Machine Learning - Homework 3

Pedro Curvo (ist1102716) | Salvador Torpes (ist1102474)

1st Term - 23/24

Pen and Paper Exercises

1st Question

Dataset

In this exercise we aim to learn a regression model for the following dataset:

Observation	x_0	x_1	x_2	output - z
$\vec{x}_1$	1	0.7	-0.3	0.8
$\vec{x}_2$	1	0.4	0.5	0.6
$\vec{x}_3$	1	-0.2	0.8	0.3
$\vec{x}_4$	1	-0.4	0.3	0.3

Table 1: Dataset

$$X = \begin{bmatrix} 1 & 0.7 & -0.3 \\ 1 & 0.4 & 0.5 \\ 1 & -0.2 & 0.8 \\ 1 & -0.4 & 0.3 \end{bmatrix} \qquad Z = \begin{bmatrix} 0.8 \\ 0.6 \\ 0.3 \\ 0.3 \end{bmatrix}$$

$$\vec{x}_1 = \begin{bmatrix} 0.7 \\ -0.3 \end{bmatrix} \qquad \vec{x}_2 = \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix} \qquad \vec{x}_3 = \begin{bmatrix} -0.2 \\ 0.8 \end{bmatrix} \qquad \vec{x}_4 = \begin{bmatrix} -0.4 \\ 0.3 \end{bmatrix}$$

a)

Transforming the data

We are transforming our original data into a new space, according to the radial basis function:

$$\phi_j(\vec{x}) = \exp\left(-\frac{||\vec{x}-c_j||^2}{2}\right)$$

$$c_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad c_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad c_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

After applying the transformation, we will have 3 new inputs for each observation. Therefore, the new dataset will look like:

$$\Phi(X) = X_{trans} = \begin{bmatrix} 1 & \phi_1(\vec{x}_1) & \phi_2(\vec{x}_1) & \phi_3(\vec{x}_1) \\ 1 & \phi_1(\vec{x}_2) & \phi_2(\vec{x}_2) & \phi_3(\vec{x}_2) \\ 1 & \phi_1(\vec{x}_3) & \phi_2(\vec{x}_3) & \phi_3(\vec{x}_3) \\ 1 & \phi_1(\vec{x}_4) & \phi_2(\vec{x}_4) & \phi_3(\vec{x}_4) \end{bmatrix}$$

Observation 1 If we apply our transformation to the first observation $\vec{x}_1$, we get:

$$\phi_1(\vec{x}_1) = \exp\left(-\frac{||\vec{x}_1-c_1||^2}{2}\right) = \exp\left(-\frac{0.58}{2}\right) = 0.74826$$

$$\phi_2(\vec{x}_1) = \exp\left(-\frac{||\vec{x}_1-c_2||^2}{2}\right) = \exp\left(-\frac{0.58}{2}\right) = 0.74826$$

$$\phi_3(\vec{x}_1) = \exp\left(-\frac{||\vec{x}_1-c_3||^2}{2}\right) = \exp\left(-\frac{4.58}{2}\right) = 0.10127$$

Observation 2 If we apply our transformation to the second observation $\vec{x}_2$, we get:

$$\phi_1(\vec{x}_2) = \exp\left(-\frac{||\vec{x}_2-c_1||^2}{2}\right) = \exp\left(-\frac{0.41}{2}\right) = 0.81465$$

$$\phi_2(\vec{x}_2) = \exp\left(-\frac{||\vec{x}_2-c_2||^2}{2}\right) = \exp\left(-\frac{2.61}{2}\right) = 0.27117$$

$$\phi_3(\vec{x}_2) = \exp\left(-\frac{||\vec{x}_2-c_3||^2}{2}\right) = \exp\left(-\frac{2.21}{2}\right) = 0.33121$$

Observation 3 If we apply our transformation to the third observation $\vec{x}_3$, we get:

$$\phi_1(\vec{x}_3) = \exp\left(-\frac{||\vec{x}_3-c_1||^2}{2}\right) = \exp\left(-\frac{0.68}{2}\right) = 0.71177$$

$$\phi_2(\vec{x}_3) = \exp\left(-\frac{||\vec{x}_3-c_2||^2}{2}\right) = \exp\left(-\frac{4.68}{2}\right) = 0.09633$$

$$\phi_3(\vec{x}_3) = \exp\left(-\frac{||\vec{x}_3-c_3||^2}{2}\right) = \exp\left(-\frac{0.68}{2}\right) = 0.71177$$

Observation 4 If we apply our transformation to the fourth observation $\vec{x}_4$, we get:

$$\phi_1(\vec{x}_4) = \exp\left(-\frac{||\vec{x}_4 - c_1||^2}{2}\right) = \exp\left(-\frac{0.25}{2}\right) = 0.88250$$

$$\phi_2(\vec{x}_4) = \exp\left(-\frac{||\vec{x}_4 - c_2||^2}{2}\right) = \exp\left(-\frac{3.65}{2}\right) = 0.16122$$

$$\phi_3(\vec{x}_4) = \exp\left(-\frac{||\vec{x}_4 - c_3||^2}{2}\right) = \exp\left(-\frac{0.85}{2}\right) = 0.65377$$

Transformed Dataset

After applying the transformation, we get the following dataset:

$$\Phi(X) = X_{trans} = \begin{bmatrix} 1 & 0.74826 & 0.74826 & 0.10127 \\ 1 & 0.81465 & 0.27117 & 0.33121 \\ 1 & 0.71177 & 0.09633 & 0.71177 \\ 1 & 0.88250 & 0.16122 & 0.65377 \end{bmatrix}$$

Observation	ϕ_0	ϕ_1	ϕ_2	ϕ_3	output - z
$\vec{x}_1$	1	0.74826	0.74826	0.10127	0.8
$\vec{x}_2$	1	0.81465	0.27117	0.33121	0.6
$\vec{x}_3$	1	0.71177	0.09633	0.71177	0.3
$\vec{x}_4$	1	0.88250	0.16122	0.65377	0.3

Table 2: Transformed Dataset

Ridge Regression

A regression model is characterized by a column matrix of weights W - if we multiply W by a new observation, we get the estimated output for that observation.

$$\hat{z} = w_0 + \sum_{j=1}^{M} w_j x_j = X \cdot W$$

X is the matrix of observations, and W is the matrix of weights:

$$X = \begin{bmatrix} 1 & \vec{x}_1^T \\ 1 & \vec{x}_2^T \\ 1 & \vec{x}_3^T \\ 1 & \vec{x}_4^T \end{bmatrix} \qquad W = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

When considering the case where we **transform** our data according to a function ϕ, the regression formula is:

$$\hat{z} = w_0 + \sum_{j=1}^{M} w_j \phi_j(x) = \Phi(X) \cdot W$$

The Ridge Regression (l_2 regularization) is a method that penalizes the weights of the model, in order to avoid overfitting. The formula for W matrix in the Ridge Regression is:

$$W = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T Z$$

Where λ is the regularization parameter ($\lambda = 0.1$), I is the identity matrix and Φ is the matrix of transformed observations.

Computing the weights

Using the formula for W, we get:

$$\Phi = \begin{bmatrix} 1 & 0.74826 & 0.74826 & 0.10127 \\ 1 & 0.81465 & 0.27117 & 0.33121 \\ 1 & 0.71177 & 0.09633 & 0.71177 \\ 1 & 0.88250 & 0.16122 & 0.65377 \end{bmatrix} \quad Z = \begin{bmatrix} 0.8 \\ 0.6 \\ 0.3 \\ 0.3 \end{bmatrix} \quad \Phi^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.74826 & 0.81465 & 0.71177 & 0.88250 \\ 0.74826 & 0.27117 & 0.09633 & 0.16122 \\ 0.10127 & 0.33121 & 0.71177 & 0.65377 \end{bmatrix}$$

$$(\Phi^T \Phi - \lambda I)^{-1} = \begin{bmatrix} 4.54826 & -3.77682 & -1.86117 & -1.86155 \\ -3.77682 & 5.98285 & -0.88543 & -1.26432 \\ -1.86117 & -0.88543 & 4.33276 & 2.72156 \\ -1.86155 & -1.26432 & 2.72156 & 4.53204 \end{bmatrix}$$

$$W = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T Z = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 0.33914 \\ 0.19945 \\ 0.40096 \\ -0.29600 \end{bmatrix}$$

Final form of the prediction function

In order to compute $\hat{z}$, we need to multiply the weights by the transformed observation:

$$\hat{z} = \sum_{j=0}^{3} w_j \phi_j(x) = \Phi(X) \cdot W \Leftrightarrow$$

$$\Leftrightarrow \hat{z} = w_0 + w_1 \cdot \phi_1 + w_2 \cdot \phi_2 + w_3 \cdot \phi_3 = 0.33914 + 0.19945 \cdot \phi_1 + 0.40096 \cdot \phi_2 - 0.29600 \cdot \phi_3$$

Using our dataset, the predicted values are:

$$\hat{z} = \begin{bmatrix} 0.75844 \\ 0.51232 \\ 0.30905 \\ 0.38629 \end{bmatrix}$$

b)

The RMSE (Root Mean Squared Error) is a metric that measures the difference between the predicted values and the actual values. It is defined as:

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

Where z_i is the actual value and $\hat{z}_i$ is the predicted value. In our case, we have the following data:

z_i	$\hat{z}_i$
0.8	0.75844
0.6	0.51232
0.3	0.30905
0.3	0.38629

Table 3: Actual and Predicted Values

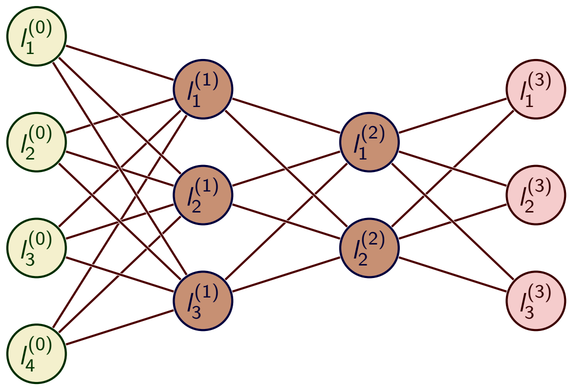
The RMSE is:

$$RMSE = \sqrt{\frac{1}{4}\sum_{i=1}^{4}(z_i - \hat{z}_i)^2} = \sqrt{\frac{1}{4} \cdot 0.01694} = 0.06508$$

2nd Question

Structure of the Network

We are considering a MLP (Multi-Layer Perceptron) with 2 hidden layers. The input and output layers have 4 and 3 nodes, respectively. This means we have 4 input features and 3 output features. Our structure is the following:



Activation Function

The activation function is the hyperbolic tangent function and it is the same for all layers:

$$\Phi(x) = f(x) = \tanh(0.5x - 2)$$

$$\Phi'(x) = f'(x) = \frac{0.5}{\cosh^2(0.5x-2)} = 0.5 \cdot \left(1 - \tanh^2(0.5x-2)\right) = 0.5 \cdot \left(1 - \Phi^2(x)\right)$$

Loss Function

The loss function is the mean square error:

$$E(W) = \frac{1}{2}\sum_{i=1}^{N} ||z_i - \hat{z}_i||^2$$

Initial Weights

We are told the initial weights are:

$$w^{[1]} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad w^{[2]} = \begin{bmatrix} 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad w^{[3]} = \begin{bmatrix} 1 & 1 \\ 3 & 1 \\ 1 & 1 \end{bmatrix}$$

$$b^{[1]} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad b^{[2]} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad b^{[3]} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Forward Propagation

According to these weights, we can compute the initial values for $X^{[1]}$, $X^{[2]}$ and $X^{[3]}$. We are considering two training observations and therefore have two different $X^{[0]}$ vectors:

$$X_1^{[0]} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad X_2^{[0]} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

With $X^{[0]}$ we can compute $X^{[1]}$, $X^{[2]}$ and $X^{[3]}$ - Propagation of both inputs through the network:

$$X_1^{[1]} = \Phi(W^{[1]} \cdot X_1^{[0]} + b^{[1]}) = \tanh\left(\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right) \cdot 0.5 - 2I\right) = \tanh\left(\begin{bmatrix} 0.5 \\ 1 \\ 0.5 \end{bmatrix}\right) = \begin{bmatrix} 0.46212 \\ 0.76159 \\ 0.46212 \end{bmatrix}$$

$$Z_1^{[1]} = W^{[1]} \cdot X_1^{[0]} + b^{[1]} = \begin{bmatrix} 5 \\ 6 \\ 5 \end{bmatrix}$$

$$X_1^{[2]} = \Phi(W^{[2]} \cdot X_1^{[1]} + b^{[2]}) = \tanh\left(\left(\begin{bmatrix} 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.46212 \\ 0.76159 \\ 0.46212 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) \cdot 0.5 - 2I\right) = \tanh\left(\begin{bmatrix} 0.45048 \\ -0.57642 \end{bmatrix}\right) = \begin{bmatrix} 0.45048 \\ -0.57642 \end{bmatrix}$$

$$Z_1^{[2]} = W^{[2]} \cdot X_1^{[1]} + b^{[2]} = \begin{bmatrix} 4.97061 \\ 2.68583 \end{bmatrix}$$

$$X_1^{[3]} = \Phi(W^{[3]} \cdot X_1^{[2]} + b^{[3]}) = \tanh\left(\left(\begin{bmatrix} 1 & 1 \\ 3 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.45048 \\ -0.57642 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right) \cdot 0.5 - 2I\right) = \tanh\left(\begin{bmatrix} -1.56297 \\ -1.11249 \\ -1.56297 \end{bmatrix}\right) = \begin{bmatrix} -0.9159 \\ -0.80494 \\ -0.9159 \end{bmatrix}$$

$$Z_1^{[3]} = W^{[3]} \cdot X_1^{[2]} + b^{[3]} = \begin{bmatrix} 0.87406 \\ 1.77503 \\ 0.87406 \end{bmatrix}$$

$$X_2^{[1]} = \Phi(W^{[1]} \cdot X_2^{[0]} + b^{[1]}) = \tanh\left(\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right) \cdot 0.5 - 2I\right) = \tanh\left(\begin{bmatrix} -1.5 \\ -1.5 \\ -1.5 \end{bmatrix}\right) = \begin{bmatrix} -0.90515 \\ -0.90515 \\ -0.90515 \end{bmatrix}$$

$$Z_2^{[1]} = W^{[1]} \cdot X_2^{[0]} + b^{[1]} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$X_2^{[2]} = \Phi(W^{[2]} \cdot X_2^{[1]} + b^{[2]}) = \tanh\left(\left(\begin{bmatrix} 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -0.90515 \\ -0.90515 \\ -0.90515 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) \cdot 0.5 - 2I\right) = \tanh\left(\begin{bmatrix} -4.21544 \\ -2.85772 \end{bmatrix}\right) = \begin{bmatrix} -0.99956 \\ -0.99343 \end{bmatrix}$$

$$Z_2^{[2]} = W^{[2]} \cdot X_2^{[1]} + b^{[2]} = \begin{bmatrix} -4.43089 \\ -1.71544 \end{bmatrix}$$

$$X_2^{[3]} = \Phi(W^{[3]} \cdot X_2^{[2]} + b^{[3]}) = \tanh\left(\left(\begin{bmatrix} 1 & 1 \\ 3 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -0.99956 \\ -0.99343 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right) \cdot 0.5 - 2I\right) = \tanh\left(\begin{bmatrix} -2.4965 \\ -3.49606 \\ -2.4965 \end{bmatrix}\right) = \begin{bmatrix} -0.98652 \\ -0.99816 \\ -0.98652 \end{bmatrix}$$

$$Z_2^{[3]} = W^{[3]} \cdot X_2^{[2]} + b^{[3]} = \begin{bmatrix} -0.993 \\ -2.99212 \\ -0.993 \end{bmatrix}$$

Output Values

Since we are working with a activation function whose range is $[-1, 1]$, the output values for both observations are:

$$t_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \qquad t_2 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

Gradient Descent

According to the gradient descent formula, in order to update the weights we need to compute the gradient of the loss function with respect to the weights. We are considering the following loss function for each observation:

$$E(W) = \frac{1}{2}||z - \hat{z}||^2 = \frac{1}{2}(z - \hat{z})^2 = \frac{1}{2}(X^{[P]} - t)^2$$

Where $z = t$ is vector of actual output values for the observation and $\hat{z} = X^{[P]}$ (P is the index of the last layer) is the vector of predicted output values for the observation. When doing the gradient descent, we need to compute the updated weights for each layer of the network. The updated weight is equal to:

$$W^{[p]}_{\text{new}} = W^{[p]} - \eta \frac{\partial E(W)}{\partial W^{[p]}}$$
$$\frac{\partial E(W)}{\partial W^{[P]}} = \delta^{[P]} \cdot \frac{\partial Z^{[P]}}{\partial W^{[P]}} = \delta^{[P]} \cdot (X^{[P-1]})^T, \text{ if } p = P \text{ (output layer)}$$
$$\frac{\partial E(W)}{\partial W^{[p]}} = \delta^{[p]} \cdot \frac{\partial Z^{[p]}}{\partial W^{[p]}} = \delta^{[p]} \cdot (X^{[p-1]})^T, \text{ if } p \neq P \text{ (hidden layer)}$$

We can compute $\delta^{[p]}$ and $\delta^{[P]}$ as:

$$\delta^{[P]} = \frac{\partial E(W)}{\partial Z^{[P]}} = \frac{\partial E(W)}{\partial X^{[P]}} \circ \frac{\partial X^{[P]}}{\partial Z^{[P]}} = (X^{[P]} - t) \circ \Phi'^{[P]}(Z^{[P]})$$
$$\delta^{[p]} = \frac{\partial E(W)}{\partial Z^{[p]}} = \left(\frac{\partial Z^{[p+1]}}{\partial X^{[p]}}\right)^T \cdot \delta^{[p+1]} \circ \frac{\partial X^{[p]}}{\partial Z^{[p]}} = (W^{[p+1]})^T \cdot \delta^{[p+1]} \circ \Phi'^{[p]}(Z^{[p]})$$

We also need to update the biases, according to the following formula:

$$b^{[p]}_{\text{new}} = b^{[p]} - \eta \frac{\partial E(W)}{\partial b^{[p]}}$$
$$\frac{\partial E(W)}{\partial b^{[p]}} = \delta^{[p]} \cdot \frac{\partial Z^{[p]}}{\partial b^{[p]}} = \delta^{[p]}$$

Therefore, we can write:

$$b^{[p]}_{\text{new}} = b^{[p]} - \eta \delta^{[p]}$$

Computing the updated weights

We are performing a batch gradient descent, therefore, the updated weight will be computed using the gradients of all observations (2 in our case):

$$W^{[n]}_{\text{new}} = W^{[n]} + \Delta W^{[n]} = W^{[n]} - \eta \sum_i \frac{\partial E(W)}{\partial W^{[n]}_i}$$

Where i is the index of the observation.

Updating $W^{[3]}$

The weight variation of $W^{[3]}$ coming from the first observation is:

$$
\begin{aligned}
\Delta W_1^{[3]} &= -\eta \frac{\partial E(W)}{\partial W_1^{[3]}} = \\
&= -\eta \cdot (X_1^{[3]} - t_1) \circ \Phi'^{[3]}(Z_1^{[3]}) \cdot (X_1^{[2]})^T = \\
&= -0.1 \cdot 0.5 \cdot (X_1^{[3]} - t_1) \circ \Big(1 - \tanh^2(Z_1^{[3]} \cdot 0.5 - 2)\Big) \cdot (X_1^{[2]})^T = \\
&= \begin{bmatrix} -0.00031 & 0.00039 \\ 0.01431 & -0.01831 \\ -0.00031 & 0.00039 \end{bmatrix}
\end{aligned}
$$

The weight variation of $W^{[3]}$ coming from the second observation is:

$$
\begin{aligned}
\Delta W_2^{[3]} &= -\eta \frac{\partial E(W)}{\partial W_2^{[3]}} = \\
&= -\eta \cdot (X_2^{[3]} - t_2) \circ \Phi'^{[3]}(Z_2^{[3]}) \cdot (X_2^{[2]})^T = \\
&= -0.1 \cdot 0.5 \cdot (X_2^{[3]} - t_2) \circ \Big(1 - \tanh^2(Z_2^{[3]} \cdot 0.5 - 2)\Big) \cdot (X_2^{[2]})^T = \\
&= \begin{bmatrix} -2.65845 \times 10^{-3} & -2.64215 \times 10^{-3} \\ 3.40000 \times 10^{-7} & 3.30000 \times 10^{-7} \\ 1.80400 \times 10^{-5} & 1.79300 \times 10^{-5} \end{bmatrix}
\end{aligned}
$$

The total weight variation of $W^{[3]}$ is:

$$
\Delta W^{[3]} = \Delta W_1^{[3]} + \Delta W_2^{[3]} = \begin{bmatrix} -0.00296 & -0.00225 \\ 0.01431 & -0.01831 \\ -0.00029 & 0.00041 \end{bmatrix}
$$

The updated weight $W^{[3]}$ is:

$$
W_{\text{new}}^{[3]} = W^{[3]} + \Delta W^{[3]} = \begin{bmatrix} 0.99704 & 0.99775 \\ 3.01431 & 0.98169 \\ 0.99971 & 1.00041 \end{bmatrix}
$$

Updating $W^{[2]}$

The weight variation of $W^{[2]}$ coming from the first observation is:

$$\begin{aligned}\Delta W_1^{[2]} &= -\eta \frac{\partial E(W)}{\partial W_1^{[2]}} = -\eta \cdot (W_1^{[3]})^T \cdot \delta_1^{[3]} \circ \Phi'^{[2]}(Z_1^{[2]}) \cdot (X_1^{[1]})^T = \\ &= -\eta \cdot (W_1^{[3]})^T \cdot \Big((X_1^{[3]} - t_1) \circ \Phi'^{[3]}(Z_1^{[3]}) \Big) \circ \Phi'^{[2]}(Z_1^{[2]}) \cdot (X_1^{[1]})^T = \\ &= -0.1 \cdot 0.5 \cdot 0.5 \cdot (W_1^{[3]})^T \cdot \Big((X_1^{[3]} - t_1) \circ \Big(1 - \tanh^2(Z_1^{[3]} \cdot 0.5 - 2)\Big)\Big) \circ \Big(1 - \tanh^2(Z_1^{[2]} \cdot 0.5 - 2)\Big) \cdot (X_1^{[1]})^T = \\ &= \begin{bmatrix} 0.01731 & 0.02852 & 0.01731 \\ 0.00469 & 0.00773 & 0.00469 \end{bmatrix}\end{aligned}$$

The weight variation of $W^{[2]}$ coming from the second observation is:

$$\begin{aligned}\Delta W_2^{[2]} &= -\eta \frac{\partial E(W)}{\partial W_2^{[2]}} = -\eta \cdot (W_2^{[3]})^T \cdot \delta_2^{[3]} \circ \Phi'^{[2]}(Z_2^{[2]}) \cdot (X_2^{[1]})^T = \\ &= -\eta \cdot (W_2^{[3]})^T \cdot \Big((X_2^{[3]} - t_2) \circ \Phi'^{[3]}(Z_2^{[3]}) \Big) \circ \Phi'^{[2]}(Z_2^{[2]}) \cdot (X_2^{[1]})^T = \\ &= -0.1 \cdot 0.5 \cdot 0.5 \cdot (W_2^{[3]})^T \cdot \Big((X_2^{[3]} - t_2) \circ \Big(1 - \tanh^2(Z_2^{[3]} \cdot 0.5 - 2)\Big)\Big) \circ \Big(1 - \tanh^2(Z_2^{[2]} \cdot 0.5 - 2)\Big) \cdot (X_2^{[1]})^T = \\ &= \begin{bmatrix} -1.04180 \times 10^{-6} & -1.04180 \times 10^{-6} & -1.04180 \times 10^{-6} \\ -1.56499 \times 10^{-5} & -1.56499 \times 10^{-5} & -1.56499 \times 10^{-5} \end{bmatrix}\end{aligned}$$

The total weight variation of $W^{[2]}$ is:

$$\Delta W^{[2]} = \Delta W_1^{[2]} + \Delta W_2^{[2]} = \begin{bmatrix} 0.0173 & 0.02852 & 0.0173 \\ 0.00468 & 0.00772 & 0.00468 \end{bmatrix}$$

The updated weight $W^{[2]}$ is:

$$W_{\text{new}}^{[2]} = W^{[2]} + \Delta W^{[2]} = \begin{bmatrix} 1.0173 & 4.02852 & 1.0173 \\ 1.00468 & 1.00772 & 1.00468 \end{bmatrix}$$

Updating $W^{[1]}$

The weight variation of $W^{[1]}$ coming from the first observation is:

$$
\begin{aligned}
\Delta W_1^{[1]} &= -\eta \frac{\partial E(W)}{\partial W_1^{[1]}} = -\eta \cdot (W_1^{[2]})^T \cdot \delta_1^{[2]} \circ \Phi'^{[1]}(Z_1^{[1]}) \cdot (X_1^{[0]})^T = \\
&= -\eta \cdot (W_1^{[2]})^T \cdot \left((W_1^{[3]})^T \cdot \delta_1^{[3]} \circ \Phi'^{[2]}(Z_1^{[2]})\right) \circ \Phi'^{[1]}(Z_1^{[1]}) \cdot (X_1^{[0]})^T = \\
&= -\eta \cdot (W_1^{[2]})^T \cdot \left((W_1^{[3]})^T \cdot \left((X_1^{[3]} - t_1) \circ \Phi'^{[3]}(Z_1^{[3]})\right) \circ \Phi'^{[2]}(Z_1^{[2]})\right) \circ \Phi'^{[1]}(Z_1^{[1]}) \cdot (X_1^{[0]})^T = \\
&= -0.1 \cdot 0.5 \cdot 0.5 \cdot 0.5 \cdot (W_1^{[2]})^T \cdot \left((W_1^{[3]})^T \cdot \left((X_1^{[3]} - t_1) \circ \left(1 - \tanh^2(Z_1^{[3]} \cdot 0.5 - 2)\right)\right) \circ \left(1 - \tanh^2(Z_1^{[2]} \cdot 0.5 - 2)\right)\right) \circ \\
&\circ \left(1 - \tanh^2(Z_1^{[1]} \cdot 0.5 - 2)\right) \cdot (X_1^{[0]})^T = \\
&= \begin{bmatrix} 0.01872 & 0.01872 & 0.01872 & 0.01872 \\ 0.03359 & 0.03359 & 0.03359 & 0.03359 \\ 0.01872 & 0.01872 & 0.01872 & 0.01872 \end{bmatrix}
\end{aligned}
$$

The weight variation of $W^{[1]}$ coming from the second observation is:

$$
\begin{aligned}
\Delta W_2^{[1]} &= -\eta \frac{\partial E(W)}{\partial W_2^{[1]}} = -\eta \cdot (W_2^{[2]})^T \cdot \delta_2^{[2]} \circ \Phi'^{[1]}(Z_2^{[1]}) \cdot (X_2^{[0]})^T = \\
&= -\eta \cdot (W_2^{[2]})^T \cdot \left((W_2^{[3]})^T \cdot \delta_2^{[3]} \circ \Phi'^{[2]}(Z_2^{[2]})\right) \circ \Phi'^{[1]}(Z_2^{[1]}) \cdot (X_2^{[0]})^T = \\
&= -\eta \cdot (W_2^{[2]})^T \cdot \left((W_2^{[3]})^T \cdot \left((X_2^{[3]} - t_2) \circ \Phi'^{[3]}(Z_2^{[3]})\right) \circ \Phi'^{[2]}(Z_2^{[2]})\right) \circ \Phi'^{[1]}(Z_2^{[1]}) \cdot (X_2^{[0]})^T = \\
&= -0.1 \cdot 0.5 \cdot 0.5 \cdot 0.5 \cdot (W_2^{[2]})^T \cdot \left((W_2^{[3]})^T \cdot \left((X_2^{[3]} - t_2) \circ \left(1 - \tanh^2(Z_2^{[3]} \cdot 0.5 - 2)\right)\right) \circ \left(1 - \tanh^2(Z_2^{[2]} \cdot 0.5 - 2)\right)\right) \circ \\
&\circ \left(1 - \tanh^2(Z_2^{[1]} \cdot 0.5 - 2)\right) \cdot (X_2^{[0]})^T = \\
&= \begin{bmatrix} 1.66619 \times 10^{-6} & 0 & 0 & -1.66619 \times 10^{-6} \\ 1.97816 \times 10^{-6} & 0 & 0 & -1.97816 \times 10^{-6} \\ 1.66619 \times 10^{-6} & 0 & 0 & -1.66619 \times 10^{-6} \end{bmatrix}
\end{aligned}
$$

The total weight variation of $W^{[1]}$ is:

$$
\Delta W^{[1]} = \Delta W_1^{[1]} + \Delta W_2^{[1]} = \begin{bmatrix} 0.01872 & 0.01872 & 0.01872 & 0.01872 \\ 0.03359 & 0.03359 & 0.03359 & 0.03359 \\ 0.01872 & 0.01872 & 0.01872 & 0.01872 \end{bmatrix}
$$

The updated weight $W^{[1]}$ is:

$$
W_{\text{new}}^{[1]} = W^{[1]} + \Delta W^{[1]} = \begin{bmatrix} 1.01872 & 1.01872 & 1.01872 & 1.01872 \\ 1.03359 & 1.03359 & 2.03359 & 1.03359 \\ 1.01872 & 1.01872 & 1.01872 & 1.01872 \end{bmatrix}
$$

Computing the updated biases

Updating $b^{[3]}$

$$\Delta b_1^{[3]} = -\eta \delta_1^{[3]} = -\eta X_1^{[3]} - t_1 \circ \Phi'^{[3]}(Z_1^{[3]}) = \begin{bmatrix} -0.00068 \\ 0.03177 \\ -0.00068 \end{bmatrix}$$

$$\Delta b_2^{[3]} = -\eta \delta_2^{[3]} = -\eta X_2^{[3]} - t_2 \circ \Phi'^{[3]}(Z_2^{[3]}) = \begin{bmatrix} 2.65961 \times 10^{-3} \\ -3.40000 \times 10^{-7} \\ -1.80500 \times 10^{-5} \end{bmatrix}$$

$$b_{\text{new}}^{[3]} = b^{[3]} + \Delta b^{[3]} = b^{[3]} + \Delta b_1^{[3]} + \Delta b_2^{[3]} = \begin{bmatrix} 1.00198 \\ 1.03177 \\ 0.9993 \end{bmatrix}$$

Updating $b^{[2]}$

$$\Delta b_1^{[2]} = -\eta \delta_1^{[2]} = -\eta (W_1^{[3]})^T \cdot \delta_1^{[3]} \circ \Phi'^{[2]}(Z_1^{[2]}) = \begin{bmatrix} 0.03745 \\ 0.01016 \end{bmatrix}$$

$$\Delta b_2^{[2]} = -\eta \delta_2^{[2]} = -\eta (W_2^{[3]})^T \cdot \delta_2^{[3]} \circ \Phi'^{[2]}(Z_2^{[2]}) = \begin{bmatrix} 1.15090 \times 10^{-6} \\ 1.72899 \times 10^{-5} \end{bmatrix}$$

$$b_{\text{new}}^{[2]} = b^{[2]} + \Delta b^{[2]} = b^{[2]} + \Delta b_1^{[2]} + \Delta b_2^{[2]} = \begin{bmatrix} 1.03745 \\ 1.01017 \end{bmatrix}$$

Updating $b^{[1]}$

$$\Delta b_1^{[1]} = -\eta \delta_1^{[1]} = -\eta (W_1^{[2]})^T \cdot \delta_1^{[2]} \circ \Phi'^{[1]}(Z_1^{[1]}) = \begin{bmatrix} 0.01872 \\ 0.03359 \\ 0.01872 \end{bmatrix}$$

$$\Delta b_2^{[1]} = -\eta \delta_2^{[1]} = -\eta (W_2^{[2]})^T \cdot \delta_2^{[2]} \circ \Phi'^{[1]}(Z_2^{[1]}) = \begin{bmatrix} 1.66619 \times 10^{-6} \\ 1.97816 \times 10^{-6} \\ 1.66619 \times 10^{-6} \end{bmatrix}$$

$$b_{\text{new}}^{[1]} = b^{[1]} + \Delta b^{[1]} = b^{[1]} + \Delta b_1^{[1]} + \Delta b_2^{[1]} = \begin{bmatrix} 1.01872 \\ 1.03359 \\ 1.01872 \end{bmatrix}$$

Results

The updated weights and biases are:

$$W_{\text{new}}^{[1]} = \begin{bmatrix} 1.01872 & 1.01872 & 1.01872 & 1.01872 \\ 1.03359 & 1.03359 & 2.03359 & 1.03359 \\ 1.01872 & 1.01872 & 1.01872 & 1.01872 \end{bmatrix} \qquad b_{\text{new}}^{[1]} = \begin{bmatrix} 1.01872 \\ 1.03359 \\ 1.01872 \end{bmatrix}$$

$$W_{\text{new}}^{[2]} = \begin{bmatrix} 1.0173 & 4.02852 & 1.0173 \\ 1.00468 & 1.00772 & 1.00468 \end{bmatrix} \qquad b_{\text{new}}^{[2]} = \begin{bmatrix} 1.03745 \\ 1.01017 \end{bmatrix}$$

$$W_{\text{new}}^{[3]} = \begin{bmatrix} 0.99704 & 0.99775 \\ 3.01431 & 0.98169 \\ 0.99971 & 1.00041 \end{bmatrix} \qquad b_{\text{new}}^{[3]} = \begin{bmatrix} 1.00198 \\ 1.03177 \\ 0.9993 \end{bmatrix}$$

Programming and Critical Analysis

Imports

```
# Sklearn Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
import warnings
```

Listing 1:

Paths and Data Loading

```
# Images
IMAGES_DIR = Path('images')
IMAGES_DIR.mkdir(parents=True, exist_ok=True)

# Data
DATA_DIR = Path('data')
DATA_DIR.mkdir(parents=True, exist_ok=True)
DATA_FILE = 'winequality-red.csv'
DATA_PATH = DATA_DIR / DATA_FILE

# Load the data
df = pd.read_csv(DATA_PATH, sep=';')

# Show the first 5 rows
df.head()
```

Listing 2:

Data Preprocessing

```
# Define features and labels
X = df.drop("quality", axis=1)
y = df["quality"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    random_state=0,
                                                    train_size=0.8)
```

Listing 3:

Question 1

```
# List for Residues
residues = []
```

```python

    # MAE list for question 2
    mae_list = []
    mae_rounded = []
    mae_bounded = []
    mae_bounded_rounded = []

    # RMSE list for question 3
    rmse_early = []

    # Loop through different values of random_state for MLPClassifier
    for random_state in range(1, 11):
        # Create the classifier
        mlp = MLPRegressor(hidden_layer_sizes=(10, 10),
                           activation='relu',
                           early_stopping=True,
                           validation_fraction=0.2,
                           random_state=random_state)

        # Fit the classifier to the training data
        mlp.fit(X_train, y_train)

        # Predict the labels of the test set
        y_pred = mlp.predict(X_test)

        # Add the residues to the list
        residues.extend(abs(y_test - y_pred).to_numpy())

        # Calculate the MAE
        mae = mean_absolute_error(y_test, y_pred)
        mae_list.append(mae)

        # Round the predictions
        y_pred_rounded = np.round(y_pred)
        mae_rounded.append(mean_absolute_error(y_test, y_pred_rounded))

        # Bound the predictions
        y_pred_bounded = np.clip(y_pred, 1, 10)
        mae_bounded.append(mean_absolute_error(y_test, y_pred_bounded))

        # Bounded after Rounded
        y_pred_bounded_rounded = np.clip(y_pred_rounded, 1, 10)
        mae_bounded_rounded.append(mean_absolute_error(y_test, y_pred_bounded_rounded))

        # Calculate the RMSE
        rmse = np.sqrt(mean_squared_error(y_test, y_pred))
        rmse_early.append(rmse)

    # Plot the distribution of residues (in absolute value)
    plt.figure(figsize=(10, 6))
    plt.hist(residues, bins='auto')
    plt.xlabel("Residue (Absolute Value)")
    plt.ylabel("Frequency (Counts)")
    plt.title("Distribution of Residues in Absolute Value")
    # Save the figure
    plt.savefig(IMAGES_DIR / "residues.png")
    plt.show()
```

Listing 4:

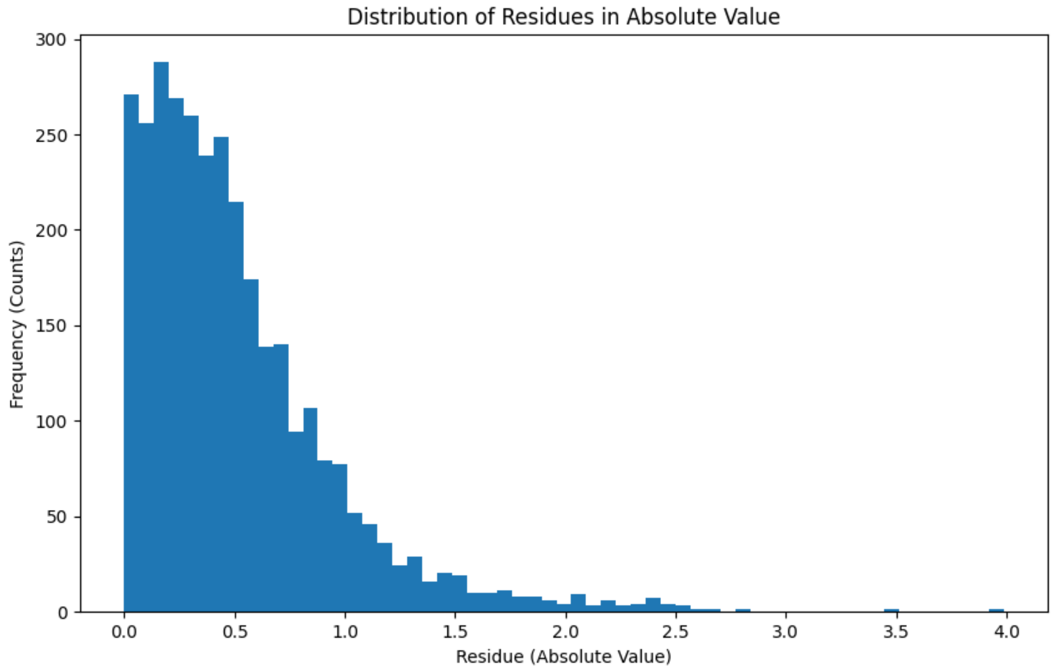


Figure 1: Distribution of Residues in Absolute Value

Comment: As one can see in the histogram above, the residues resemble to follow an exponential distribution, indicating that the majority of predictions exhibit relatively small errors, while larger prediction errors are less frequent. This suggests that the model is performing well, as the majority of predictions are close to the true values. However, it also highlights that there are instances where the model produces larger prediction errors, resulting in the exponential tail of the distribution. These larger errors are likely due to more complex or unusual points that are outside the scope of the training of the model.

Question 2

```
# Plot the MAE for the different values of random_state
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), mae_list, 'o-', label="MAE", alpha=0.5)
plt.plot(range(1, 11), mae_rounded, 'o-', label="MAE Rounded", alpha=0.5)
plt.plot(range(1, 11), mae_bounded, 'o--', label="MAE Bounded", alpha=0.5)
plt.plot(range(1, 11), mae_bounded_rounded, 'o--', label="MAE Bounded Rounded", alpha=
0.5)
plt.xlabel("Random State")
plt.ylabel("MAE")
plt.title("MAE for Different Values of Random State")
# Add the Mean MAE for each method on a text box
plt.text(1, 0.47,
         f"Mean MAE: {np.mean(mae_list):.5f} \
         \nMean MAE Rounded: {np.mean(mae_rounded):.5f} \
         \nMean MAE Bounded: {np.mean(mae_bounded):.5f} \
         \nMean MAE Bounded after Rounded: {np.mean(mae_bounded_rounded):.5f}",
         bbox=dict(facecolor='white', alpha=0.5))


plt.legend()
# Save the figure
plt.savefig(IMAGES_DIR / "mae.png")
plt.show()
```

Listing 5:

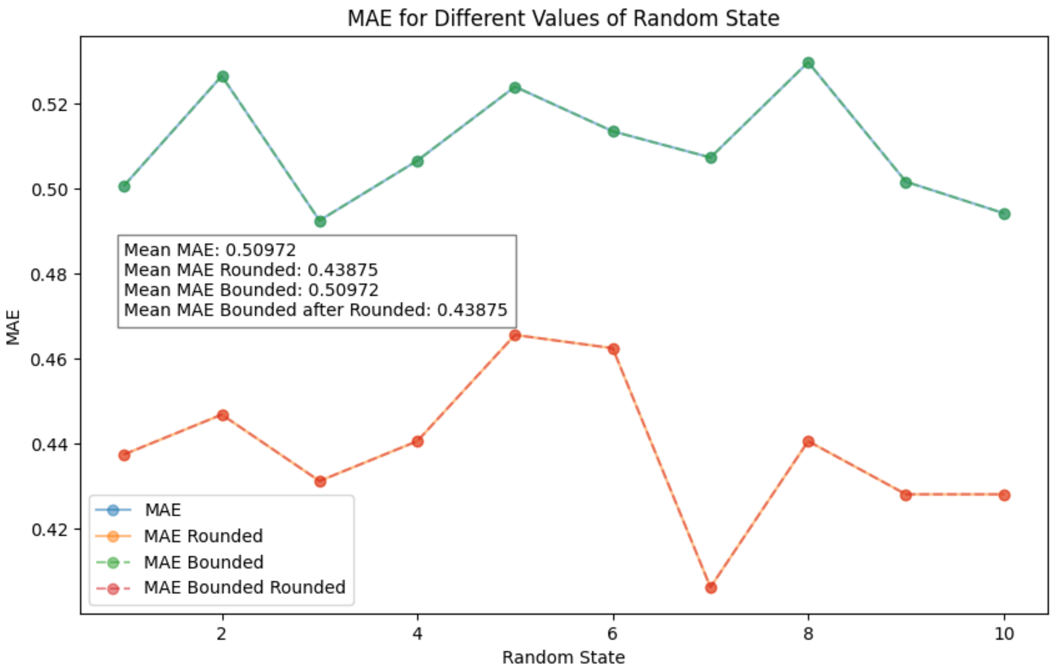


Figure 2: MAE for Different Values of Random State

Comment: As seen by the plot above and by the Mean values of the different MAE we can observe that Bounding the data do not change the MAE of the model. This indicates that there are no values for the MAE outside the bounds defined for this operation. Beyond that, one can also see that rounding the data to the nearest integer leads to a reduction in the MAE, as this operation minimizes the discrepancy between predicted and actual quality ratings. Hence, the model's performance may appear to improve, especially if the original predictions had small fractional errors. It's important to highlight that while rounding and bounding (even if in this case bounding does not change nothing at all) can improve the apparent accuracy of the model, it may lead to a loss of information. The original fractional predictions might carry valuable nuances that are discarded in the rounding and bounding process.

Question 3

```
iterations = [20, 50, 100, 200]
rmse_results_matrix = np.zeros((10, len(iterations)))

for random_state in range(1, 11):
    for num_iterations in iterations:
        # Create the classifier
        mlp = MLPRegressor(hidden_layer_sizes=(10, 10),
                        activation='relu',
                        validation_fraction=0.2,
                        max_iter=num_iterations,
                        random_state=random_state)

        # Fit the classifier to the training data
```

```
            with warnings.catch_warnings():
                warnings.simplefilter("ignore") # Ignore the ConvergenceWarning from sklearn
                mlp.fit(X_train, y_train)

            # Make predictions and calculate RMSE
            predictions = mlp.predict(X_test)
            rmse = np.sqrt(mean_squared_error(y_test, predictions))
            rmse_results_matrix[random_state - 1, iterations.index(num_iterations)] = rmse

    # Plot the RMSE for the different values of random_state
    plt.figure(figsize=(10, 6))

    # Plot the RMSE for the different values of random_state
    for i in range(4):
        plt.plot(range(1, 11), rmse_results_matrix[:, i], 'o--', label=f"{iterations[i]} Iterations", alpha=0.5)

    # Plot the mean RMSE
    plt.plot(range(1, 11), np.mean(rmse_results_matrix, axis=1), 'o-', label="Mean RMSE")

    # Graph labels
    plt.legend()
    plt.xlabel("Random State")
    plt.ylabel("RMSE")
    plt.title("RMSE for Different Values of Number of Iterations")

    # Save the figure
    plt.savefig(IMAGES_DIR / "rmse.png")
    plt.show()

    # Plot the RMSE for different values of iterations for each random state
    plt.figure(figsize=(10, 6))
    for i in range(10):
        plt.plot(iterations, rmse_results_matrix[i, :], 'o--', label=f"Random State = {i + 1}", alpha=0.5)

    # Plot the mean RMSE
    plt.plot(iterations, np.mean(rmse_results_matrix, axis=0), 'o-', label="Mean RMSE")
    plt.axhline(y=np.mean(rmse_early), color='r', linestyle='-', label="Mean RMSE Early Stopping")
    plt.legend()
    plt.xlabel("Number of Iterations")
    plt.ylabel("RMSE")
    plt.title("RMSE for Different Values of Number of Iterations for Each Random State")
    # Save the figure
    plt.savefig(IMAGES_DIR / "rmse_iterations_random_state.png")
    plt.show()
```

Listing 6:

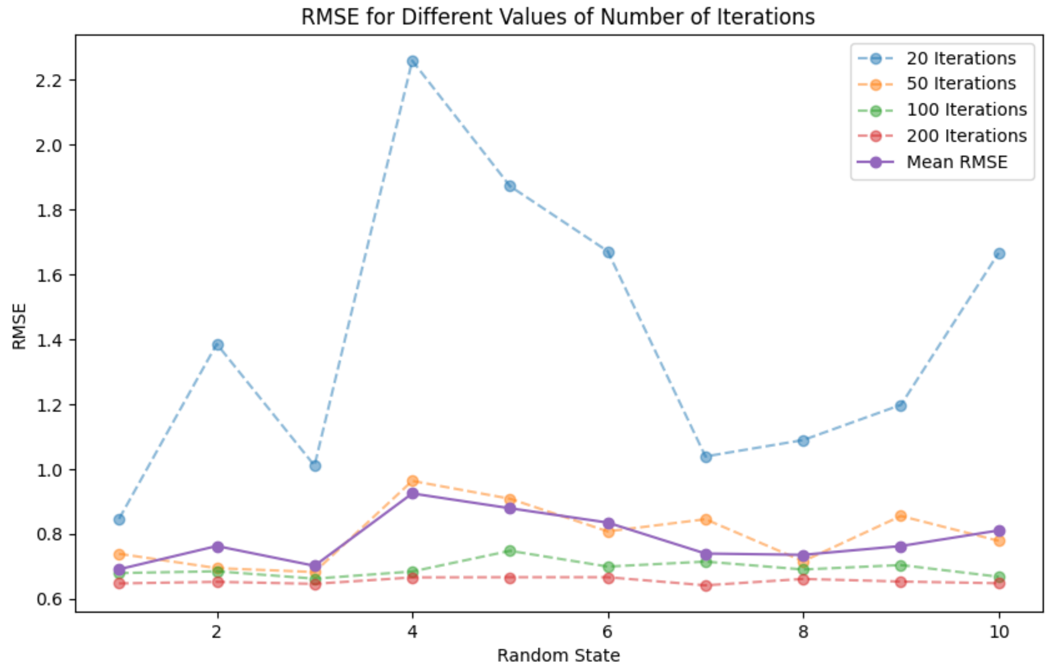


Figure 3: RMSE for Different Values of Number of Iterations

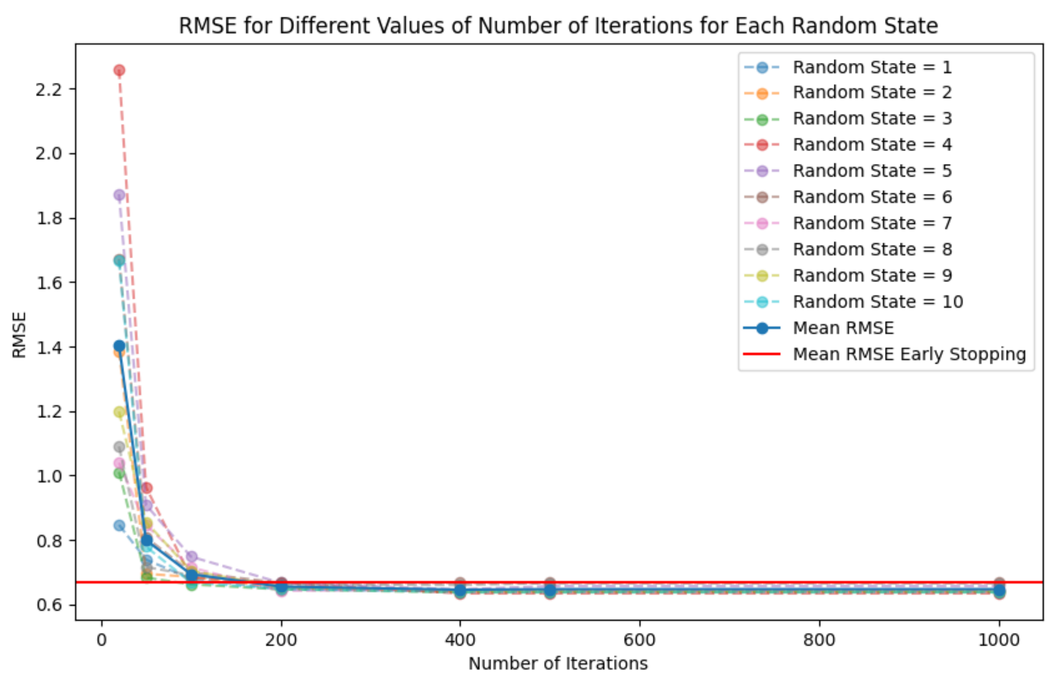


Figure 4: RMSE for Different Values of Number of Iterations for Each Random State

Question 4

First of all, one need to have in mind that Early Stopping is a technique used to prevent overfitting by monitoring the model's performance on a validation dataset during training and stopping when the performance starts to decrease. It essentially determines the optimal number of iterations automatically. In contrast, setting a specific number of iterations in advance implies that the model will be trained for a fixed number of batch iterations, regardless of its performance.

While using early stopping, the model is trained until the validation performance starts to deteriorate, at which point the training is stopped. This ensures that the model generalizes well to unseen data, improving its performance on the test dataset. In contrast, using a predefined number of iterations may lead to overfitting if the model is trained for too long, or suboptimal performance if the model is trained for too little time.

By analyzing the plot from the previous question we can see that we have two situations:

- First, one can see a Decreasing RMSE, which is expected, since the number of iterations is relative small, hence the model do not have sufficient training to converge, resulting in a higher RMSE.
- Secondly, one can see a Steady RMSE, which is also expected, since the number of iterations is appropriate to allow the model to converge to a stable solution. RMSE then remains consistent with early stopping. This is because the model has had enough training to reach a satisfactory performance level.

What one can also expect is that if the number of iterations is very large, the model might start to overfit the training data, causing RMSE to increase. This is because the model continues to improve on the training data while its generalization capability deteriorates. However, the number of iterations used in this case is not large enough to cause overfitting, so the RMSE remains steady.

In general, Early Stopping prevents overfitting by terminating training when the validation performance starts to be steady or when deteriorates. This ensures that the model generalizes well to unseen data. Beyond that, it can also be computationally efficient because it adapts the number of iterations required for training. Using a fixed number of iterations might seem a wrong choice, but it can also have advantages, since it may lead to faster convergence in some cases. This is advantageous in situations where computational resources or time constraints are limiting factors, because one can have a decreasing RMSE for a long time, but the change might be so small that it is not worth the computational effort. Hence, fixing the number of iterations will stop the training while the early stopping will continue to train the model. However, one have to have in mind, that very small number of iterations can lead to suboptimal performance and very large number of iterations can lead to overfitting.