Aprendizagem 2023
Homework I — Group 003
(ist1107028, ist1107137)

Part I: Pen and paper

For questions in this group, show your numerical results with 5 decimals or scientific notation. Hint: we highly recommend the use of `numpy` (e.g., `linalg.pinv` for inverse) or other programmatic facilities to support the calculus involved in both questions (1) and (2).

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

   The first step will be to calculate the transformed feature space for each observation using the polynomial basis function:

   $$\phi(y_1, y_2) = y_1 \times y_2 \tag{1}$$

   By applying (1) to each observation, we get the following transformed feature space:

   $$\phi(x_1) = 1 \times 1 = 1 \qquad \phi(x_2) = 1 \times 3 = 3 \qquad \phi(x_3) = 3 \times 2 = 6$$
   $$\phi(x_4) = 3 \times 3 = 9 \qquad \phi(x_5) = 2 \times 4 = 8$$

   We end up with the following transformed feature space: $\phi(y_1, y_2) = [1, 3, 6, 9, 8]$

   The regression model in the transformed feature space is given by:

   $$z = w_0 + w_1 \cdot \phi(y_1, y_2)$$

   The next step is to learn a regression model using the OLS closed form solution which, for this exercise, is given by:

   $$W = (\Phi^T \Phi)^{-1} \Phi^T y_{\text{num}}$$

where Φ is the design matrix and y_{num} is the continuous output variable.

$$\Phi = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 9 \\ 1 & 8 \end{bmatrix} \qquad y_{\text{num}} = \begin{bmatrix} 1.25 \\ 7.0 \\ 2.7 \\ 3.2 \\ 5.5 \end{bmatrix}$$

As suggested in the hint, we will use `numpy` to calculate the weights W:

```python
import numpy as np

M = np.array([[1, 1], [1, 3], [1, 6], [1, 9], [1, 8]])
y_num = np.array([1.25, 7.0, 2.7, 3.2, 5.5])

W = np.linalg.inv(M.T @ M) @ M.T @ y_num
print(W)
```

The output of the code above is $[3.3159292, 0.11371681]$.
Therefore, the weights are $w_0 = 3.3159292$ and $w_1 = 0.11371681$.

The regression model is then given by:

$$y_{\text{num}} = 3.3159292 + 0.11371681 \cdot \phi(y_1, y_2)$$

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.**

The Ridge regression model is given by:

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

where Φ is the design matrix, y_{num} is the continuous output variable (which remain the same as the previous exercise), and λ is the penalty factor (this being the only difference compared to the first exercise).

```python
import numpy as np

M = np.array([[1, 1], [1, 3], [1, 6], [1, 9], [1, 8]])
lambda_value = 1.0
I = np.array([[1, 0], [0, 1]])  # Identity matrix
y_num = np.array([1.25, 7.0, 2.7, 3.2, 5.5])

W = np.linalg.inv(M.T @ M + lambda_value * I) @ M.T @ y_num
print(W)
```

The output of the code above is $[1.81808511, 0.32375887]$.
Therefore, the weights are $w_0 = 1.81808511$ and $w_1 = 0.32375887$.

The Ridge regression model is then given by:

$$y_{\text{num}} = 1.81808511 + 0.32375887 \cdot \phi(y_1, y_2)$$

Comparing the weights obtained in the OLS and Ridge regression models, we can see that in the Ridge regression model w_0 is smaller and w_1 is larger than the corresponding OLS model weights.

The regularization term in the Ridge regression model penalizes large weights, but it allows for the weights to increase or decrease relative to one another based on the data. In this case, the regularization reduces the bias term w_0, as the model learns to rely more on the feature w_1 to explain the variance in the target variable. This reduces the need for a large bias term, helping to prevent overfitting and stabilizing the model's predictions.

These results suggest that the feature has a strong positive correlation with the target variable. As a result, w_1 increases to better capture this relationship, allowing w_0 to decrease.

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

4. **Consider an MLP to predict the output $\mathbf{y_{class}}$ 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

$$\text{softmax}(z_c^{[\text{out}]}) = \frac{e^{z_c^{[\text{out}]}}}{\sum_{l=1}^{|C|} e^{z_c^{[\text{out}]}}}$$

, no activations on the hidden layer(s) and the cross-entropy loss:

$$\text{CE} = -\sum_{i=1}^{N} \sum_{l=1}^{|C|} t_l^{(\text{i})} \log(X_l^{[\text{out}](\text{i})}) \tag{2}$$

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.

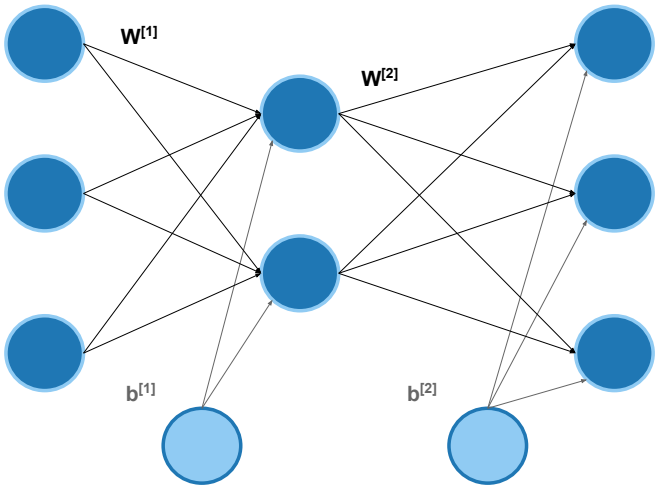

Figure 1: MLP architecture

We may also draw the corresponding scheme:

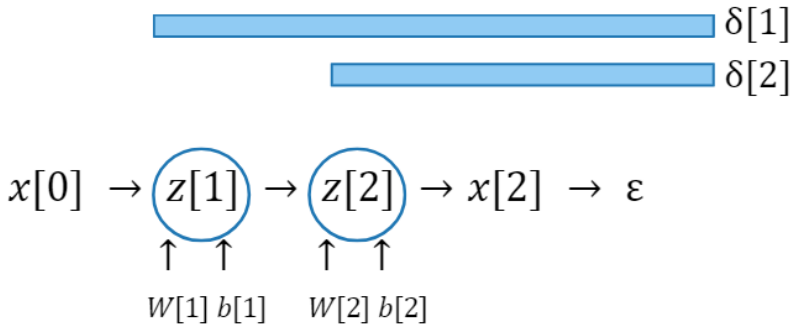

Figure 2: MLP Scheme

To begin, we initiate the process with the **forward propagation**. Below are the key equations to calculate the values of each layer, $x_i^{[p]}$:

$$z_i^{[p]} = W^{[p]}\, x_i^{[p-1]} + b^{[p]} \qquad\qquad x_i^{[p]} = f\left(z_i^{[p]}\right)$$

f is the activation function, for every layer, provided in the statement.

$$f = \text{softmax}(z_c^{[\text{out}]}) = \frac{e^{z_c^{[\text{out}]}}}{\sum_{l=1}^{|C|} e^{z_c^{[\text{out}]}}}$$

In this notation, the value p is the index of the MLP layer.

We may now calculate the values of each node in the network for the training observation:

$\boxed{x_1}$

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

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

$$x^{[2]} = f\left(z^{[2]}\right) = \begin{bmatrix} 0.461 \\ 0.309 \\ 0.229 \end{bmatrix}$$

Now, we can calculate the **backward propagation** to update the weights and biases. Firstly, we calculate the derivatives:

- Derivative of the cross-entropy loss function:

$$\frac{\partial \mathrm{CE}}{\partial Z_l^{[\mathrm{out}]}} = -\sum_{i=1}^{N} \frac{t_l^{(\mathrm{i})}}{X_l^{[\mathrm{out}](\mathrm{i})}} = x^{[2]} - t$$

- Derivative of the softmax function:

$$\frac{\partial f}{\partial z_c^{[\mathrm{out}]}} = f(z_c^{[\mathrm{out}]}) \cdot (1 - f(z_c^{[\mathrm{out}]}))$$

- Derivatives for the z[i]:

$$\frac{\partial x^{[i]}(z^{[i]})}{\partial z^{[i]}} = f(z^{[i]})(1 - f(z^{[i]})) \qquad \frac{\partial z^{[i]}(W^{[i]}, b^{[i]}, x^{[i-1]})}{\partial W^{[i]}} = x^{[i-1]}$$

$$\frac{\partial z^{[i]}(W^{[i]}, b^{[i]}, x^{[i-1]})}{\partial b^{[i]}} = 1 \qquad \frac{\partial z^{[i]}(W^{[i]}, b^{[i]}, x^{[i-1]})}{\partial x^{[i-1]}} = W^{[i]}$$

To start the recursion for the backward propagation, we calculate the delta in the output layer:

$$\delta^{[2]} = \frac{\partial E}{\partial \mathbf{x}^{[2]}} \circ \frac{\partial \mathbf{x}^{[2]}}{\partial \mathbf{z}^{[2]}} = (\mathbf{x}^{[2]} - t) \circ f(\mathbf{z}^{[2]})(1 - f(\mathbf{z}^{[2]}))$$

$$= \left(\begin{pmatrix} 0.461 \\ 0.309 \\ 0.229 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right) \circ f \begin{pmatrix} 2.7 \\ 2.3 \\ 2.0 \end{pmatrix} \cdot \left(1 - f \begin{pmatrix} 2.7 \\ 2.3 \\ 2.0 \end{pmatrix} \right) = \begin{pmatrix} 0.106 \\ 0.071 \\ 0.052 \end{pmatrix}$$

We can now calculate the delta for the hidden layer:

$$\delta^{[1]} = \left(\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{x}^{[1]}}\right)^T \cdot \delta^{[2]} \circ \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{x}^{[1]}} = \left(\mathbf{W}^{[2]}\right)^T \cdot \delta^{[2]} \circ f(\mathbf{z}^{[1]})(1 - f(\mathbf{z}^{[1]}))$$

$$= \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \\ 2 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0.106 \\ 0.071 \\ 0.052 \end{pmatrix} \circ f \begin{pmatrix} 0.3 \\ 0.3 \\ 0.4 \end{pmatrix} \circ \left(1 - f \begin{pmatrix} 0.3 \\ 0.3 \\ 0.4 \end{pmatrix}\right) = \begin{pmatrix} 0.074 \\ 0.131 \\ 0.108 \end{pmatrix}$$

Finally, we can calculate the gradients for the weights and biases:

$$\frac{\partial E}{\partial \mathbf{W}^{[1]}} = \delta^{[1]} \cdot \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}} = \delta^{[1]} \cdot \left(\mathbf{x}^{[0]}\right)^T$$
$$= \begin{pmatrix} 0.074 \\ 0.131 \\ 0.108 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \end{pmatrix} = \begin{pmatrix} 0.074 & 0.074 \\ 0.131 & 0.131 \\ 0.108 & 0.108 \end{pmatrix}$$

$$\mathbf{W}^{[1]} = \mathbf{W}^{[1]} - \eta \cdot \frac{\partial E}{\partial \mathbf{W}^{[1]}} = \begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.2 \\ 0.2 & 0.1 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 0.074 & 0.074 \\ 0.131 & 0.131 \\ 0.108 & 0.108 \end{pmatrix} = \begin{pmatrix} 0.0926 & 0.0926 \\ 0.0869 & 0.1869 \\ 0.1892 & 0.0892 \end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[1]}} = \delta^{[1]} \cdot \frac{\partial \mathbf{z}^{[1]^T}}{\partial \mathbf{b}^{[1]}} =$$

$$\mathbf{b}^{[1]} = \mathbf{b}^{[1]} - \eta \cdot \frac{\partial E}{\partial \mathbf{b}^{[1]}} = \begin{pmatrix} 0.1 \\ 0 \\ 0.1 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} \\ \\ \end{pmatrix} = \begin{pmatrix} 0.1 \\ 0 \\ 0.1 \end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{W}^{[2]}} = \delta^{[2]} \cdot \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{W}^{[2]}} = \delta^{[2]} \cdot \left(\mathbf{z}^{[1]}\right)^T$$
$$= \begin{pmatrix} 0.106 \\ 0.071 \\ 0.052 \end{pmatrix} \cdot \begin{pmatrix} 0.3 & 0.3 & 0.4 \end{pmatrix} = \begin{pmatrix} 0.032 & 0.032 & 0.042 \\ 0.021 & 0.021 & 0.028 \\ 0.016 & 0.016 & 0.021 \end{pmatrix}$$

$$\mathbf{W}^{[2]} = \mathbf{W}^{[2]} - \eta \cdot \frac{\partial E}{\partial \mathbf{W}^{[2]}} = \begin{pmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 0.032 & 0.032 & 0.042 \\ 0.021 & 0.021 & 0.028 \\ 0.016 & 0.016 & 0.021 \end{pmatrix}$$
$$= \begin{pmatrix} 0.997 & 1.997 & 1.997 \\ 0.998 & 1.998 & 0.998 \\ 0.998 & 0.998 & 0.998 \end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[2]}} = \delta^{[2]} \cdot \frac{\partial \mathbf{z}^{[2]^T}}{\partial \mathbf{b}^{[2]}} =$$

$$\mathbf{b}^{[2]} = \mathbf{b}^{[2]} - \eta \cdot \frac{\partial E}{\partial \mathbf{b}^{[2]}} = \begin{pmatrix} 0.1 \\ 0 \\ 0.1 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.1 \\ 0 \\ 0.1 \end{pmatrix}$$

Part II: Programming

Consider the parkinsons.csv dataset (available at the course's webpage), where the goal is to predict a patient's score on the Unified Parkinson's Disease Rating Scale based on various biomedical measurements. To answer question (5), average the performance of the models over 10 separate runs. In each run, use a different $80-20$ train test split by setting a `random_state = i`, with $i = 1...10$.

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

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 the results from the boxplots.**

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.