Aprendizagem 2022/23

Homework III – Group 018

I. Pen-and-paper [12v]

Consider the problem of learning a regression model from 5 univariate observations ((0.8), (1), (1.2), (1.4), (1.6)) with targets (24,20,10,13,12).

1) [5v] Consider the basis function, $\phi_j(x) = x^j$, for performing a 3-order polynomial regression,

$$\hat{z}(x, w) = \sum_{j=0}^{3} w_j \phi_j(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3.$$

Learn the Ridge regression (l_2 regularization) on the transformed data space using the closed form solution with $\lambda = 2$.

Hint: use *numpy* matrix operations (e.g., *linalg.pinv* for inverse) to validate your calculus.

	$\boldsymbol{\phi_1}$	$\boldsymbol{\phi_2}$	$\boldsymbol{\phi_3}$	$\boldsymbol{z}$
$\boldsymbol{x_1}$	0.8	0.64	0.512	24
$\boldsymbol{x_2}$	1	1	1	20
$\boldsymbol{x_3}$	1.2	1.44	1.728	10
$\boldsymbol{x_4}$	1.4	1.96	2.744	13
$\boldsymbol{x_5}$	1.6	2.56	4.096	12

Para Ridge regression, temos que:

- $w = (x^T x + \lambda I)^{-1}\, x^T z, \quad \lambda = 2.$

Para calcularmos a matriz de pesos w, precisamos de:

$$x = \begin{bmatrix} 1 & 0.8 & 0.64 & 0.512 \\ 1 & 1 & 1 & 1 \\ 1 & 1.2 & 1.44 & 1.728 \\ 1 & 1.4 & 1.96 & 2.744 \\ 1 & 1.6 & 2.56 & 4.096 \end{bmatrix}, \qquad x^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0.8 & 1 & 1.2 & 1.4 & 1.6 \\ 0.64 & 1 & 1.44 & 1.96 & 2.56 \\ 0.512 & 1 & 1.728 & 2.744 & 4.096 \end{bmatrix};$$

$$(x^T x) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0.8 & 1 & 1.2 & 1.4 & 1.6 \\ 0.64 & 1 & 1.44 & 1.96 & 2.56 \\ 0.512 & 1 & 1.728 & 2.744 & 4.096 \end{bmatrix} \begin{bmatrix} 1 & 0.8 & 0.64 & 0.512 \\ 1 & 1 & 1 & 1 \\ 1 & 1.2 & 1.44 & 1.728 \\ 1 & 1.4 & 1.96 & 2.744 \\ 1 & 1.6 & 2.56 & 4.096 \end{bmatrix} = \begin{bmatrix} 5 & 6 & 7.6 & 10.08 \\ 6 & 7.6 & 10.08 & 13.878 \\ 7.6 & 10.08 & 13.878 & 19.68 \\ 10.08 & 13.878 & 19.68 & 28.555 \end{bmatrix}$$

$$(x^T x + 2I)^{-1} = \begin{bmatrix} 7 & 6 & 7.6 & 10.08 \\ 6 & 9.6 & 10.08 & 13.878 \\ 7.6 & 10.08 & 15.878 & 19.68 \\ 10.08 & 13.878 & 19.68 & 30.555 \end{bmatrix}^{-1} = \begin{bmatrix} 0.342 & -0.121 & -0.075 & 10.08 \\ -0.121 & 0.389 & -0.097 & -0.074 \\ -0.075 & -0.097 & 0.373 & -0.171 \\ -0.009 & -0.074 & -0.171 & 0.180 \end{bmatrix}$$

$$x^T z = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0.8 & 1 & 1.2 & 1.4 & 1.6 \\ 0.64 & 1 & 1.44 & 1.96 & 2.56 \\ 0.512 & 1 & 1.728 & 2.744 & 4.096 \end{bmatrix} \begin{bmatrix} 24 \\ 20 \\ 10 \\ 13 \\ 12 \end{bmatrix} = \begin{bmatrix} 79 \\ 88.6 \\ 105.96 \\ 134.392 \end{bmatrix};$$

Desta forma, podemos calcular:

$$w = (x^T x + 2I)^{-1}\, x^T z = \begin{bmatrix} 0.342 & -0.121 & -0.075 & 10.08 \\ -0.121 & 0.389 & -0.097 & -0.074 \\ -0.075 & -0.097 & 0.373 & -0.171 \\ -0.009 & -0.074 & -0.171 & 0.180 \end{bmatrix} \begin{bmatrix} 79 \\ 88.6 \\ 105.96 \\ 134.392 \end{bmatrix} = \begin{bmatrix} 7.045 \\ 4.641 \\ 1.967 \\ -1.301 \end{bmatrix} \square$$

- $w_0 = 7.045\,,\ \ w_1 = 4.641,\ \ w_2 = 1.967,\ \ w_3 = -1.301\,;$

Logo, temos que:

$$\hat{z}(x, w) = 7.045 + 4.641x + 1.967x^2 - 1.301x^3 \ \ \square$$

Podemos agora calcular as nossas estimativas para as observações:

- $\hat{z}(x_1, w) = 7.045 + 4.641 \times 0.8 + 1.967 \times 0.8^2 - 1.301 \times 0.8^3 = 11.351$
- $\hat{z}(x_2, w) = 7.045 + 4.641 \times 1 + 1.967 \times 1^2 - 1.301 \times 1^3 = 12.352$
- $\hat{z}(x_3, w) = 7.045 + 4.641 \times 1.2 + 1.967 \times 1.2^2 - 1.301 \times 1.2^3 = 13.199$
- $\hat{z}(x_4, w) = 7.045 + 4.641 \times 1.4 + 1.967 \times 1.4^2 - 1.301 \times 1.4^3 = 13.828$
- $\hat{z}(x_5, w) = 7.045 + 4.641 \times 1.6 + 1.967 \times 1.6^2 - 1.301 \times 1.6^3 = 14.177$

	y_1	z	$\hat{z}$
x_1	0.8	24	11.351
x_2	1	20	12.352
x_3	1.2	10	13.199
x_4	1.4	13	13.828
x_5	1.6	12	14.177

2) [1v] Compute the training RMSE for the learnt regression model.

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

Logo, para o modelo de regressão de Ridge da alínea anterior, podemos afirmar que:

$$RMSE = \sqrt{\frac{1}{5}[(24-11.351)^2 + (20-12.352)^2 + (10-13.199)^2 + (13-13.828)^2 + (12-14.177)^2]} \approx 6.843 \; ❒$$

3) [6v] Consider a multi-layer perceptron characterized by one hidden layer with 2 nodes. Using the activation function $f(x) = e^{0.1x}$ on all units, all weights initialized as 1 (including biases), and the half-squared error loss, perform one batch gradient descent update (with learning rate η = 0.1) for the first three observations (0.8), (1) and (1.2).

$$função\ ativação(x) = f(x) = \phi(x) = e^{0.1x} = \phi^{[1]} = \phi^{[2]}, \quad \eta = 0.1$$

I. Forward propagation: $z^{[i]} = W^{[i]}v^{[i-1]} + b^{[i]}\,, \quad v^{[i]} = \phi\left(z^{[i]}\right);$

$$W^{[1]} = \begin{bmatrix}1\\1\end{bmatrix}, \quad b^{[1]} = \begin{bmatrix}1\\1\end{bmatrix}; \qquad W^{[2]} = \begin{bmatrix}1 & 1\end{bmatrix}, \quad b^{[2]} = [1];$$

- Para ($v^{[0]} = x^{[0]} = [0.8]$):

$$\boldsymbol{z}^{[\mathbf{1}]} = W^{[1]}v^{[0]} + b^{[1]} = \begin{bmatrix}1\\1\end{bmatrix}[0.8] + \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}1.8\\1.8\end{bmatrix}; \qquad \boldsymbol{v}^{[\mathbf{1}]} = \phi\left(z^{[1]}\right) = \begin{bmatrix}1.1972\\1.1972\end{bmatrix}$$

$$\boldsymbol{z}^{[\mathbf{2}]} = W^{[2]}v^{[1]} + b^{[2]} = \begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1.1972\\1.1972\end{bmatrix} + [1] = [3.3944]; \quad \boldsymbol{v}^{[\mathbf{2}]} = \phi\left(z^{[2]}\right) = [1.4042]$$

- Para ($v^{[0]} = x^{[0]} = [1]$):

$$\boldsymbol{z}^{[\mathbf{1}]} = W^{[1]}v^{[0]} + b^{[1]} = \begin{bmatrix}1\\1\end{bmatrix}[1] + \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}2\\2\end{bmatrix}; \qquad \boldsymbol{v}^{[\mathbf{1}]} = \phi\left(z^{[1]}\right) = \begin{bmatrix}1.2214\\1.2214\end{bmatrix}$$

$$\boldsymbol{z}^{[\mathbf{2}]} = W^{[2]}v^{[1]} + b^{[2]} = \begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1.2214\\1.2214\end{bmatrix} + [1] = [3.4428]; \quad \boldsymbol{v}^{[\mathbf{2}]} = \phi\left(z^{[2]}\right) = [1.4110]$$

- Para ($v^{[0]} = x^{[0]} = [1.2]$):

$$\boldsymbol{z}^{[\mathbf{1}]} = W^{[1]}v^{[0]} + b^{[1]} = \begin{bmatrix}1\\1\end{bmatrix}[1.2] + \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}2.2\\2.2\end{bmatrix}; \qquad \boldsymbol{v}^{[\mathbf{1}]} = \phi\left(z^{[1]}\right) = \begin{bmatrix}1.2461\\1.2461\end{bmatrix}$$

$$\boldsymbol{z}^{[\mathbf{2}]} = W^{[2]}v^{[1]} + b^{[2]} = \begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1.2461\\1.2461\end{bmatrix} + [1] = [3.4922]; \quad \boldsymbol{v}^{[\mathbf{2}]} = \phi\left(z^{[2]}\right) = [1.4180]$$

Aprendizagem 2022/23

Homework III – Group 018

II. Back propagation:

i. Usando *half-squared error loss*: $E\left(v^{[i]}, z\right) = \frac{1}{2}\left(v^{[i]} - z\right)^2$

ii. Derivar funções:

$$\frac{\partial E}{\partial v^{[i]}} = \frac{1}{2}\left(2v^{[i]} - 2z\right) = v^{[i]} - z\,;$$

$$\frac{\partial v^{[i]}}{\partial z^{[i]}} = \frac{\partial \phi^{[i]}}{\partial z^{[i]}} = \phi'^{\left(z^{[i]}\right)} = f'^{\left(z^{[i]}\right)} = 0.1 \times e^{0.1 z^{[i]}};$$

$$\frac{\partial z^{[i]}}{\partial W^{[i]}} = v^{[i-1]}\,, \qquad \frac{\partial z^{[i]}}{\partial b^{[i]}} = 1\,, \qquad \frac{\partial z^{[i]}}{\partial v^{[i-1]}} = W^{[i]}\,.$$

iii. Calcular deltas:

Com $x_1 = [0.8],\ x_2 = [1]\ e\ x_3 = [1.2]$.

- Last layer: $\delta^{[i]} = \frac{\partial E}{\partial v^{[i]}} \circ \frac{\partial v^{[i]}}{\partial z^{[i]}} = \left(v^{[i]} - z\right) \circ \phi'^{\left(z^{[i]}\right)}$

$$\delta_{x_1}^{[2]} = \left(v^{[2]} - z\right) \circ \phi'^{\left(z^{[2]}\right)} = ([1.4042] - [24]) \circ \phi'([3.3944]) = [-22.5958] \circ [0.1404] = [-3.1725]$$

$$\delta_{x_2}^{[2]} = \left(v^{[2]} - z\right) \circ \phi'^{\left(z^{[2]}\right)} = ([1.4110] - [20]) \circ \phi'([3.4428]) = [-18.5890] \circ [0.1411] = [-2.6229]$$

$$\delta_{x_3}^{[2]} = \left(v^{[2]} - z\right) \circ \phi'^{\left(z^{[2]}\right)} = ([1.4180] - [10]) \circ \phi'([3.4922]) = [-8.5820] \circ [0.1418] = [-1.2169]$$

- Hidden layer: $\delta^{[i]} = \left(\frac{\partial z^{[i+1]}}{\partial v^{[i]}}\right)^T . \delta^{[i+1]} \circ \frac{\partial v^{[i]}}{\partial z^{[i]}} = W^{[i+1]^T} \cdot \delta^{[i+1]} \circ \phi'^{\left(z^{[i]}\right)}$

$$\delta_{x_1}^{[1]} = W^{[2]^T} \cdot \delta^{[2]} \circ \phi'^{\left(z^{[1]}\right)} = \left(\begin{bmatrix}1\\1\end{bmatrix} \cdot [-3.1725]\right) \circ \phi'\left(\begin{bmatrix}1.8\\1.8\end{bmatrix}\right) = \begin{bmatrix}-3.1725\\-3.1725\end{bmatrix} \circ \begin{bmatrix}0.1197\\0.1197\end{bmatrix} = \begin{bmatrix}-0.3797\\-0.3797\end{bmatrix}$$

$$\delta_{x_2}^{[1]} = W^{[2]^T} \cdot \delta^{[2]} \circ \phi'^{\left(z^{[1]}\right)} = \left(\begin{bmatrix}1\\1\end{bmatrix} \cdot [-2.6229]\right) \circ \phi'\left(\begin{bmatrix}2\\2\end{bmatrix}\right) = \begin{bmatrix}-2.6229\\-2.6229\end{bmatrix} \circ \begin{bmatrix}0.1221\\0.1221\end{bmatrix} = \begin{bmatrix}-0.3203\\-0.3203\end{bmatrix}$$

$$\delta_{x_3}^{[1]} = W^{[2]^T} \cdot \delta^{[2]} \circ \phi'^{\left(z^{[1]}\right)} = \left(\begin{bmatrix}1\\1\end{bmatrix} \cdot [-1.2169]\right) \circ \phi'\left(\begin{bmatrix}2.2\\2.2\end{bmatrix}\right) = \begin{bmatrix}-1.2169\\-1.2169\end{bmatrix} \circ \begin{bmatrix}0.1246\\0.1246\end{bmatrix} = \begin{bmatrix}-0.1516\\-0.1516\end{bmatrix}$$

iv. Atualizar pesos e bias:

$$\frac{\partial E}{\partial W^{[i]}} = \frac{\partial E}{\partial v^{[i]}} \circ \frac{\partial v^{[i]}}{\partial z^{[i]}} \left(\frac{\partial z^{[i]}}{\partial W^{[i]}}\right)^T = \delta^{[i]} \cdot v^{[i-1]^T}; \qquad \frac{\partial E}{\partial b^{[i]}} = \frac{\partial E}{\partial v^{[i]}} \circ \frac{\partial v^{[i]}}{\partial z^{[i]}} \left(\frac{\partial z^{[i]}}{\partial b^{[i]}}\right)^T = \delta^{[i]};$$

Tendo em conta os cálculos anteriores e sabendo que

$$W_{new}^{[i]} = W^{[i]} - \eta \sum_{i=1}^{3} \frac{\partial E}{\partial W^{[i]}} = W^{[i]} - \eta \sum_{i=1}^{3} \delta^{[i]} \cdot v^{[i-1]^T};$$

$$b_{new}^{[i]} = b^{[i]} - \eta \sum_{i=1}^{3} \frac{\partial E}{\partial b^{[i]}} = b^{[i]} - \eta \sum_{i=1}^{3} \delta^{[i]} :$$

- Para x_1:

$$\frac{\partial E}{\partial W^{[1]}} = \delta_{x_1}^{[1]} \cdot v^{[0]^T} = \begin{bmatrix}-0.3797\\-0.3797\end{bmatrix} \cdot [0.8] = \begin{bmatrix}-0.3038\\-0.3038\end{bmatrix}; \qquad \frac{\partial E}{\partial b^{[1]}} = \delta_{x_1}^{[1]} = \begin{bmatrix}-0.3797\\-0.3797\end{bmatrix};$$

$$\frac{\partial E}{\partial W^{[2]}} = \delta_{x_1}^{[2]} \cdot v^{[1]^T} = [-3.1725] \cdot [1.1972 \quad 1.1972] = [-3.7981 \quad -3.7981]; \qquad \frac{\partial E}{\partial b^{[2]}} = \delta_{x_1}^{[2]} = [-3.1725];$$

- Para x_2:

$$\frac{\partial E}{\partial W^{[1]}} = \delta_{x_2}^{[1]} \cdot v^{[0]^T} = \begin{bmatrix}-0.3203\\-0.3203\end{bmatrix} \cdot [1] = \begin{bmatrix}-0.3203\\-0.3203\end{bmatrix}; \qquad \frac{\partial E}{\partial b^{[1]}} = \delta_{x_2}^{[1]} = \begin{bmatrix}-0.3203\\-0.3203\end{bmatrix};$$

$$\frac{\partial E}{\partial W^{[2]}} = \delta_{x_2}^{[2]} \cdot v^{[1]^T} = [-2.6229] \cdot [1.2214 \quad 1.2214] = [-3.2036 \quad -3.2036]; \qquad \frac{\partial E}{\partial b^{[2]}} = \delta_{x_2}^{[2]} = [-2.6229];$$

- Para x_3:

$$\frac{\partial E}{\partial W^{[1]}} = \delta_{x_3}^{[1]} \cdot v^{[0]^T} = \begin{bmatrix}-0.1516\\-0.1516\end{bmatrix} \cdot [1.2] = \begin{bmatrix}-0.1819\\-0.1819\end{bmatrix}; \qquad \frac{\partial E}{\partial b^{[1]}} = \delta_{x_3}^{[1]} = \begin{bmatrix}-0.1516\\-0.1516\end{bmatrix};$$

$$\frac{\partial E}{\partial W^{[2]}} = \delta_{x_3}^{[2]} \cdot v^{[1]^T} = [-1.2169] \cdot [1.2461 \quad 1.2461] = [-1.5164 \quad -1.5164]; \qquad \frac{\partial E}{\partial b^{[2]}} = \delta_{x_3}^{[2]} = [-1.2169];$$

$$W^{[1]} = W^{[1]} - \eta \sum_{i=1}^{3} \frac{\partial E}{\partial W^{[1]}} = \begin{bmatrix}1\\1\end{bmatrix} - 0.1 \left(\begin{bmatrix}-0.3038\\-0.3038\end{bmatrix} + \begin{bmatrix}-0.3203\\-0.3203\end{bmatrix} + \begin{bmatrix}-0.1819\\-0.1819\end{bmatrix}\right) = \begin{bmatrix}1.0806\\1.0806\end{bmatrix}$$

$$W^{[2]} = W^{[2]} - \eta \sum_{i=1}^{3} \frac{\partial E}{\partial W^{[2]}} = [1 \quad 1] - 0.1\,([-3.7981 \quad -3.7981] + [-3.2036 \quad -3.2036] + [-1.5164 \quad -1.5164]) = [1.8518 \quad 1.8518]$$

$$b^{[1]} = b^{[1]} - \eta \sum_{i=1}^{3} \frac{\partial E}{\partial b^{[1]}} = \begin{bmatrix}1\\1\end{bmatrix} - 0.1 \left(\begin{bmatrix}-0.3797\\-0.3797\end{bmatrix} + \begin{bmatrix}-0.3203\\-0.3203\end{bmatrix} + \begin{bmatrix}-0.1516\\-0.1516\end{bmatrix}\right) = \begin{bmatrix}1.0852\\1.0852\end{bmatrix}$$

$$b^{[2]} = b^{[2]} - \eta \sum_{i=1}^{3} \frac{\partial E}{\partial b^{[2]}} = [1] - 0.1\,([-3.1725] + [-2.6229] + [-1.2169]) = [1.7012]$$

Aprendizagem 2022/23

Homework III – Group 018

II. Programming and critical analysis [8v]

Consider the following three regressors applied on *kin8nm.arff* data (available at the webpage):

- linear regression with Ridge regularization term of 0.1;
- two MLPs – $MLP1$ and $MLP2$ – each with two hidden layers of size 10, hyperbolic tangent function as the activation function of all nodes, a maximum of 500 iterations, and a fixed seed ($random\_state = 0$). $MLP1$ should be parameterized with early stopping while $MLP2$ should not consider early stopping. Remaining parameters (e.g., loss function, batch size, regularization term, solver) should be set as default.

Using a 70-30 training-test split with a fixed seed ($random\_state = 0$):

4) [4v] Compute the MAE of the three regressors: linear regression, $MLP1$ and $MLP2$.

$$MAE_{linear\ regression} = 0.162829976437694$$

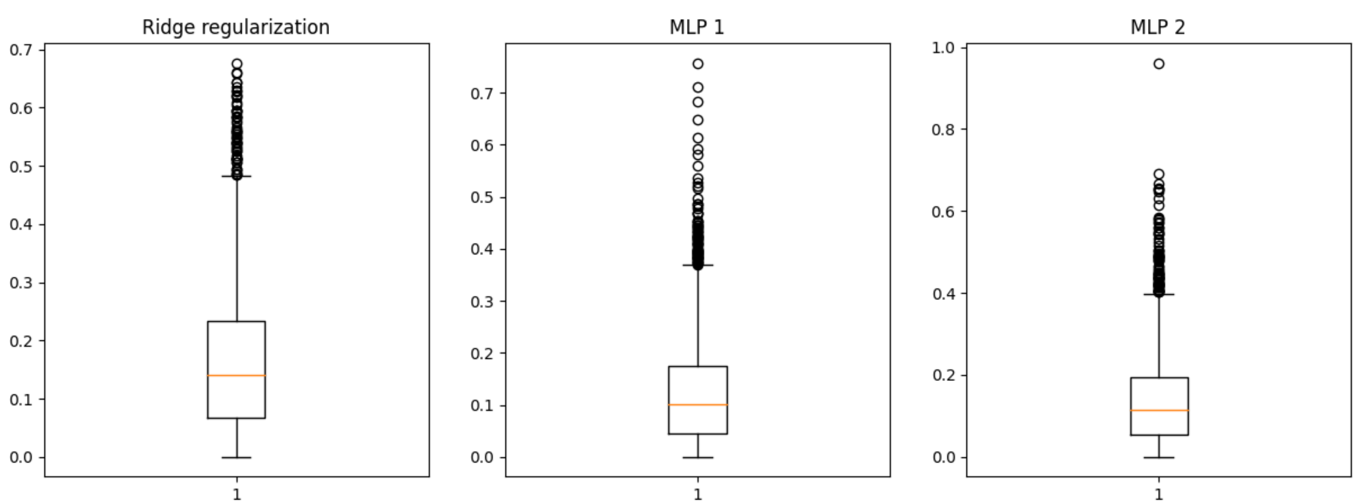
$$MAE_{MLP1} = 0.12495149587528392$$

$$MAE_{MLP2} = 0.13791486374334738$$

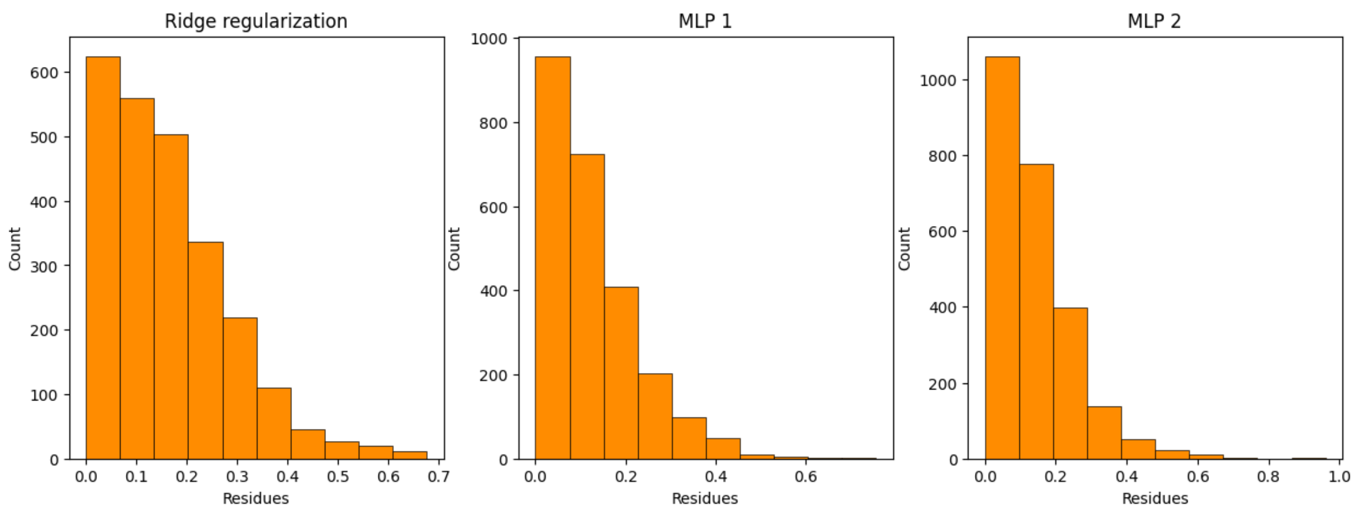
5) [1.5v] Plot the residues (in absolute value) using two visualizations: boxplots and histograms. *Hint*: consider using boxplot and hist functions from *matplotlib.pyplot* to this end

---

Boxplots:



Histograms:



6) [1v] How many iterations were required for $MLP1$ and $MLP2$ to converge?

Adicionando o parâmetro $verbose = True$ ao regressor, recebemos o output seguinte:

```
MLP1:
 - Iteration 248, loss = 0.01235355
   Validation score: 0.637879
   Validation score did not improve more than tol=0.000100 for 10 consecutive epochs.
Stopping.

MLP2:
 - Iteration 61, loss = 0.01494471
   Training loss did not improve more than tol=0.000100 for 10 consecutive epochs.
Stopping.
```

E, portanto, podemos concluir que:

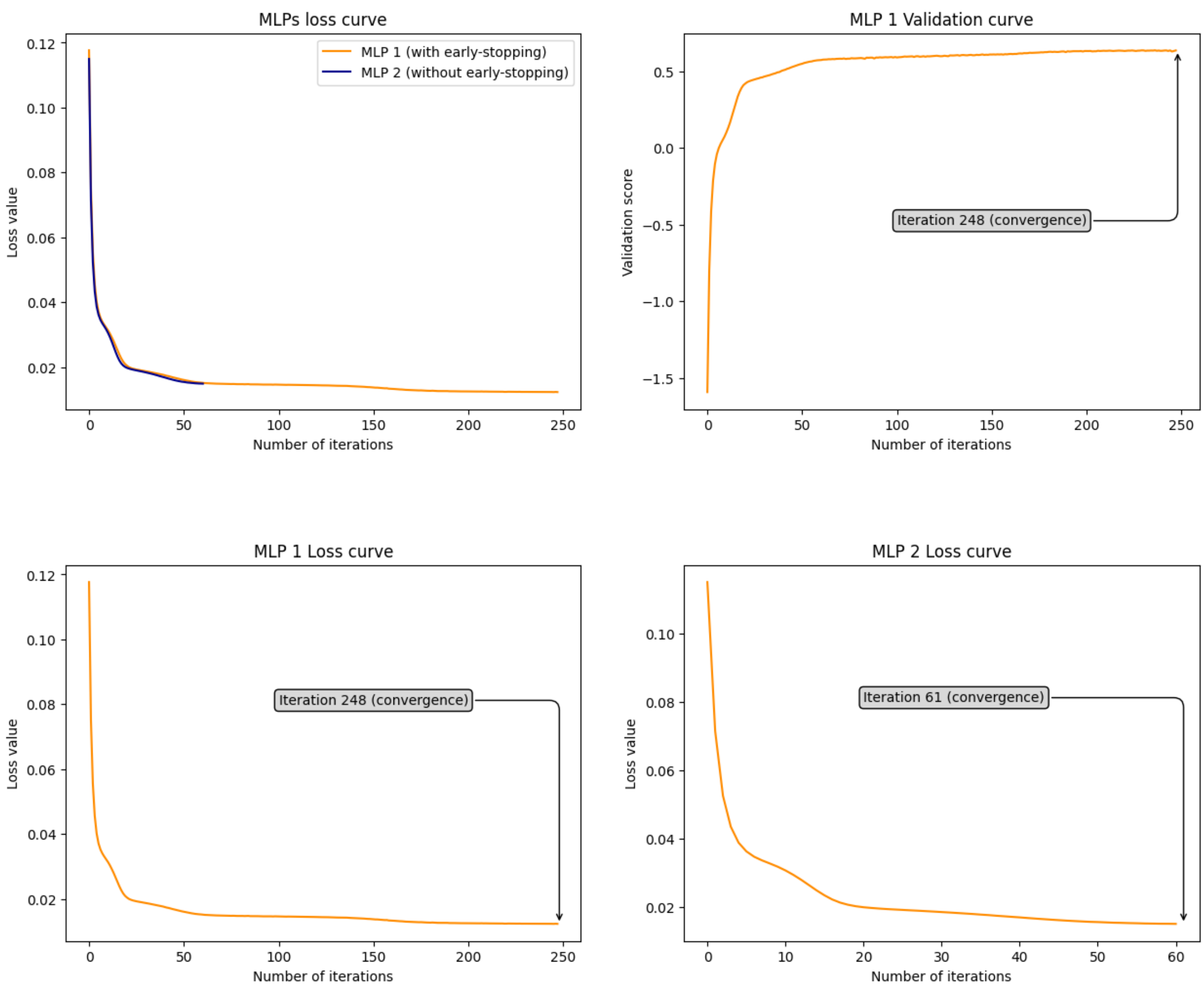
- O MLP1 demora 248 iterações a convergir;
- O MLP2 demora 61 iterações a convergir.

Aprendizagem 2022/23

Homework III – Group 018

7) [1.5v] What can be motivating the unexpected differences on the number of iterations? Hypothesize one reason underlying the observed performance differences between the MLPs.

MLPs loss and validation curves:



A razão para a enorme diferença no número de iterações até à convergência é, claramente, haver ou não early-stopping.

Quando utilizamos early-stopping, temos em consideração um conjunto de validação, que nos é útil de duas maneiras:

- Prevenir overfitting, ou seja, que o modelo se ajuste demasiado aos dados do conjunto de teste (evitando que o modelo tenha uma má performance quando consideramos outras observações que não as do conjunto de teste);
- Prevenir underfitting, ou seja, que o modelo termine a sua evolução cedo demais, levando-o a parar num mínimo local (devemos ter em conta que ambos os MLPs consideram "momentum" no processo de diminuição do erro, algo que também ajuda a prevenir a paragem em mínimos locais).

Todo este processo de ter em conta um conjunto de validação vai levar a uma evolução do modelo muito mais controlada, apesar da minimização do erro se tornar também bem mais prolongada, sendo por isso necessárias muitas mais iterações até à convergência do MLP1 (olhando para o gráfico do validation score do MLP1, podemos ter noção da sua evolução, paralela à do modelo).

A conclusão acima, é reforçada pela análise dos gráficos da diminuição do erro de ambos os MLPs (Loss curves). Reparamos que, para o MLP2 (sem early-stopping), existe uma diminuição do erro um pouco mais acentuada do que para o MLP1 (com early-stopping). No entanto, a convergência é atingida muito mais cedo no MLP2 (61 iterações) do que no MLP1 (248 iterações).

Analisando ainda os MAEs de ambos os MLPs, reparamos num contraste em termos de performance. A maior rapidez a convergir tem claramente as suas desvantagens: o menor controlo nos critérios de convergência do modelo levam a um erro médio maior, ≈ 0.138 para o MLP2, comparado com ≈ 0.125 para o MLP1 (uma diferença considerável).

Em suma, podemos afirmar que, por um lado, uma solução sem early-stopping é claramente mais rápida a convergir. Por outro, esta não terá uma performance tão boa quando comparada com uma que considere early-stopping, que, no entanto, torna mais lenta a convergência (para o caso estudado).

Encontramos, portanto, as seguintes implicações:

- Convergência mais rápida $\rightarrow$ maior erro médio (menor performance);
- Convergência mais demorada $\rightarrow$ menor erro médio (maior performance).

Aprendizagem 2022/23
Homework III – Group 018

Appendix

```
# Import wall
from scipy.io.arff import loadarff

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import Ridge
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

# -------------------------------------------------------- #

# Load and prepare data.
data = loadarff('kin8nm.arff')
df = pd.DataFrame(data[0])

X = df.drop("y", axis=1)
y = df['y']

# -------------------------------------------------------- #

# Creates the regression model we'll be using:
# - linear regression model with ridge regularization of 0.1.
reg_term = 0.1
ridge_regr = Ridge(alpha = reg_term)


# Creates the MLPs we'll be using:
# - two MLPs with 2 hidden layers (size 10), tanh as the activation function and max iterations of 500;
# - MLP1 has early stopping;
# - MLP2 doesn't have early stopping.
mlp_1 = MLPRegressor(hidden_layer_sizes = (2, 10), activation = 'tanh', max_iter = 500, early_stopping
= True, random_state = 0, verbose = True)
mlp_2 = MLPRegressor(hidden_layer_sizes = (2, 10), activation = 'tanh', max_iter = 500, early_stopping
= False, random_state = 0, verbose = True)


# Apply 70-30 training-testing split to our data.
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, random_state = 0)
```

```
# Fit the three regressors to our training data and apply regression to the test values.
ridge_regr.fit(X_train.values, y_train)
ridge_pred = ridge_regr.predict(X_test)

mlp_1.fit(X_train.values, y_train)
mlp_1_pred = mlp_1.predict(X_test)

mlp_2.fit(X_train.values, y_train)
mlp_2_pred = mlp_2.predict(X_test)


# Question 4:
# Compute the MAE of our three regressors.
MAE_ridge = mean_absolute_error(y_test, ridge_pred)
MAE_mlp_1 = mean_absolute_error(y_test, mlp_1_pred)
MAE_mlp_2 = mean_absolute_error(y_test, mlp_2_pred)

print("Linear regression / Ridge regularization (MAE): " + str(MAE_ridge) + "\n" \
      "MLP1 regression (MAE): " + str(MAE_mlp_1) + "\n" \
      "MLP2 regression (MAE): " + str(MAE_mlp_2) + "\n")


# Question 5:

# Get the residue arrays for our three regressors.
ridge_residues = np.absolute(np.subtract(y_test, ridge_pred))
mlp_1_residues = np.absolute(np.subtract(y_test, mlp_1_pred))
mlp_2_residues = np.absolute(np.subtract(y_test, mlp_2_pred))

data = {"Ridge regularization": ridge_residues, \
        "MLP 1": mlp_1_residues, \
        "MLP 2": mlp_2_residues}


# Boxplots.
fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize = (15, 5))

ax1.boxplot(ridge_residues)
ax1.set_title("Ridge regularization")

ax2.boxplot(mlp_1_residues)
ax2.set_title("MLP 1")

ax3.boxplot(mlp_2_residues)
ax3.set_title("MLP 2")
```

```
# Histograms.
fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize = (15, 5))

ax1.hist(ridge_residues, color = "darkorange", lw = 0.5, edgecolor = "black")
ax1.set_xlabel("Residues")
ax1.set_ylabel("Count")
ax1.set_title("Ridge regularization")

ax2.hist(mlp_1_residues, color = "darkorange", lw = 0.5, edgecolor = "black")
ax2.set_xlabel("Residues")
ax2.set_ylabel("Count")
ax2.set_title("MLP 1")

ax3.hist(mlp_2_residues, color = "darkorange", lw = 0.5, edgecolor = "black")
ax3.set_xlabel("Residues")
ax3.set_ylabel("Count")
ax3.set_title("MLP 2")




# Question 6:
# By adding the 'verbose' parameter to our MLP regressors:
# - MLPRegressor(..., verbose = 500)

# We get the following output:
'''
MLP1:
 - Iteration 248, loss = 0.01235355
   Validation score: 0.637879
   Validation score did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.

MLP2:
 - Iteration 61, loss = 0.01494471
   Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
'''

# And so:
# - takes 248 iterations for MLP1 to converge;
# - takes 61 iterations for MLP2 to converge.
```

```
# Loss function plots.

# ------------------------------------------------------ #
bbox = dict(boxstyle ="round", fc ="0.85")
arrowprops = dict(arrowstyle = "->", connectionstyle = "angle, angleA = 0, angleB = 90, rad = 10")
# ------------------------------------------------------ #


plt.figure(figsize=(15, 5))

plt.subplot(121)
plt.plot(mlp_1.loss_curve_, label = "MLP 1 (with early-stopping)", color = "darkorange")
plt.plot(mlp_2.loss_curve_, label = "MLP 2 (without early-stopping)", color = "darkblue")
plt.legend(loc = "upper right")
plt.xlabel("Number of iterations")
plt.ylabel("Loss value")
plt.title("MLPs loss curve")

plt.subplot(122)
plt.plot(mlp_1.validation_scores_, color = "darkorange")
plt.annotate("Iteration 248 (convergence)", xy = (248, 0.637879), bbox = bbox, arrowprops = arrowprops,
xytext =(100, -0.5))
plt.xlabel("Number of iterations")
plt.ylabel("Validation score")
plt.title("MLP 1 Validation curve")


plt.figure(figsize=(15, 5))

plt.subplot(121)
plt.plot(mlp_1.loss_curve_, color = "darkorange")
plt.annotate("Iteration 248 (convergence)", xy = (248, 0.01235355), bbox = bbox, arrowprops =
arrowprops, xytext =(100, 0.08))
plt.xlabel("Number of iterations")
plt.ylabel("Loss value")
plt.title("MLP 1 Loss curve")

plt.subplot(122)
plt.plot(mlp_2.loss_curve_, color = "darkorange")
plt.annotate("Iteration 61 (convergence)", xy = (61, 0.01494471), bbox = bbox, arrowprops = arrowprops,
xytext =(20, 0.08))
plt.xlabel("Number of iterations")
plt.ylabel("Loss value")
plt.title("MLP 2 Loss curve")
```