 = [1, 1, 3, 3, 2], \quad y_2 = [1, 3, 2, 3, 4]$$

Transform the features using ϕ:

$$\phi(y_1, y_2) = [1 \times 1, 1 \times 3, 3 \times 2, 3 \times 3, 2 \times 4] = [1, 3, 6, 9, 8]$$

So the transformed feature space (matrix X), after adding the intercept term is:

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

The target values y_{num} are:

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

Now we can compute the OLS solution by following this formula:

$$w = (X^T X)^{-1} X^T y$$

Where:
X^T is the transpose of the design matrix X.
$\left(X^T X\right)^{-1}$ is the inverse of the product of X^T and X.
y is the target vector.

R: OLS coefficients: [3.31593, 0.11372]

2. Ridge Regression

The input variables are:

$$y_1 = [1, 1, 3, 3, 2], \quad y_2 = [1, 3, 2, 3, 4]$$

Transform the features using ϕ:

$$\phi(y_1, y_2) = [1 \times 1, 1 \times 3, 3 \times 2, 3 \times 3, 2 \times 4] = [1, 3, 6, 9, 8]$$

So the transformed feature space (matrix X), after adding the intercept term is:

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

The target values y_{num} are:

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

Now we can compute the Ridge regression with penalty factor $\lambda = 1$ by following the formula below:

$$w = \left(X^T X + \lambda \cdot I\right)^{-1} X^T y$$

Where:
X^T is the transpose of the design matrix X.
I is the identity matrix of size equal to the number of features (including the intercept).
$\lambda = 1$ is the regularization parameter.
y is the target vector.

I matrix would look like this:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Ridge coefficients ($\lambda = 1$): [1.81809, 0.32376]

R:OLS regression allowed the intercept to grow larger (3.316), while assigning a small weight (0.114) to the interaction feature $\phi(y_1, y_2)$. This suggests the OLS model might be more reliant on the intercept to minimize errors.

Ridge regression with $\lambda = 1$ shrunk the intercept (1.818) and shifted more importance to the interaction term (0.324). This suggests that the Ridge model distributes the weight more evenly between the bias and the feature, potentially leading to better performance on new data.

Regularization in Ridge regression reduces the magnitude of coefficients, promoting more balanced models that generalize better. However, in this case, the slope increased while the intercept shrank, which indicates that the interaction term gained importance when the intercept was penalized.

3. Test RMSE of both models

Training Data:

$y_1 = [1, 1, 3, 3, 2]$

$y_2 = [1, 3, 2, 3, 4]$

Target output:

$y_{\text{num\_train}} = [1.25, 7.0, 2.7, 3.2, 5.5]$

Test Data:

$y_1 = [2, 1, 5]$

$y_2 = [2, 2, 1]$

Target output:

$y_{\text{num\_test}} = [0.7, 1.1, 2.2]$

Construct the Design Matrices:

For Training Data:

$\phi_{\text{train}} = y_1 \cdot y_2$

$$X_{\text{train}} = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 9 \\ 1 & 8 \end{bmatrix}$$

For Test Data:

$\phi_{\text{test}} = y_1 \cdot y_2$

$$X_{\text{test}} = \begin{bmatrix} 1 & 4 \\ 1 & 2 \\ 1 & 5 \end{bmatrix}$$

Calculate the Coefficients:

OLS Coefficients:

$$w_{\text{ols}} = \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} X_{\text{train}}^T y_{\text{num\_train}}$$

Ridge Coefficients with $\lambda = 1$:

$$w_{\text{ridge}} = \left(X_{\text{train}}^T X_{\text{train}} + \lambda I\right)^{-1} X_{\text{train}}^T y_{\text{num\_train}}$$

Predictions:

For OLS Model:
Predictions on training data:

$$y_{\text{pred\_train\_ols}} = X_{\text{train}} w_{\text{ols}}$$

Predictions on test data:

$$y_{\text{pred\_test\_ols}} = X_{\text{test}} w_{\text{ols}}$$

For Ridge Model:
Predictions on training data:

$$y_{\text{pred\_train\_ridge}} = X_{\text{train}} w_{\text{ridge}}$$

Predictions on test data:

$$y_{\text{pred\_test\_ridge}} = X_{\text{test}} w_{\text{ridge}}$$

After the calculus, i obtained these predictions:

OLS:

$$\text{Training Predictions: } y_{\text{pred\_train\_ols}} = \begin{bmatrix} 3.42965 \\ 3.65708 \\ 3.99823 \\ 4.33938 \\ 4.22567 \end{bmatrix}$$

$$\text{Testing Predictions: } y_{\text{pred\_test\_ols}} = \begin{bmatrix} 3.77080 \\ 3.54336 \\ 3.88451 \end{bmatrix}$$

Ridge:

$$\text{Training Predictions: } y_{\text{pred\_train\_ridge}} = \begin{bmatrix} 2.14184 \\ 2.78936 \\ 3.76063 \\ 4.73191 \\ 4.40816 \end{bmatrix}$$

$$\text{Testing Predictions: } y_{\text{pred\_test\_ridge}} = \begin{bmatrix} 3.11312 \\ 2.46560 \\ 3.43688 \end{bmatrix}$$

Lastly, we need to do the RSME calculation:

OLS RMSE (Training and Test):

$$\text{RMSE}_{\text{train\_ols}} = \text{rmse}(y_{\text{num\_train}}, y_{\text{pred\_train\_ols}})$$

$$\text{RMSE}_{\text{test\_ols}} = \text{rmse}(y_{\text{num\_test}}, y_{\text{pred\_test\_ols}})$$

Ridge RMSE (Training and Test):

$$\text{RMSE}_{\text{train\_ridge}} = \text{rmse}(y_{\text{num\_train}}, y_{\text{pred\_train\_ridge}})$$

$$\text{RMSE}_{\text{test\_ridge}} = \text{rmse}(y_{\text{num\_test}}, y_{\text{pred\_test\_ridge}})$$

Finally, we obtained these RMSE results:

OLS RMSE (Training and Test):

$$\text{RMSE}_{\text{train\_ols}} = 2.02650$$

$$\text{RMSE}_{\text{test\_ols}} = 2.46559$$

Ridge RMSE (Training and Test):

$$\text{RMSE}_{\text{train\_ridge}} = 2.15354$$

$$\text{RMSE}_{\text{test\_ridge}} = 1.75289$$

Analysing the results obtained:

RMSE Analysis:

OLS shows lower training RMSE (2.03) compared to Ridge (2.15), indicating better fit to training data. Ridge achieves a lower test RMSE (1.75) than OLS (2.47), demonstrating better generalization to unseen data.

Prediction Patterns:

OLS predictions are generally higher, reflecting its tendency to fit the training data closely, while Ridge provides more conservative predictions. Ridge's test predictions align more closely with the target values, reinforcing its effectiveness in preventing overfitting.

Bias-Variance Tradeoff:

OLS risks overfitting (higher test RMSE), while Ridge balances bias and variance effectively due to its regularization penalty.

Conclusion:

The results align with expectations: OLS fits training data better, but Ridge generalizes more effectively, highlighting the importance of regularization in regression models.

4. MLP

Training Data

Input features:

$$x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

True labels (Class B):

$$y_{\text{true}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Initialize Weights and Biases

Weights (layer 1):

$$W_1 = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.1 \end{bmatrix}$$

Weights (layer 2):

$$W_2 = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix}$$

Biases (layer 1):

$$b_1 = \begin{bmatrix} 0.1 \\ 0.0 \\ 0.1 \end{bmatrix}$$

Biases (layer 2):

$$b_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Forward Pass

Calculate Hidden Layer Output

The pre-activation values for the hidden layer were computed as:

$$Z[1] = W_1 \cdot x_1 + b_1$$

After performing matrix multiplication and addition, I obtained:

$$Z[1] = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.4 \end{bmatrix}$$

Calculate Output Layer Pre-Activation
The output layer pre-activation values were calculated using:

$$Z[2] = W_2 \cdot Z[1] + b_2$$

This computation resulted in:

$$Z[2] = \begin{bmatrix} 2.7 \\ 2.3 \\ 2 \end{bmatrix}$$

Apply Softmax Activation
To obtain class probabilities, I applied the softmax function:

$$y_{\text{pred}} = \text{softmax}(Z[2])$$

This gave me the predicted probabilities:

$$y_{\text{pred}} \approx \begin{bmatrix} 0.46149 \\ 0.30934 \\ 0.22918 \end{bmatrix}$$

Cross-Entropy Loss Calculation
I calculated the cross-entropy loss to evaluate the model's performance:

$$L = -\sum(y_{\text{true}} \cdot \log(y_{\text{pred}}))$$

Using the predicted probabilities and the true labels, I computed: Cross-Entropy Loss ≈ 1.17330

Backward Pass

In the backward pass, I computed the gradients required to update the weights and biases.

Gradient w.r.t. Z[2] Output Layer Pre-Activation
The gradient of the loss concerning the output layer's pre-activation values was calculated as:

$$dZ[2] = y_{\text{pred}} - y_{\text{true}}$$

This gave me:

$$dZ[2] \approx \begin{bmatrix} 0.46149 \\ -0.69066 \\ 0.22918 \end{bmatrix}$$

Gradients for Weights and Biases of the Output Layer

The gradients for the weights were computed as:

$$dW[2] = dZ[2] \cdot Z[1]^T$$

The gradient for the biases was simply:

$$db[2] = dZ[2]$$

Gradient w.r.t. Z[2] (propagating back to hidden layer)

The gradient was propagated back to the hidden layer:

$$dZ[1] = W_2^T \cdot dZ[2]$$

This resulted in:

$$dZ[1] \approx \begin{bmatrix} 0 \\ -0.22918 \\ -0.46149 \end{bmatrix}$$

Gradients for Weights and Biases of the Hidden Layer

The gradients for the weights of the hidden layer were calculated as:

$$dW[1] = dZ[1] \cdot x_1^T$$

The gradient for the biases was:

$$db[1] = dZ[1]$$

Weight and Bias Updates

Using a learning rate of 0.1, I updated the weights and biases as follows:

Update Output Layer Weights and Biases:

$$W_2 \leftarrow W_2 - \eta \cdot dW[2]$$

$$b_2 \leftarrow b_2 - \eta \cdot db[2]$$

Update Hidden Layer Weights and Biases:

$$W_1 \leftarrow W_1 - \eta \cdot dW[1]$$

$$b_1 \leftarrow b_1 - \eta \cdot db[1]$$

Finally i obtained the updated Weights and Biases:

Updated W_1:

$$W_1 = \begin{bmatrix} 0.1 & 0.1 \\ 0.12292 & 0.22292 \\ 0.15385 & 0.05385 \end{bmatrix}$$

Updated b_1:

$$b_1 = \begin{bmatrix} 0.1 \\ 0.02292 \\ 0.05385 \end{bmatrix}$$

Updated W_2:

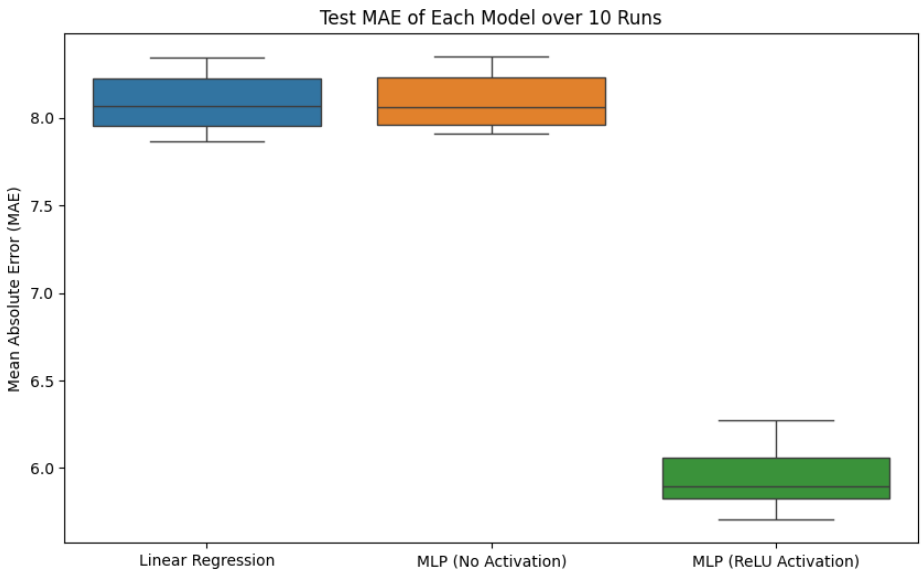
$$W_2 = \begin{bmatrix} 0.98616 & 1.98616 & 1.98154 \\ 1.02072 & 2.02072 & 1.02763 \\ 0.99312 & 0.99312 & 0.99083 \end{bmatrix}$$

Updated b_2:

$$b_2 = \begin{bmatrix} 0.95385 \\ 1.06907 \\ 0.97708 \end{bmatrix}$$

Part II: Programming

5. Graphic for exercise 5

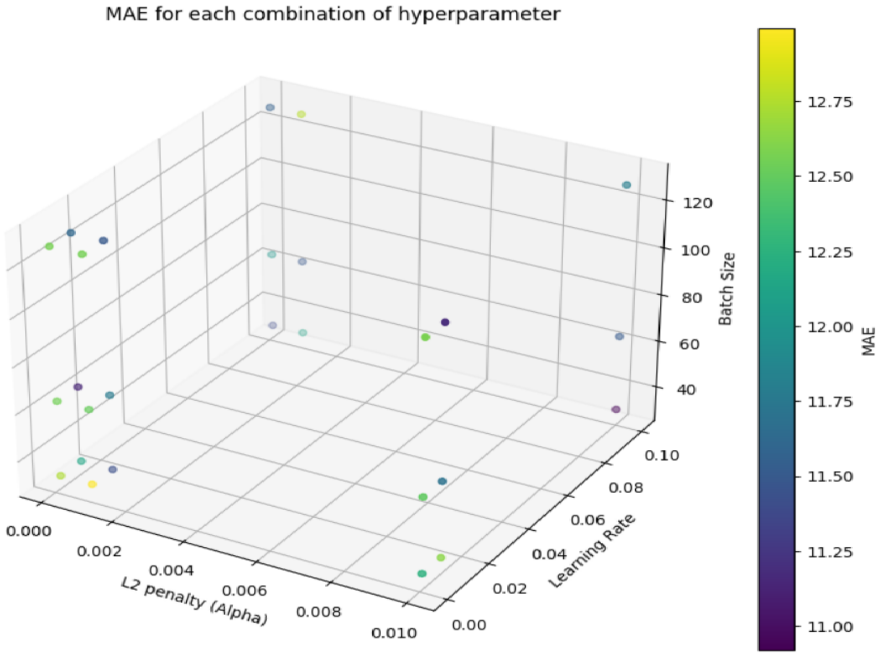


6. Compare a Linear Regression with a MLP with no activations

   An MLP without activation functions behaves exactly like a Linear Regression (as we can see by analyzing the experimental results), limiting its ability to model non-linear patterns. Using activation functions in neural networks is important because it introduces non-linearity, allowing the MLP to learn more complex patterns and, consequently, perform better on non-linear tasks.

   If we were to use an MLP with activation functions like ReLU, we would expect a reduction in error (MAE), as the model would be better able to capture the complexity of the data.

7. Graphic for exercise 7.



   Best hyperparameter combination:

   $$\{\text{alpha} : 0.01,\ \text{batch\_size} : 32,\ \text{learning\_rate\_init} : 0.1\}$$

1. L2 Penalty (Alpha):

Low (0.0001): Flexible but with a higher risk of overfitting.
Moderate (0.001): A balance between flexibility and regularization.
High (0.01): Reduces overfitting but may lead to underfitting.

2. Learning Rate:

Low (0.001): More precise convergence but slower.
Moderate (0.01): A good balance between speed and stability.
High (0.1): Fast convergence but with a risk of instability.

3. Batch Size:

Small (32): More frequent updates but higher variance in updates.
Moderate (64): A good balance between updates and stability.
Large (128): Stable convergence but with fewer updates.

Best Combination:
Alpha: 0.01; Batch Size: 32; Learning Rate: 0.1

This combination offers a good balance between regularization, stability in learning, and frequent updates.