Aprendizagem 2024/25

Homework III – Group 014

(ist106266, ist106583)

I. Pen-and-paper

1) Given the training dataset D, it is possible to define the following matrix:

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

where ϕ represents the transformation of D according to $\phi(y_1, y_2) = y_1 \times y_2$. Note that the first column was added to account for the model's bias. With this, the weights of the regression model are expressed as:

$$w = \phi^{+} y_{num} = (\phi^T \phi)^{-1} \phi^T y_{num}$$

where ϕ^{+} denotes the Moore-Penrose inverse of ϕ and y_{num} the target vector of dataset D. Further development of the equation then yields:

$$w = \begin{bmatrix} 0.72566372 & 0.48672566 & 0.12831858 & -0.2300885 & -0.11061947 \\ -0.09734513 & -0.05309735 & 0.01327434 & 0.07964602 & 0.05752212 \end{bmatrix} \begin{bmatrix} 1.25 \\ 7.0 \\ 2.7 \\ 3.2 \\ 5.5 \end{bmatrix} = \begin{bmatrix} 3.31593 \\ 0.11372 \end{bmatrix}$$

where w_1 denotes the model's bias and w_2 the weight of $\phi(y_1, y_2)$.

2) Considering the transformed dataset ϕ from the previous exercise, the weights of the Ridge regression model, with the penalty factor $\lambda = 1$, can be computed in the following manner:

$$w = (\phi^T \phi + \lambda I)^{-1} \phi^T y_{num} = (\phi^T \phi + I)^{-1} \phi^T y_{num}$$

$$w = \begin{bmatrix} 1.81809 \\ 0.32376 \end{bmatrix}$$

where w_1 denotes the model's bias and w_2 the weight of $\phi(y_1, y_2)$. While the previous regression minimizes the half squared error loss, the Ridge regression model minimizes the following loss function:

$$E(w) = \frac{1}{2} \sum_{i=1}^{n} (z_i - w^T \cdot x_i)^2 + \frac{\lambda}{2} ||w||_2^2$$

Therefore, the weights are minimized in the process of minimizing this loss function, which explains why the magnitude of the weight vector yielded by the Ridge regression is smaller than the magnitude of the weight vector learned by the previous regression model. This effect would be even more pronounced if the penalty factor was greater.

3) Before using the models to predict the test observations' y_{num} output, it is necessary to transform their features according to $\phi(y_1, y_2) = y_1 \times y_2$ and to extend them with a 1, to account for the bias:

$$x_6' = (1,\ 2 \times 2) = (1, 4)$$

$$x_7' = (1,\ 1 \times 2) = (1, 2)$$

$$x_8' = (1,\ 5 \times 1) = (1, 5)$$

The real values of y_{num} were omitted since they are not necessary for the regression process. To obtain the predicted y_{num} value for one of the given observations, x', the following suffices:

$$\hat{y}_{num} = w^T \cdot x'$$

Therefore, the first model, without regularization, yields:

$$\hat{y}_{num,6} = w^T \cdot x_6' = 3.77081$$

$$\hat{y}_{num,7} = w^T \cdot x_7' = 3.54337$$

$$\hat{y}_{num,8} = w^T \cdot x_8' = 3.88453$$

And the Ridge regression yields:

$$\hat{y}_{num,6} = w^T \cdot x_6' = 3.11313$$

$$\hat{y}_{num,7} = w^T \cdot x_7' = 2.46561$$

$$\hat{y}_{num,8} = w^T \cdot x_8' = 3.43689$$

Finally, the RMSE for each model can be computed as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{num,i} - \hat{y}_{num,i})^2}$$

OLS regression model:

$$\sum_{i=1}^{n}(y_{num,i} - \hat{y}_{num,i})^2 = (1.25 - 3.42965)^2 + (7.0 - 3.65709)^2$$

$$+ (2.7 - 3.99825)^2 + (3.2 - 4.33941)^2 + (5.5 - 4.22569)^2 = 20.53350$$

$$RMSE_{OLS,train} = \sqrt{\frac{1}{5}(20.53350)} = 2.02650$$

$$RMSE_{OLS,test} = \sqrt{\frac{1}{3}((0.7 - 3.77081)^2 + (1.1 - 3.54337)^2 + (2.2 - 3.88453)^2} = 2.46560$$

Ridge regression Model

$$\sum_{i=1}^{n}(y_{num,i}-\widehat{y}_{num,i})^2 = (1.25.-2.14185)^2+(7.0-2.78937)^2$$

$$+(2.7-3.76065)^2+(3.2-4.73193)^2+(5.5-4.40817)^2 = 23.18868$$

$$RMSE_{Ridge,train} = \sqrt{\tfrac{1}{5}(23.18868)} = 2.15354$$

$$RMSE_{Ridge,test} = \sqrt{\tfrac{1}{3}((0.7-3.11313)^2+(1.1-2.46561)^2+(2.2-3.43689)^2} = 1.75290$$

The test RMSE of the Ridge regression is clearly less than the simple regression, which shows the Ridge regression fared better than the simple regression when faced with observations outside the training data. This was expected since the regularization makes the learnt regression smoother, which should lead to better generalization capabilities. However, the train RMSE of the Ridge regression is greater than the OLS regression. This is also unsurprising since the aforementioned smoothing caused by the regularization makes the regression less fitted to the training data.

4) Given the observation x_1=(1, 1, B), with each entry representing y_1, y_2 and y_{class} respectively, it is first necessary to perform forward propagation to calculate the output of x_1 according to the MLP. This is done by computing the following equation at each layer:

$$x^{[p]} = \phi^{[p]}(W^{[p]}x^{[p-1]}+b^{[p]})$$

From the given weight matrices, it is possible to deduce that the MLP has a single hidden layer with 3 neurons, all of them without an activation function, which means that $\phi^{[1]}(x) = x$. With these facts in mind, the forward propagation phase is as follows:

$$x^{[1]} = \phi^{[1]}\left(W^{[1]}x^{[0]}+b^{[1]}\right) = W^{[1]}x^{[0]}+b^{[1]} = \begin{bmatrix}0.1 & 0.1\\0.1 & 0.2\\0.2 & 0.1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}+\begin{bmatrix}0.1\\0\\0.1\end{bmatrix} = \begin{bmatrix}0.3\\0.3\\0.4\end{bmatrix}$$

$$o = x^{[2]} = \phi^{[2]}\left(W^{[2]}x^{[1]}+b^{[2]}\right) = softmax\left(\begin{bmatrix}1 & 2 & 2\\1 & 2 & 1\\1 & 1 & 1\end{bmatrix}\begin{bmatrix}0.3\\0.3\\0.4\end{bmatrix}+\begin{bmatrix}1\\1\\1\end{bmatrix}\right) =$$

$$= softmax\left(\begin{bmatrix}2.7\\2.3\\2\end{bmatrix}\right) = \begin{bmatrix}\frac{e^{2.7}}{e^{2.7}+e^{2.3}+e^{2}}\\\frac{e^{2.3}}{e^{2.7}+e^{2.3}+e^{2}}\\\frac{e^{2}}{e^{2.7}+e^{2.3}+e^{2}}\end{bmatrix} = \begin{bmatrix}0.46149\\0.30934\\0.22917\end{bmatrix}$$

Note that $x^{[0]}$ is just the y_1 and y_2 features from x_1.

With the forward propagation finished, it is time to perform backpropagation. The update rules of MLPs are expressed as:

$$W^{[p]}_{new} = W^{[p]}-\eta\frac{\partial E}{\partial W^{[p]}},\ \frac{\partial E}{\partial W^{[p]}} = \delta^{[p]}\left(x^{[p-1]}\right)^T$$

$$b^{[p]}_{new} = b^{[p]}-\eta\frac{\partial E}{\partial b^{[p]}},\ \frac{\partial E}{\partial b^{[p]}} = \delta^{[p]}$$

Furthermore, in this specific MLP, with no activation functions on the hidden layer and a softmax activation function in the output layer along with a cross-entropy loss function, the deltas are computed as follows:

$$\delta^{[2]} = \left(x^{[2]} - t\right) = \begin{bmatrix}0.46149\\0.30934\\0.22917\end{bmatrix} - \begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}0.46149\\-0.69066\\0.22917\end{bmatrix}$$

$$\delta^{[1]} = \left(W^{[2]}\right)^T \cdot \delta^{[2]} \circ \frac{\partial\phi^{[1]}\left(z^{[1]}\right)}{\partial z^{[1]}} = \left(W^{[2]}\right)^T \cdot \delta^{[2]} \circ 1 = \left(W^{[2]}\right)^T \cdot \delta^{[2]} =$$
$$= \begin{bmatrix}1&1&1\\2&2&1\\2&1&1\end{bmatrix}\begin{bmatrix}0.46149\\-0.69066\\0.22917\end{bmatrix} = \begin{bmatrix}0\\-0.22917\\0.46149\end{bmatrix}$$

With this, it is now possible to update the weights and biases:

$$\frac{\partial E}{\partial W^{[2]}} = \delta^{[2]}\left(x^{[1]}\right)^T = \begin{bmatrix}0.46149\\-0.69066\\0.22917\end{bmatrix}\begin{bmatrix}0.3&0.3&0.4\end{bmatrix} = \begin{bmatrix}0.13845&0.13845&0.18460\\-0.20720&-0.20720&-0.27626\\0.06875&0.06875&0.09167\end{bmatrix}$$

$$\frac{\partial E}{\partial W^{[1]}} = \delta^{[1]}\left(x^{[0]}\right)^T = \begin{bmatrix}0\\-0.22917\\0.46149\end{bmatrix}\begin{bmatrix}1&1\end{bmatrix} = \begin{bmatrix}0&0\\-0.22917&-0.22917\\0.46149&0.46149\end{bmatrix}$$

$$W^{[2]}_{new} = \begin{bmatrix}1&2&2\\1&2&1\\1&1&1\end{bmatrix} - 0.1\begin{bmatrix}0.13845&0.13845&0.18460\\-0.20720&-0.20720&-0.27626\\0.06875&0.06875&0.09167\end{bmatrix} = \begin{bmatrix}0.98616&1.98616&1.98154\\1.02072&2.02072&1.02763\\0.99313&0.99313&0.99083\end{bmatrix}$$

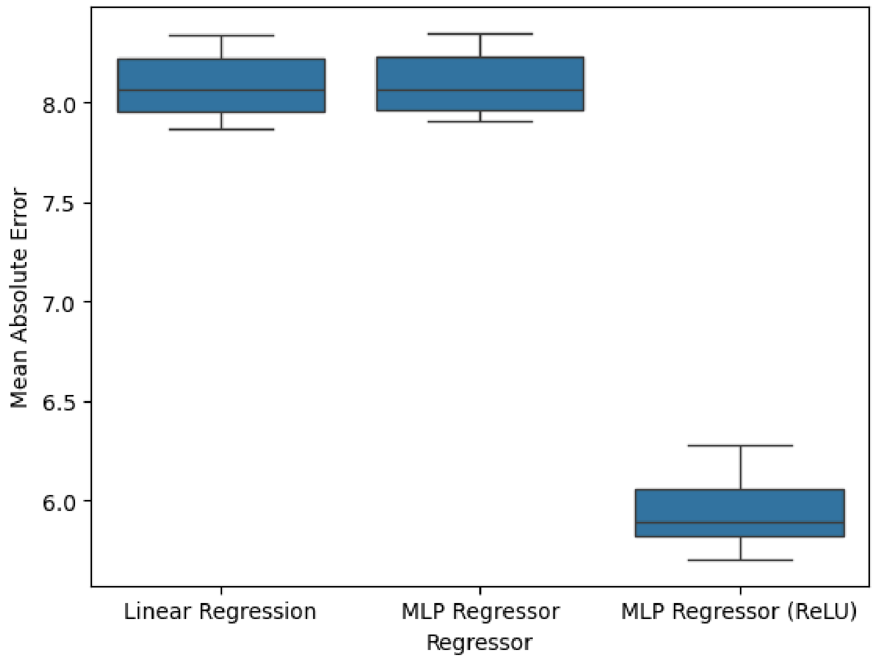
$$W^{[1]}_{new} = \begin{bmatrix}0.1&0.1\\0.1&0.2\\0.2&0.1\end{bmatrix} - 0.1\begin{bmatrix}0&0\\-0.22917&-0.22917\\0.46149&0.46149\end{bmatrix} = \begin{bmatrix}0.1&0.1\\0.12292&0.22292\\0.15385&0.05385\end{bmatrix}$$

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

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

II. Programming and critical analysis

5)



6) Since the MLP has no activation functions, the output of the model at a given layer p can be expressed in the following fashion:

$$x^{[p]} = \phi^{[p]}\left(z^{[p]}\right) = z^{[p]} = W^{[p]}x^{[p-1]} + b^{[p]} = W^{[p]}\left(W^{[p-1]}x^{[p-2]} + b^{[p-1]}\right) + b^{[p]} =$$

$$= W^{[p]}W^{[p-1]}x^{[p-2]} + b^{[p]} + W^{[p]}b^{[p-1]} = \left(\prod_{i=0}^{p-1} W^{[p-i]}\right)x^{[0]} + b^{[p]} + \sum_{i=1}^{p-1}\left[\left(\prod_{j=0}^{i-1} W^{[p-i]}\right)b^{[p-i]}\right] =$$

$$= Wx^{[0]} + B,\ W = \left(\prod_{i=0}^{p-1} W^{[p-i]}\right),\ B = b^{[p]} + \sum_{i=1}^{p-1}\left[\left(\prod_{j=0}^{i-1} W^{[p-i]}\right)b^{[p-i]}\right]$$

Additionally, since this specific MLP has a single output, $Wx^{[0]}$ is simply the dot product between a weight vector and the input vector, with B being a scalar value, which can be written in this way:

$$\hat{z} = Wx^{[0]} + B = w_0 + \sum_{i=1}^{I} w_i x_i$$

Where w_0 is the scalar denoted by B, w_i the i-th value from W and x_i the i-th element from $x^{[0]}$. Therefore, the output of the MLP has the exact same format as the output of the linear regression. Since both methods also minimize the same loss function, it's expected they should converge on essentially the same weights and biases.

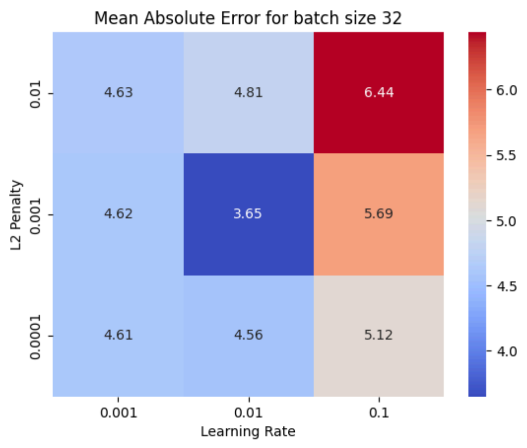
This is exactly what the boxplot shows. Both the Linear Regression model and the MLP without activation functions have basically the same test MAE, which indicates they have approximated the data in very much the same way. Furthermore, this only highlights the importance of activation functions. When compared to the other two models, the MLP with ReLU activation functions clearly outperforms them on the test data. This is because, while a MLP without activation functions is limited to linearly approximating the data, a MLP with activation functions can learn a nonlinear function, which enables the modeling of more complex data.

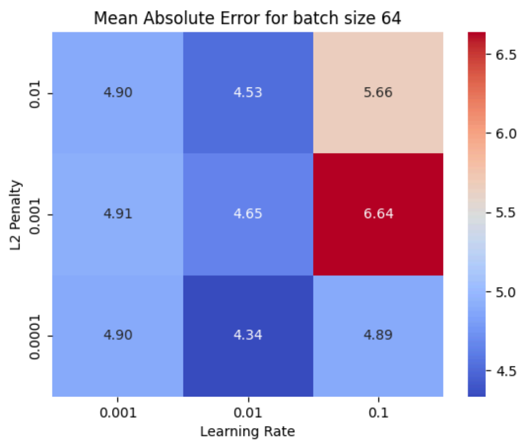
7) As illustrated in the heatmaps below, the best combination of values seems to be a L2 penalty of 0.001, a learning rate of 0.01 and a batch size of 32, with a MAE of 3.65.

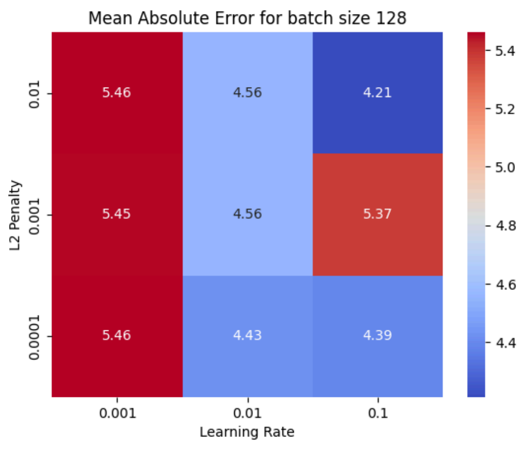
Overall, the L2 penalty does not appear to have much effect in this specific case, however, in theory, it should allow for greater generalization capabilities since it forces the parameters of the model to be smaller, which in turn makes it less sensible, in a way, to the possible oddities of a particular training dataset. It should also be noted that while the L2 penalty can help with overfitting, increasing it too much may cause underfitting.

Nevertheless, the batch size and learning rate seem to play a greater role. The performance seems to decrease for combinations of low batch size and high learning rate due to the gradient descent updates being very erratic: the high learning rate allows for big changes per update, even though the model only considers a small portion of the dataset. Furthermore, the performance also worsens for large batch sizes and small learning rates because the model takes too long to converge and hits the max iteration limit.

Therefore, there seems to be a certain sweet spot, as the heatmaps show. On the one hand, if the batch size is small, the learning rate should not be very high since each update does not take into account many observations. On the other hand, if the batch size is large, it is better to use higher learning rates so the model converges faster.







END