HW 3

13 de outubro de 2023 16:29

1. a) • Obter a design matrix (Φ)

radial basis function: $\phi_j(x) = \exp\left(-\frac{\|x-c_j\|^2}{2}\right)$

$$\Phi = \begin{bmatrix} 1 & \phi_1(x_1) & \phi_2(x_1) & \phi_3(x_1) \\ 1 & \phi_1(x_2) & \phi_2(x_2) & \phi_3(x_2) \\ 1 & \phi_1(x_3) & \phi_2(x_3) & \phi_3(x_3) \\ 1 & \phi_1(x_4) & \phi_2(x_4) & \phi_3(x_4) \end{bmatrix}$$

(1: coluna dos bias)

$$\phi_1(x_1) = \exp\left(-\frac{\|(0.7,-0.3)-(0,0)\|^2}{2}\right) = \exp\left(-\frac{\left(\sqrt{(0.7-0)^2+(0.3-0)^2}\right)^2}{2}\right) = 0.7483$$

$$\phi_2(x_1) = \exp\left(-\frac{\|(0.7,-0.3)-(1,-1)\|^2}{2}\right) = \exp\left(-\frac{\left(\sqrt{(0.7-1)^2+(-0.3+1)^2}\right)^2}{2}\right) = 0.7483$$

$$\phi_3(x_1) = \exp\left(-\frac{\|(0.7,-0.3)-(-1,1)\|^2}{2}\right) = \exp\left(-\frac{\left(\sqrt{(0.7+1)^2+(-0.3-1)^2}\right)^2}{2}\right) = 0.1013$$

Calculando para as outras entradas da matriz com recurso ao Python, obtemos a seguinte matriz:

$$\Phi = \begin{bmatrix} 1 & 0.7483 & 0.7483 & 0.1013 \\ 1 & 0.8146 & 0.2712 & 0.3312 \\ 1 & 0.7118 & 0.0963 & 0.7118 \\ 1 & 0.8825 & 0.1612 & 0.6538 \end{bmatrix} \quad \Phi^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.7483 & 0.8146 & 0.7118 & 0.8825 \\ 0.7483 & 0.2712 & 0.0963 & 0.1612 \\ 0.1013 & 0.3312 & 0.7118 & 0.6538 \end{bmatrix}$$

• Regressão de Ridge

Para a regressão de Ridge, tem-se que $E(w) = \frac{1}{2}\sum_{i=1}^{n}(z_i - w^T x_i)^2 + \frac{\lambda}{2}\|w\|_2^2$

Quer-se $\nabla E(w) = 0$. Resolvendo isto, encontra-se a solução fechada para a regressão de Ridge:

$w = (X^T.X + \lambda.I)^{-1}.X^T.z$, No nosso caso, $X = \Phi$, logo temos $w = (\Phi^T.\Phi + \lambda.I)^{-1}.\Phi^T.z$

$$(\Phi^T\Phi + \lambda I)^{-1} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.6833 & 0.7260 & 0.6621 & 0.7188 \\ 0.6633 & 0.4458 & 0.3390 & 0.3847 \\ 0.3430 & 0.4155 & 0.6621 & 0.6307 \end{bmatrix} \begin{bmatrix} 1 & 0.6833 & 0.6833 & 0.3430 \\ 1 & 0.7260 & 0.4458 & 0.4155 \\ 1 & 0.6621 & 0.3390 & 0.6621 \\ 1 & 0.7188 & 0.3847 & 0.6307 \end{bmatrix} + \lambda \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)^{-1}$$

(Φ^T; Φ; $I(4\times4)$)

$$\Phi^T z = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.6833 & 0.7260 & 0.6621 & 0.7188 \\ 0.6633 & 0.4458 & 0.3390 & 0.3847 \\ 0.3430 & 0.4155 & 0.6621 & 0.6307 \end{bmatrix} \begin{bmatrix} 0.8 \\ 0.6 \\ 0.3 \\ 0.3 \end{bmatrix}$$

(z (targets))

$$w = (\Phi^T.\Phi + \lambda I)^{-1}.\Phi^T.z = \begin{bmatrix} 0.3391 \\ 0.1995 \\ 0.4010 \\ -0.2960 \end{bmatrix} \begin{matrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{matrix}$$

Utilizando Python e funções como linalg.pinv e numpy.dot

Resultado da regressão: $y(x) = \underbrace{0.3391}_{w_0} + \underbrace{0.1995}_{w_1} e^{-\frac{\|x-\overbrace{(0,0)}^{c_0}\|^2}{2}} + \underbrace{0.4010}_{w_2} e^{-\frac{\|x-\overbrace{(1,-1)}^{c_1}\|^2}{2}} - \underbrace{0.2960}_{w_3} e^{-\frac{\|x-\overbrace{(-1,1)}^{c_2}\|^2}{2}}$

b) $RMSE(z,\hat{z}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_i-\hat{z}_i)^2}$ $\quad z = (0.8, 0.6, 0.3, 0.3)$

$y(x_1) = 0.3391 + 0.1995\times 0.7483 + 0.4010\times 0.7483 - 0.2960\times 0.1013 = 0.7585 = \hat{z}_1$

$y(x_2) = 0.3391 + 0.1995\times 0.8146 + 0.4010\times 0.2712 - 0.2960\times 0.3312 = 0.5123 = \hat{z}_2$

$y(x_3) = 0.3391 + 0.1995\times 0.7118 + 0.4010\times 0.0963 - 0.2960\times 0.7118 = 0.3090 = \hat{z}_3$

$y(x_2) = 0.3391 + 0.1995 \times 0.8146 + 0.4010 \times 0.2712 - 0.2960 \times 0.3312 = 0.5123 = \hat{z}_2$

$y(x_3) = 0.3391 + 0.1995 \times 0.7118 + 0.4010 \times 0.0963 - 0.2960 \times 0.7118 = 0.3090 = \hat{z}_3$

$y(x_4) = 0.3391 + 0.1995 \times 0.8825 + 0.4010 \times 0.1612 - 0.2960 \times 0.6538 = 0.3863 = \hat{z}_4$

$$\text{RMSE}(z,\hat{z}) = \sqrt{\frac{1}{4}\left((0.8-0.7584)^2 + (0.6-0.5123)^2 + (0.3-0.3090)^2 + (0.3-0.3863)^2\right)} \approx \boxed{0.065}$$

2. • Forward propagation

Propagar para $x_1 = \begin{pmatrix}1\\1\\1\\1\end{pmatrix}$:

→ função de ativação, $f(x) = \tanh(0.5x-2)$

$$z^{1(1)} = W^1 x_1 + b^1 = \begin{bmatrix}1&1&1&1\\1&1&2&1\\1&1&1&1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} + \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}4\\5\\4\end{bmatrix} + \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}5\\6\\5\end{bmatrix} \qquad x^{1(1)} = f(z^{1(1)}) = \begin{bmatrix}\tanh(0.5\times5-2)\\\tanh(0.5\times6-2)\\\tanh(0.5\times5-2)\end{bmatrix} = \begin{bmatrix}0.4621\\0.7616\\0.4621\end{bmatrix}$$

$$z^{2(1)} = W^2 x^{1(1)} + b^2 = \begin{bmatrix}1&4&1\\1&1&1\end{bmatrix}\begin{bmatrix}0.4621\\0.7616\\0.4621\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}4.9706\\2.6858\end{bmatrix} \qquad x^{2(1)} = f(z^{2(1)}) = \begin{bmatrix}\tanh(0.5\times4.9706-2)\\\tanh(0.5\times2.6858-2)\end{bmatrix} = \begin{bmatrix}0.4505\\-0.5764\end{bmatrix}$$

$$z^{3(1)} = W^3 x^{2(1)} + b^3 = \begin{bmatrix}1&1\\3&1\\1&1\end{bmatrix}\begin{bmatrix}0.4505\\-0.5764\end{bmatrix} + \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}0.8741\\1.7751\\0.8741\end{bmatrix} \qquad x^{3(1)} = f(z^{3(1)}) = \begin{bmatrix}\tanh(0.5\times0.8741-2)\\\tanh(0.5\times1.7751-2)\\\tanh(0.5\times0.8741-2)\end{bmatrix} = \begin{bmatrix}-0.9159\\-0.8049\\-0.9159\end{bmatrix}$$

Propagando agora para $x_2 = \begin{pmatrix}1\\0\\0\\-1\end{pmatrix}$:

$$z^{1(2)} = W^1 x_2 + b^1 = \begin{bmatrix}1&1&1&1\\1&1&2&1\\1&1&1&1\end{bmatrix}\begin{bmatrix}1\\0\\0\\-1\end{bmatrix} + \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} \qquad x^{1(2)} = f(z^{1(2)}) = \begin{bmatrix}\tanh(0.5\times1-2)\\\tanh(0.5\times1-2)\\\tanh(0.5\times1-2)\end{bmatrix} = \begin{bmatrix}-0.9051\\-0.9051\\-0.9051\end{bmatrix}$$

$$z^{2(2)} = W^2 x^{1(2)} + b^2 = \begin{bmatrix}1&4&1\\1&1&1\end{bmatrix}\begin{bmatrix}-0.9051\\-0.9051\\-0.9051\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}-4.4306\\-1.7153\end{bmatrix} \qquad x^{2(2)} = f(z^{2(2)}) = \begin{bmatrix}\tanh(0.5\times(-4.4306)-2)\\\tanh(0.5\times(-1.7153)-2)\end{bmatrix} = \begin{bmatrix}-0.9996\\-0.9934\end{bmatrix}$$

$$z^{3(2)} = W^3 x^{2(2)} + b^3 = \begin{bmatrix}1&1\\3&1\\1&1\end{bmatrix}\begin{bmatrix}-0.9996\\-0.9934\end{bmatrix} + \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}-0.9930\\-2.9922\\-0.9930\end{bmatrix} \qquad x^{3(2)} = f(z^{3(2)}) = \begin{bmatrix}\tanh(0.5\times(-0.9930)-2)\\\tanh(0.5\times(-2.9922)-2)\\\tanh(0.5\times(-0.9930)-2)\end{bmatrix} = \begin{bmatrix}-0.9865\\-0.9982\\-0.9865\end{bmatrix}$$

• Cálculos auxiliares (necessários para os próximos passos)

$$\frac{\partial f(x)}{\partial x} = \frac{\partial}{\partial x}(\tanh(0.5x-2)) = 0.5(1-\tanh^2(0.5x-2)) \qquad E(t,x^3) = \frac{1}{2}(x^3-t)^2$$

$$\frac{\partial E(t,x^3)}{\partial x^3} = x^3 - t \qquad \frac{\partial z^i(w^i,b^i,x^{i-1})}{\partial w^i} = x^{i-1} \qquad \frac{\partial z^i(w^i,b^i,x^{i-1})}{\partial x^{i-1}} = w^i \qquad \frac{\partial z^i(w^i,b^i,x^{i-1})}{\partial b^i} = 1$$

Cálculo dos deltas

⇒ Para a última camada ($\delta^{3(1)}$ e $\delta^{3(2)}$), utiliza-se a fórmula: $\delta^{[p]} = \frac{\partial E}{\partial x^{[p]}} \circ \frac{\partial x^{[p]}}{\partial z^{[p]}} = (x^{[p]} - t) \circ \phi'^{[p]}(z^{[p]})$

produto Hadamard ↓

$$\delta^{3(1)} = (x^{3(1)} - t) \circ f'^3(z^{3(1)}) = \left(\begin{bmatrix}-0.9159\\-0.8049\\-0.9159\end{bmatrix} - \begin{bmatrix}-1\\1\\-1\end{bmatrix}\right) \circ \left(\frac{1}{2}\left(1 - \tanh^2\left(\frac{1}{2}\begin{bmatrix}0.8741\\1.7751\\0.8741\end{bmatrix} - 2\right)\right)\right)$$

↳ Uma vez que ... [-1,1]

$$\delta^{3(1)} = (x^{3(1)} - b) \circ f'^{3}(z^{3(1)}) = \left(\begin{bmatrix} -0.8049 \\ -0.9159 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) \circ \left(\frac{1}{2}\left(1 - \tanh^2\left(\frac{1}{2}\begin{bmatrix} 1.7751 \\ 0.8741 \end{bmatrix} - 2\right)\right)\right)$$

↳ Uma vez que x_1 está associado a B, e o domínio de tanh é $[-1, 1]$, a entrada para B fica a 1 e os restantes a -1

$$= \begin{bmatrix} 0.0841 \\ -1.8049 \\ 0.0841 \end{bmatrix} \circ \begin{bmatrix} 0.0806 \\ 0.1760 \\ 0.0806 \end{bmatrix} = \begin{bmatrix} 0.0068 \\ -0.3177 \\ 0.0068 \end{bmatrix}$$

o mesmo que acima, mas x_2 está associado a A

$$\delta^{3(2)} = (x^{3(2)} - b) \circ f'^{3}(z^{3(2)}) = \left(\begin{bmatrix} -0.9865 \\ -0.9982 \\ -0.9865 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}\right) \circ \left(\frac{1}{2}\left(1 - \tanh^2\left(\frac{1}{2}\begin{bmatrix} -0.9330 \\ -2.9932 \\ -0.9930 \end{bmatrix} - 2\right)\right)\right)$$

$$= \begin{bmatrix} -1.9865 \\ 0.0018 \\ 0.0135 \end{bmatrix} \circ \begin{bmatrix} 0.0134 \\ 0.0018 \\ 0.0134 \end{bmatrix} = \begin{bmatrix} -0.0266 \\ 3.3687 \times 10^{-6} \\ 1.8046 \times 10^{-4} \end{bmatrix}$$

⇒ Para as restantes camadas, utiliza-se a fórmula: $\delta^{[p]} = \left(\frac{\partial z^{[p+1]}}{\partial x^{[p]}}\right)^T \cdot \delta^{[p+1]} \circ \frac{\partial x^{[p]}}{\partial z^{[p]}} = W^{[p+1]^T} \cdot \delta^{[p+1]} \circ \phi'^{[p]}(z^{[p]})$

$$\delta^{2(1)} = W^{3^T} \cdot \delta^{3(1)} \circ f'(z^{2(1)}) = \begin{bmatrix} 1 & 3 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0.0068 \\ -0.3177 \\ 0.0068 \end{bmatrix} \circ \begin{bmatrix} 0.3385 \\ 0.3339 \end{bmatrix} = \begin{bmatrix} -0.3745 \\ -0.1016 \end{bmatrix}$$

$$\delta^{1(1)} = W^{2^T} \cdot \delta^{2(1)} \circ f'(z^{1(1)}) = \begin{bmatrix} 1 & 1 \\ 4 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -0.3745 \\ -0.1016 \end{bmatrix} \circ \begin{bmatrix} 0.3932 \\ 0.2100 \\ 0.3932 \end{bmatrix} = \begin{bmatrix} -0.1872 \\ -0.3359 \\ -0.1872 \end{bmatrix}$$

$$\delta^{2(2)} = W^{3^T} \cdot \delta^{3(2)} \circ f'(z^{2(2)}) = \begin{bmatrix} 1 & 3 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.0266 \\ 3.3687 \times 10^{-6} \\ 1.8046 \times 10^{-4} \end{bmatrix} \circ \begin{bmatrix} 4.3587 \times 10^{-4} \\ 0.006546 \end{bmatrix} = \begin{bmatrix} -1.1509 \times 10^{-5} \\ -1.7290 \times 10^{-4} \end{bmatrix}$$

$$\delta^{1(2)} = W^{2^T} \cdot \delta^{2(2)} \circ f'(z^{1(2)}) = \begin{bmatrix} 1 & 1 \\ 4 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -1.1509 \times 10^{-5} \\ -1.7290 \times 10^{-4} \end{bmatrix} \circ \begin{bmatrix} 0.09035 \\ 0.09035 \\ 0.09035 \end{bmatrix} = \begin{bmatrix} -1.6619 \times 10^{-5} \\ -1.9782 \times 10^{-5} \\ -1.6619 \times 10^{-5} \end{bmatrix}$$

Atualização dos pesos

⇒ Para atualizar os pesos, usamos a seguinte fórmula: $W^{[p]} = W^{[p]} - \eta \underbrace{\frac{\partial E}{\partial W^{[p]}}}_{= \delta^{[p]} x^{[p-1]^T}}$

$$\frac{\partial E}{\partial W^1} = \delta^{1(1)} \frac{\partial z^{1(1)}}{\partial w^1} + \delta^{1(2)} \frac{\partial z^{1(2)}}{\partial w^1} = \begin{bmatrix} -0.1872 \\ -0.3359 \\ -0.1872 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} -1.6619 \times 10^{-5} \\ -1.9782 \times 10^{-5} \\ -1.6619 \times 10^{-5} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -0.1872 & -0.1872 & -0.1872 & -0.1872 \\ -0.3359 & -0.3359 & -0.3359 & -0.3359 \\ -0.1872 & -0.1872 & -0.1872 & -0.1872 \end{bmatrix}$$

($x^{i-1(1)} = x_1$, $x^{i-1(2)} = x_2$)

$$W^1_{(novo)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} - 0.1 \begin{bmatrix} -0.1872 & -0.1872 & -0.1872 & -0.1872 \\ -0.3359 & -0.3359 & -0.3359 & -0.3359 \\ -0.1872 & -0.1872 & -0.1872 & -0.1872 \end{bmatrix} = \begin{bmatrix} 1.0187 & 1.0187 & 1.0187 & 1.0187 \\ 1.0335 & 1.0335 & 1.0335 & 1.0335 \\ 1.0187 & 1.0187 & 1.0187 & 1.0187 \end{bmatrix}$$

$$\frac{\partial E}{\partial W^2} = \delta^{2(1)} \frac{\partial z^{2(1)}}{\partial w^2} + \delta^{2(2)} \frac{\partial z^{2(2)}}{\partial w^2} = \begin{bmatrix} -0.3745 \\ -0.1016 \end{bmatrix} \begin{bmatrix} 0.4621 \\ 0.7616 \\ 0.4621 \end{bmatrix}^T + \begin{bmatrix} -1.1509 \times 10^{-5} \\ -1.7290 \times 10^{-4} \end{bmatrix} \begin{bmatrix} -0.9051 \\ -0.9051 \\ -0.9051 \end{bmatrix}^T = \begin{bmatrix} -0.1730 & -0.2852 & -0.1730 \\ -0.0468 & -0.0772 & -0.0468 \end{bmatrix}$$

($x^{1(1)}$, $x^{1(2)}$)

$$W^2_{(novo)} = \begin{bmatrix} 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix} - 0.1 \begin{bmatrix} -0.1730 & -0.2852 & -0.1730 \\ -0.0468 & -0.0772 & -0.0468 \end{bmatrix} = \begin{bmatrix} 1.0173 & 1.0285 & 1.0173 \\ 1.0047 & 1.0077 & 1.0047 \end{bmatrix}$$

$$\frac{\partial E}{\partial W^3} = \delta^{3(1)} \frac{\partial z^{3(1)}}{\partial w^3} + \delta^{3(2)} \frac{\partial z^{(2)}}{\partial w^3} = \begin{bmatrix} 0.0068 \\ -0.3177 \\ \end{bmatrix} \begin{bmatrix} 0.4505 \\ -0.5764 \end{bmatrix}^T + \begin{bmatrix} -0.0266 \\ 3.3687 \times 10^{-6} \\ \end{bmatrix} \begin{bmatrix} -0.9996 \\ -0.9934 \end{bmatrix}^T = \begin{bmatrix} 0.0296 & 0.0225 \\ -0.1431 & 0.1831 \\ \end{bmatrix}$$

($x^{1(1)}$, $x^{1(2)}$)

$$\frac{\partial E}{\partial w^3} = \delta^{3(1)}\frac{\partial z^{3(1)}}{\partial w^3} + \delta^{3(2)}\frac{\partial z^{(2)}}{\partial w^3} = \begin{bmatrix} 0.0068 \\ -0.3177 \\ 0.0068 \end{bmatrix}\begin{bmatrix} 0.4505 \\ -0.5764 \end{bmatrix}^T + \begin{bmatrix} -0.0266 \\ 3.3687\times10^{-6} \\ 1.8046\times10^{-4} \end{bmatrix}\begin{bmatrix} -0.9996 \\ -0.9934 \end{bmatrix}^T = \begin{bmatrix} 0.0296 & 0.0225 \\ -0.1931 & 0.1831 \\ 0.0029 & -0.0041 \end{bmatrix}$$

$$W^3_{(novo)} = \begin{bmatrix} 1 & 1 \\ 3 & 1 \\ 1 & 1 \end{bmatrix} - 0.1\begin{bmatrix} 0.0296 & 0.0225 \\ -0.1931 & 0.1831 \\ 0.0029 & -0.0041 \end{bmatrix} = \begin{bmatrix} 0.9970 & 0.9977 \\ 3.0193 & 0.9817 \\ 0.9997 & 1.0004 \end{bmatrix}$$

Atualização dos bias

⇒ Para proceder à atualização dos bias, utilizamos a fórmula:

$$b^{[p]} = b^{[p]} - \eta\frac{\partial E}{\partial b^{[p]}}, \text{ onde } \frac{\partial E}{\partial b^{[p]}} = \delta^{[p]}\underbrace{\frac{\partial z^{[p]}}{\partial b^{[p]}}}_{=1}; \text{ logo, } \boxed{b^{[p]} = b^{[p]} - \eta\delta^{[p]}}$$

$$b^1_{(new)} = b^1 - \eta\left(\delta^{1(1)} + \delta^{1(2)}\right) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - 0.1\left(\begin{bmatrix} -0.1872 \\ -0.3359 \\ -0.1872 \end{bmatrix} + \begin{bmatrix} -1.6619\times10^{-5} \\ -1.9782\times10^{-5} \\ -1.6619\times10^{-5} \end{bmatrix}\right) = \begin{bmatrix} 1.0187 \\ 1.0336 \\ 1.0187 \end{bmatrix}$$

$$b^2_{(new)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 0.1\left(\begin{bmatrix} -0.3745 \\ -0.1016 \end{bmatrix} + \begin{bmatrix} -1.1509\times10^{-5} \\ -1.7290\times10^{-4} \end{bmatrix}\right) = \begin{bmatrix} 1.0374 \\ 1.0102 \end{bmatrix}$$

$$b^3_{(new)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - 0.1\left(\begin{bmatrix} 0.0068 \\ -0.3177 \\ 0.0068 \end{bmatrix} + \begin{bmatrix} -0.0266 \\ 3.3687\times10^{-6} \\ 1.8046\times10^{-4} \end{bmatrix}\right) = \begin{bmatrix} 1.0020 \\ 1.0318 \\ 0.9993 \end{bmatrix}$$

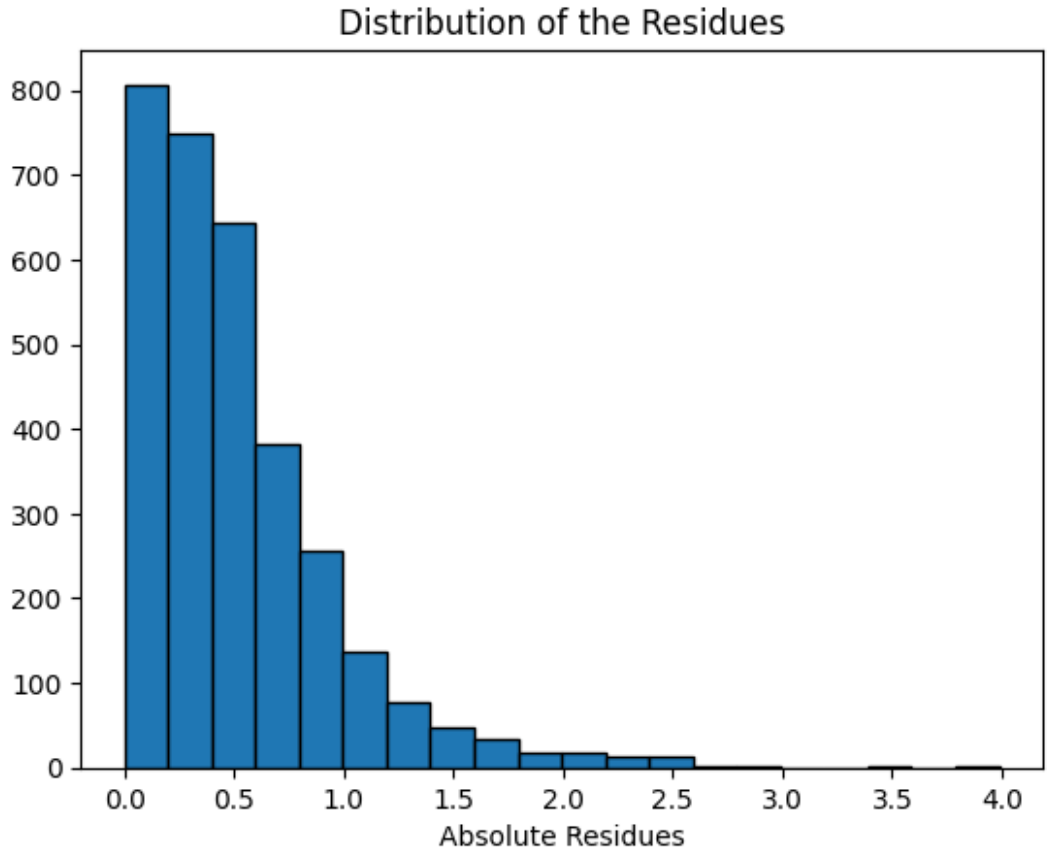
Consider the winequality-red.csv dataset (available at the webpage) where the goal is to estimate the quality (sensory appreciation) of a wine based on physicochemical inputs. Using a 80-20 training-test split with a fixed seed (random_state=0), you are asked to learn MLP regressors to answer the following questions. Given their stochastic behavior, average the performance of each MLP from 10 runs (for reproducibility consider seeding the MLPs with random_state $\in\{1..10\}$).

1.

Learn a MLP regressor with 2 hidden layers of size 10, rectifier linear unit activation on all nodes, and early stopping with 20% of training data set aside for validation. All remaining parameters (e.g., loss, batch size, regularization term, solver) should be set as default. Plot the distribution of the residues (in absolute value) using a histogram.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
import matplotlib.pyplot as plt
from warnings import filterwarnings
filterwarnings('ignore')

# winequality-red.csv dataset
data = pd.read_csv("winequality-red.csv", delimiter=';')
X = data.drop('quality', axis=1)
y = np.ravel(data['quality'])

# 80-20 training-test split with a fixed seed
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

residues = []
num_runs = 10

# average the performance of each MLP Regressor over 10 runs
for seed in range(1, num_runs + 1):
    # Mlp regressor
    mlp_regressor = MLPRegressor(hidden_layer_sizes=(10, 10),
                                 activation='relu',
                                 early_stopping=True,
                                 validation_fraction=0.2,
                                 random_state=seed)

    mlp_regressor.fit(X_train, y_train)
    y_pred = mlp_regressor.predict(X_test)

    # residues = y_test - y_pred
```

```
    residues.extend(np.abs(y_test - y_pred))

# Plot the distribution of the residues (in absolute value) using a
histogram
plt.hist(residues, bins=20, edgecolor='k')
plt.xlabel('Absolute Residues')
plt.title('Distribution of the Residues')
plt.show()
```



2.

Since we are in the presence of a integer regression task, a recommended trick is to round and bound estimates. Assess the impact of these operations on the MAE of the MLP learnt in previous question.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error
from warnings import filterwarnings
```

```
filterwarnings('ignore')

# winequality-red.csv dataset
data = pd.read_csv("winequality-red.csv", delimiter=';')
X = data.drop('quality', axis=1)
y = np.ravel(data['quality'])

# 80-20 training-test split with a fixed seed
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

mae_original = []
mae_with_round_and_bound = []
num_runs = 10

#bounds
bound1 = 1
bound2 = 10

for seed in range(1, num_runs + 1):
    mlp_regressor = MLPRegressor(hidden_layer_sizes=(10, 10),
                                 activation='relu',
                                 early_stopping=True,
                                 validation_fraction=0.2,
                                 random_state=seed)

    mlp_regressor.fit(X_train, y_train)
    y_pred = mlp_regressor.predict(X_test)


    # MAE without rounding or bounding
    mae_original.append(mean_absolute_error(y_test, y_pred))

    # MAE with rounding and bounding
    y_pred_rounded = np.round(y_pred)
    y_pred_round_bound = np.clip(y_pred_rounded, bound1, bound2)
    mae_with_round_and_bound.append(mean_absolute_error(y_test,
y_pred_round_bound))

#calculate average MAE
avg_mae = np.mean(mae_original)
avg_mae_round_bound = np.mean(mae_with_round_and_bound)

print(f"MAE: {avg_mae:.6f}")
print(f"MAE with rounding and bounding: {avg_mae_round_bound:.6f}")

MAE: 0.509717
MAE with rounding and bounding: 0.438750
```

3.

Similarly assess the impact on RMSE from replacing early stopping by a well-defined number of iterations in {20,50,100,200} (where one iteration corresponds to a batch).

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from warnings import filterwarnings
filterwarnings('ignore')

# winequality-red.csv dataset
data = pd.read_csv("winequality-red.csv", delimiter=';')
X = data.drop('quality', axis=1)
y = np.ravel(data['quality'])

# 80-20 training-test split with a fixed seed
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

num_runs = 10
rmse_values = []
iteration_counts = [20, 50, 100, 200]

for num_iterations in iteration_counts:
    rmse_iterations = []

    for seed in range(1, num_runs + 1):

        mlp_regressor = MLPRegressor(hidden_layer_sizes=(10, 10),
                                     activation='relu',
                                     max_iter=num_iterations,
                                     random_state=seed)

        mlp_regressor.fit(X_train, y_train)
        y_pred = mlp_regressor.predict(X_test)

        rmse = np.sqrt(mean_squared_error(y_test,y_pred))
        rmse_iterations.append(rmse)

    rmse_values.append(np.mean(rmse_iterations))

for i, num_iterations in enumerate(iteration_counts):
    print(f"Average RMSE with {num_iterations} iterations:
{rmse_values[i]:.5f}")
```

```
Average RMSE with 20 iterations: 1.40398
Average RMSE with 50 iterations: 0.79961
```

```
Average RMSE with 100 iterations: 0.69404
Average RMSE with 200 iterations: 0.65545
```

4.

Critically comment the results obtained in previous question, hypothesizing at least one reason why early stopping favors and/or worsens performance.

Analisando os resultados obtidos na questão anterior, podemos observar que a diminuição mais acentuada do RMSE ocorre entre 20 e 50 iterações (diminuição de 0.60437), e diminui de modo algo significativo ainda entre as 50 e 100 iterações (diminuição de 0.10557); entre as 100 e 200 iterações, o RMSE mal diminui (diminuição de 0.03858), indicando que começa a estagnar em torno desse valor (0.65545). Podemos então concluir que, com base na diferença entre as 20 e 50 iterações o modelo aqui ainda é passível de melhoria, e mais iterações serão benéficas para melhorar a sua performance, e ainda não se encontra a convergir para uma solução ótima; com base na diferença entre as 50 e 100 iterações, percebemos que a performance do modelo ainda melhorou mas que este começa a convergir, e que apesar de se poderem executar mais iterações, a melhoria será cada vez mais pequena. Finalmente, observando a pequena diferença entre as 100 e 200 iterações, conclui-se que mais iterações dificilmente irão favorecer a performance do modelo, que aparenta estar a convergir.

Como visto nas teóricas, o early stopping é uma boa técnica para prevenir overfitting do modelo, que no caso de iterações bem definidas aparenta ocorrer a partir das 200; portanto, apresenta-se como melhor alternativa para o caso de iterações bem definidas superiores às 200. Comparando com o RMSE para 50, 100 e 200 iterações bem definidas, este varia entre 0.79961 (50 iterações) e 0.65545 (200 iterações), sendo as iterações bem definidas uma melhor alternativa ao early stopping neste intervalo, que apresenta um RMSE de 0.81888. Finalmente, o early stopping constitui uma melhor atlternativa ao caso de 20 iterações bem definidas (em que o RMSE é 1.40398), onde o modelo aparenta não estar próximo de convergir ainda, com um erro elevado.