Homework III – Group 35
(ist1100032, ist1100070)

I. Pen-and-paper

1)

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

$z = (0,8; 0,6; 0,3; 0,3)$

$X = \left\{ \begin{pmatrix} 0,7 \\ -0,3 \end{pmatrix}, \begin{pmatrix} 0,4 \\ 0,5 \end{pmatrix}, \begin{pmatrix} -0,2 \\ 0,8 \end{pmatrix}, \begin{pmatrix} -0,4 \\ 0,3 \end{pmatrix} \right\}$

$c_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $\quad c_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ $\quad c_3 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

Ridge: $W = (X^T X + \lambda I)^{-1} X^T z$, $\lambda = 0,1$

$\phi_1\begin{pmatrix} 0,7 \\ -0,3 \end{pmatrix} = 0,74826$ $\qquad \phi_2\begin{pmatrix} 0,7 \\ -0,3 \end{pmatrix} = 0,74826$

$\phi_1\begin{pmatrix} 0,4 \\ 0,5 \end{pmatrix} = 0,81465$ $\qquad \phi_2\begin{pmatrix} 0,4 \\ 0,5 \end{pmatrix} = 0,27117$

$\phi_1\begin{pmatrix} -0,2 \\ 0,8 \end{pmatrix} = 0,71177$ $\qquad \phi_2\begin{pmatrix} -0,2 \\ 0,8 \end{pmatrix} = 0,09633$

$\phi_1\begin{pmatrix} -0,4 \\ 0,3 \end{pmatrix} = 0,88250$ $\qquad \phi_2\begin{pmatrix} -0,4 \\ 0,3 \end{pmatrix} = 0,16122$

$\phi_3\begin{pmatrix} 0,7 \\ -0,3 \end{pmatrix} = 0,10127$

$\phi_3\begin{pmatrix} 0,4 \\ 0,5 \end{pmatrix} = 0,33121$

$\phi_3\begin{pmatrix} -0,2 \\ 0,8 \end{pmatrix} = 0,71177$

$\phi_3\begin{pmatrix} -0,4 \\ 0,3 \end{pmatrix} = 0,65377$

$$X = \begin{bmatrix} 1 & 0,74826 & 0,74826 & 0,10127 \\ 1 & 0,81465 & 0,27117 & 0,33121 \\ 1 & 0,71177 & 0,09633 & 0,71177 \\ 1 & 0,88250 & 0,16122 & 0,65377 \end{bmatrix}$$

Homework III – Group 35

(ist1100032, ist1100070)

$$X^T X = \begin{bmatrix} 4 & 3{,}15718 & 1{,}27698 & 1{,}79802 \\ 3{,}15718 & 2{,}50897 & 0{,}99165 & 1{,}42916 \\ 1{,}27698 & 0{,}99165 & 0{,}66870 & 0{,}33955 \\ 1{,}79802 & 1{,}42916 & 0{,}33955 & 1{,}05399 \end{bmatrix}$$

$$X^T X + \lambda I = \begin{bmatrix} 4{,}1 & 3{,}15718 & 1{,}27698 & 1{,}79802 \\ 3{,}15718 & 2{,}60897 & 0{,}99165 & 1{,}42916 \\ 1{,}27698 & 0{,}99165 & 0{,}76870 & 0{,}33955 \\ 1{,}79802 & 1{,}42916 & 0{,}33955 & 1{,}15399 \end{bmatrix}$$

$$(X^T X + \lambda I)^{-1} = \begin{bmatrix} 4{,}54826 & -3{,}77682 & -1{,}86117 & -1{,}86155 \\ -3{,}77682 & 5{,}98285 & -0{,}88543 & -1{,}26432 \\ -1{,}86117 & -0{,}88543 & 4{,}33276 & 2{,}72156 \\ -1{,}86155 & -1{,}26432 & 2{,}72156 & 4{,}53204 \end{bmatrix}$$

$$(X^T X + \lambda I)^{-1} X^T = \begin{bmatrix} 0{,}14105 & 0{,}35022 & 0{,}35575 & -0{,}30185 \\ -0{,}09064 & 0{,}43823 & -0{,}50361 & 0{,}53370 \\ 0{,}99394 & -0{,}50615 & -0{,}13610 & -0{,}16477 \\ -0{,}31222 & -0{,}65246 & 0{,}72647 & 0{,}42436 \end{bmatrix}$$

$$(X^T X + \lambda I)^{-1} X^T y = \begin{bmatrix} 0{,}33914 & 0{,}19945 & 0{,}40096 & -0{,}29600 \end{bmatrix}$$

b- Predictions:

$$X \cdot W = [\,0{,}75844 \quad 0{,}51232 \quad 0{,}30905 \quad 0{,}38629\,]$$

$$RMSE = \sqrt{\sum_{i=1}^{m} \frac{(\hat{y}_i - y_i)^2}{m}} =$$

$$= \sqrt{\frac{(0{,}75844 - 0{,}8)^2}{4} + \frac{(0{,}51232 - 0{,}6)^2}{4} + \frac{(0{,}30905 - 0{,}3)^2}{4} + \frac{(0{,}38629 - 0{,}3)^2}{4}} = 0{,}065082$$

Código Python auxiliar:

1a)

```python
import numpy as np

column1 = [0.7, 0.4, -0.2, -0.4]
column2 = [-0.3, 0.5, 0.8, 0.3]
column3 = [0.8, 0.6, 0.3, 0.3]

X = np.column_stack((column1, column2))

y = np.array(column3)

X
```

```
array([[ 0.7, -0.3],
       [ 0.4,  0.5],
       [-0.2,  0.8],
       [-0.4,  0.3]])
```

```python
# Regularization parameter (λ)
alpha = 0.1

# Define the RBF centers
centers = [(0, 0), (1, -1), (-1, 1)]

# Initialize an empty array for the transformed data
X_transformed = np.zeros((len(column1), len(centers)))

# Apply the radial basis function for each center
for j, center in enumerate(centers):
    for i in range(len(column1)):
        x = np.array([column1[i], column2[i]])
        c = np.array(center)
        # Compute the radial basis function value and store it in the transformed_data array
        X_transformed[i, j] = np.exp(-np.linalg.norm(x - c)**2 / 2)

# Add a column of 1s for the bias term
bias_column = np.ones((len(column1), 1))
X_transformed = np.hstack((bias_column, X_transformed))

print(X_transformed)
```

```
[[1.         0.74826357 0.74826357 0.10126646]
 [1.         0.81464732 0.27117254 0.33121088]
 [1.         0.71177032 0.09632764 0.71177032]
 [1.         0.8824969  0.16121764 0.65376979]]
```

Learn the Ridge regression (*l*2 regularization) using the closed solution

```python
np.matmul(X_transformed.T, X_transformed)
```

```
array([[4.        , 3.15717811, 1.27698138, 1.79801745],
       [3.15717811, 2.50896639, 0.99164557, 1.42916086],
       [1.27698138, 0.99164557, 0.66870305, 0.33955168],
       [1.79801745, 1.42916086, 0.33955168, 1.05398747]])
```

```python
np.matmul(X_transformed.T, X_transformed) + alpha * np.identity(4)
```

```
array([[4.1       , 3.15717811, 1.27698138, 1.79801745],
       [3.15717811, 2.60896639, 0.99164557, 1.42916086],
       [1.27698138, 0.99164557, 0.76870305, 0.33955168],
       [1.79801745, 1.42916086, 0.33955168, 1.15398747]])
```

```
inv = np.linalg.pinv(np.matmul(X_transformed.T, X_transformed) + alpha * np.identity(4))
inv
```

```
array([[ 4.54826202, -3.77681832, -1.86116983, -1.86155421],
       [-3.77681832,  5.98284561, -0.88542926, -1.26432443],
       [-1.86116983, -0.88542926,  4.33275508,  2.72155678],
       [-1.86155421, -1.26432443,  2.72155678,  4.53204296]])
```

```
moore_penrose = np.matmul(inv, X_transformed.T)
moore_penrose
```

```
array([[ 0.14104789,  0.35022196,  0.3557537 , -0.30184975],
       [-0.09064104,  0.43822869, -0.50360629,  0.53370047],
       [ 0.99394091, -0.506149  , -0.13690469, -0.16477025],
       [-0.31221638, -0.65245932,  0.726472  ,  0.42435912]])
```

```
W_ridge = np.matmul(moore_penrose, y)
W_ridge
```

```
array([ 0.33914267,  0.19945264,  0.40096085, -0.29599936])
```

1b)

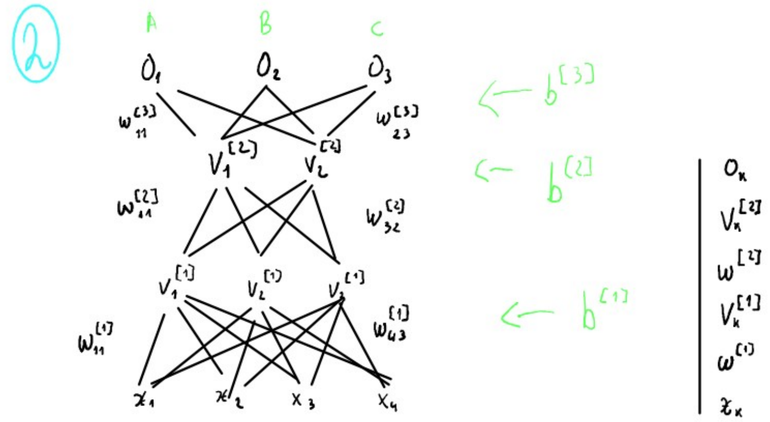
```
y_pred = np.dot(X_transformed, W_ridge)

rmse = np.sqrt(np.mean((y - y_pred)**2))

print("RMSE:", rmse)
```

```
RMSE: 0.06508238153393446
```

Homework III – Group 35

(ist1100032, ist1100070)

2)



$$W^{[1]} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, b^{[1]} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, W^{[2]} = \begin{pmatrix} 1 & 4 & 1 \\ 1 & 1 & 1 \end{pmatrix}, b^{[2]} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, W^{[3]} = \begin{pmatrix} 1 & 1 \\ 3 & 1 \\ 1 & 1 \end{pmatrix}, b^{[3]} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$X^{[0](1)} = x_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \hat{z}_1 = B \qquad \eta = 0.1$$

$$X^{[0](2)} = x_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} \quad \hat{z}_2 = A$$

$$\text{activation} = f(x) = \frac{e^{0.5x-2} - e^{-0.5x+2}}{e^{0.5x-2} + e^{-0.5x+2}} = tanh(0.5x - 2)$$

Squared error loss : $\frac{1}{2} \| z - \hat{z} \|_2^2$

Forward propagation:

input $X^{[0]\boxed{(1)}} = x_{\boxed{1}} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

- $$Z^{[1](1)} = W^{[1]} \cdot X^{[0](1)} + b^{[1]} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{3\times4} \cdot \underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}}_{4\times1} + \underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}_{3\times1} = \begin{bmatrix} 5 \\ 6 \\ 5 \end{bmatrix}$$

- $$X^{[1](1)} = \text{activation}(z^{[1](1)}) = \underbrace{\begin{bmatrix} 0.46212 \\ 0.76159 \\ 0.46212 \end{bmatrix}}_{3\times1}$$

- $$Z^{[2](1)} = W^{[2]} \cdot X^{[1](1)} + b^{[2]} = \underbrace{\begin{bmatrix} 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}}_{2\times3} \cdot \underbrace{\begin{bmatrix} 0.46212 \\ 0.76159 \\ 0.46212 \end{bmatrix}}_{3\times1} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 4.97061 \\ 2.68582 \end{bmatrix}}_{2\times1}$$

- $X^{[2](1)} = activation(z^{[2](1)}) = \begin{bmatrix} 0.45068 \\ -0.57642 \end{bmatrix}_{2\times1}$

- $Z^{[3](1)} = W^{[3]} \cdot X^{[2](1)} + b^{[3]} = \underset{3\times2}{\begin{bmatrix} 1 & 1 \\ 3 & 1 \\ 1 & 1 \end{bmatrix}} \cdot \begin{bmatrix} 0.45068 \\ -0.57642 \end{bmatrix} + \underset{3\times1}{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}} = \underset{3\times1}{\begin{bmatrix} 0.87406 \\ 1.77502 \\ 0.87406 \end{bmatrix}}$

- $X^{[3](1)} = activation(z^{[3](1)}) = \underset{3\times1}{\begin{bmatrix} -0.91590 \\ -0.80493 \\ -0.91590 \end{bmatrix}}$

input $X^{[0]\boxed{(2)}} = x_{\boxed{2}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$

- $Z^{[1](2)} = W^{[1]} \cdot X^{[0](2)} + b^{[1]} = \underset{3\times4}{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}} \cdot \underset{4\times1}{\begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}} + \underset{3\times1}{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

- $X^{[1](2)} = activation(z^{[1](2)}) = \underset{3\times1}{\begin{bmatrix} -0.90515 \\ -0.90515 \\ -0.90515 \end{bmatrix}}$

- $Z^{[2](2)} = W^{[2]} \cdot X^{[1](2)} + b^{[2]} = \underset{2\times3}{\begin{bmatrix} 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}} \cdot \begin{bmatrix} -0.90515 \\ -0.90515 \\ -0.90515 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -4.43089 \\ -1.71544 \end{bmatrix}$

- $x^{[2](2)} = \text{activation}(z^{[2](2)}) = \underset{2\times 1}{\begin{bmatrix} -0.99056 \\ -0.99343 \end{bmatrix}}$

- $z^{[3](2)} = w^{[3]} \cdot x^{[2](2)} + b^{[3]} = \underset{3\times 2}{\begin{bmatrix} 1 & 1 \\ 3 & 1 \\ 1 & 1 \end{bmatrix}} \cdot \underset{2\times 1}{\begin{bmatrix} -0.99056 \\ -0.99343 \end{bmatrix}} + \underset{3\times 1}{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}} = \underset{3\times 1}{\begin{bmatrix} -0.99299 \\ -2.99212 \\ -0.99299 \end{bmatrix}}$

- $x^{[3](2)} = \text{activation}(z^{[3](2)}) = \underset{3\times 1}{\begin{bmatrix} -0.98652 \\ -0.99816 \\ -0.98632 \end{bmatrix}}$

Backpropagation

derive the activation function:

- $\dfrac{df(x)}{dx} = \dfrac{\partial \tanh(0.5x-2)}{\partial x} = \operatorname{sech}^2(0.5x-2)\cdot 0.5$

- $E(t, x^{[i]}) = \frac{1}{2}(x^{[i]} - t)^2$

- $\dfrac{\partial E(t, x^{[i]})}{\partial x^{[i]}} = x^{[i]} - t$

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

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

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

cálculo dos δ $\rightsquigarrow$ squared error loss

$\delta^{[3]}$ (last layer)

- $$\delta^{[3](1)} = \frac{\partial E}{\partial x^{[3](1)}} \circ \frac{\partial x^{[3](1)}}{\partial z^{[3](1)}} = \left(x^{[3](1)} - t^{(1)}\right) \circ \frac{\partial(\tanh(0.5 \cdot z^{[3](1)} - 2))}{\partial z^{[3](1)}} =$$

pwder rual

$$= \left(x^{[3](1)} - t^{(1)}\right) \circ \left(0{,}5 \cdot \operatorname{sech}^2(0{,}5\, z^{[3](1)})\right) = \begin{bmatrix} 0.00677 \\ -0.31773 \\ 0.00677 \end{bmatrix}$$

- $$\delta^{[3](2)} = \frac{\partial E}{\partial x^{[3](2)}} \circ \frac{\partial x^{[3](2)}}{\partial z^{[3](2)}} = \left(x^{[3](2)} - t^{(2)}\right) \circ \frac{\partial(\tanh(0.5 \cdot z^{[3](2)} - 2))}{\partial z^{[3](2)}} =$$

pwder rual

$$= \left(x^{[3](1)} - t^{(1)}\right) \circ \left(0{,}5 \cdot \operatorname{sech}^2(0{,}5\, z^{[3](1)})\right) = \begin{bmatrix} -2.659 \times 10^{-2} \\ 3.369 \times 10^{-6} \\ 1.804 \times 10^{-4} \end{bmatrix}$$

$\delta^{[2]}$

$\rightarrow \tanh(0.5 \cdot z^{[2](1)} - 2)$

- $$\delta^{[2](1)} = \left(\frac{\partial z^{[3](1)}}{\partial x^{[2](1)}}\right)^T \cdot \delta^{[3](1)} \circ \frac{\partial x^{[2](1)}}{\partial z^{[2](1)}} = (w^{[3]})^T \cdot \delta^{[3](1)} \circ \left(0{,}5 \cdot \operatorname{sech}^2(0{,}5\, z^{[2](1)})\right) =$$

$$= \begin{bmatrix} -0.37448 \\ -0.10156 \end{bmatrix}$$

- $$\delta^{[2](2)} = \left(\frac{\partial z^{[3](2)}}{\partial x^{[2](2)}}\right)^T \cdot \delta^{[3](2)} \circ \frac{\partial x^{[2](2)}}{\partial z^{[2](2)}} = (w^{[3]})^T \cdot \delta^{[3](2)} \circ \left(0{,}5 \cdot \operatorname{sech}^2(0{,}5\, z^{[2](2)})\right) =$$

$$= \begin{bmatrix} -1.13 \times 10^{-5} \\ -1.729 \times 10^{-4} \end{bmatrix}$$

$\delta^{[1]}$

$\to \tanh(0.5 \cdot z^{[2](1)} - 2)$

- $\delta^{[1](1)} = \left(\frac{\partial z^{[2](1)}}{\partial x^{[1](1)}}\right)^T \cdot \delta^{[2](1)} \circ \frac{\partial x^{[1](1)}}{\partial z^{[1](1)}} = (w^{[2]})^T \cdot \delta^{[2](1)} \circ (0{,}5 \cdot \text{sech}^2(0{,}5\, Z^{[1](1)}) =$

$$= \begin{bmatrix} -0.18719 \\ -0.33587 \\ -0.18719 \end{bmatrix}$$

- $\delta^{[1](2)} = \left(\frac{\partial z^{[2](2)}}{\partial x^{[1](2)}}\right)^T \cdot \delta^{[2](2)} \circ \frac{\partial x^{[1](2)}}{\partial z^{[1](2)}} = (w^{[2]})^T \cdot \delta^{[2](2)} \circ (0{,}5 \cdot \text{sech}^2(0{,}5\, Z^{[1](2)}) =$

$$= \begin{bmatrix} -1.666 \times 10^{-5} \\ -1.978 \times 10^{-5} \\ -1.666 \times 10^{-5} \end{bmatrix}$$

Updates

$$w^{[i]} = w^{[i]} - \eta \cdot \frac{\partial E}{\partial w^{[i]}} \rightsquigarrow \frac{\partial E}{\partial w^{[i]}} = \sum_{j=1}^{N} \underbrace{\frac{\partial E}{\partial x^{[i](j)}} \circ \frac{\partial x^{[i](j)}}{\partial z^{[i](j)}}}_{\delta^{[i](j)}} \cdot \underbrace{\left(\frac{\partial z^{[i](j)}}{\partial w^{[i](j)}}\right)^T}_{(x^{[i-1](j)})^T}$$

$N \to$ nº obs batch

$$b^{[i]} = b^{[i]} - \eta \cdot \frac{\partial E}{\partial b^{[i]}} \rightsquigarrow \frac{\partial E}{\partial b^{[i]}} = \sum_{j=1}^{N} \delta^{[i](j)}$$

nível 3

- $w^{[3]} = w^{[3]} - \eta \cdot \frac{\partial E}{\partial w^{[3]}} = \begin{bmatrix} 0.99703 & 0.99775 \\ 3.01431 & 0.98169 \\ 0.99871 & 1.00041 \end{bmatrix}$

aux

- $\frac{\partial E}{\partial w^{[3]}} = \delta^{[3](1)} \cdot (x^{[2](1)})^T + \delta^{[3](2)} \cdot (x^{[2](2)})^T = \begin{bmatrix} 0.02964 & 0.022516 \\ -0.14314 & 0.18314 \\ 0.00287 & -0.00408 \end{bmatrix}$

- $$b^{[3]} = b^{[3]} - \eta \frac{\partial E}{\partial b^{[3]}} = \begin{bmatrix} 1.00198 \\ 1.03177 \\ 0.99930 \end{bmatrix}$$

Aux

- $$\frac{\partial E}{\partial b^{[3]}} = \delta^{[3](1)} + \delta^{[3](2)} = \begin{bmatrix} -0.01982 \\ -0.31773 \\ 0.00696 \end{bmatrix}$$

nível 2

- $$w^{[2]} = w^{[2]} - \eta \cdot \frac{\partial E}{\partial w^{[2]}} = \begin{bmatrix} 1.01730 & 4.02851 & 1.01730 \\ 1.00467 & 1.00772 & 1.00468 \end{bmatrix}$$

aux

- $$\frac{\partial E}{\partial w^{[2]}} = \delta^{[2](1)} \cdot \left(x^{[1](1)}\right)^T + \delta^{[2](2)} \cdot \left(x^{[1](2)}\right)^T = \begin{bmatrix} -0.17304 & -0.28519 & -0.17304 \\ -0.04677 & -0.07719 & -0.04677 \end{bmatrix}$$

- $$b^{[2]} = b^{[2]} - \eta \frac{\partial E}{\partial b^{[2]}} = \begin{bmatrix} 1.03743 \\ 1.01017 \end{bmatrix}$$

Aux

- $$\frac{\partial E}{\partial b^{[2]}} = \delta^{[2](1)} + \delta^{[2](2)} = \begin{bmatrix} -0.37449 \\ -0.10173 \end{bmatrix}$$

Homework III – Group 35

(ist1100032, ist1100070)

nível 1

- $$w^{[1]} = w^{[1]} - \eta \cdot \frac{\partial E}{\partial w^{[1]}} = \begin{bmatrix} 1.01872 & 1.01872 & 1.01872 & 1.01872 \\ 1.03359 & 1.03359 & 2.03359 & 1.03338 \\ 1.01872 & 1.01872 & 1.01872 & 1.01871 \end{bmatrix}$$

aux

- $$\frac{\partial E}{\partial w^{[1]}} = \delta^{[1](1)} \cdot (x^{[0](1)})^T + \delta^{[1](2)} \cdot (x^{[0](2)})^T = \begin{bmatrix} -0.18720 & -0.18719 & -0.18719 & -0.18717 \\ -0.33589 & -0.33587 & -0.33587 & -0.33585 \\ -0.18721 & -0.18719 & -0.18719 & -0.18717 \end{bmatrix}$$

- $$b^{[1]} = b^{[1]} - \eta \frac{\partial E}{\partial b^{[1]}} = \begin{bmatrix} 1.01872 \\ 1.03359 \\ 1.01872 \end{bmatrix}$$

Aux

- $$\frac{\partial E}{\partial b^{[1]}} = \delta^{[1](1)} + \delta^{[1](2)} = \begin{bmatrix} -0.18720 \\ -0.33589 \\ -0.18721 \end{bmatrix}$$

Aprendizagem 2023/24

Homework III – Group 35

(ist1100032, ist1100070)

Código python auxiliar:

```
learning_rate = 0.1

W1 = np.array([[1, 1, 1, 1],
               [1, 1, 2, 1],
               [1, 1, 1, 1]])
b1 = np.array([[1],
               [1],
               [1]])


W2 = np.array([[1, 4, 1],
               [1, 1, 1]])
b2 = np.array([[1],
               [1]])


W3 = np.array([[1, 1],
               [3, 1],
               [1, 1]])
b3 = np.array([[1],
               [1],
               [1]])
```

```
X0_1 = np.array([[1],
                 [1],
                 [1],
                 [1]])
X0_2 = np.array([[1],
                 [0],
                 [0],
                 [-1]])

t1 = np.array([[-1],
               [1],
               [-1]]) #B

t2 = np.array([[1],
               [-1],
               [-1]]) #A

def activation_function(x):
    return np.tanh(0.5 * x - 2)
```

Forward Propagation

Para o input X0_1 (x1 do enunciado) -> _1 indica isso

```
Z1_1 = np.matmul(W1, X0_1) + b1
Z1_1
```

```
array([[5],
       [6],
       [5]])
```

```
X1_1 = activation_function(Z1_1)
X1_1
```

```
array([[0.46211716],
       [0.76159416],
       [0.46211716]])
```

```
Z2_1 = np.matmul(W2, X1_1) + b2
Z2_1
```

```
array([[4.97061094],
       [2.68582847]])
```

```
X2_1 = activation_function(Z2_1)
X2_1
```

```
array([[ 0.45048251],
       [-0.57642073]])
```

```
Z3_1 = np.matmul(W3, X2_1) + b3
Z3_1
```

```
array([[0.87406178],
       [1.77502679],
       [0.87406178]])
```

```
X3_1 = activation_function(Z3_1)
X3_1
```

```
array([[-0.91590016],
       [-0.80493961],
       [-0.91590016]])
```

Para o input X0_2 (x2 do enunciado) _2 indica isso

```
Z1_2 = np.matmul(W1, X0_2) + b1
Z1_2
```

```
array([[1],
       [1],
       [1]])
```

```
X1_2 = activation_function(Z1_2)
X1_2
```

```
array([[-0.90514825],
       [-0.90514825],
       [-0.90514825]])
```

```
Z2_2 = np.matmul(W2, X1_2) + b2
Z2_2
```

```
array([[-4.43088952],
       [-1.71544476]])
```

```
X2_2 = activation_function(Z2_2)
X2_2
```

```
array([[-0.99956404],
       [-0.99343227]])
```

```
Z3_2 = np.matmul(W3, X2_2) + b3
Z3_2
```

```
array([[-0.99299631],
       [-2.99212439],
       [-0.99299631]])
```

```
X3_2 = activation_function(Z3_2)
X3_2
```

```
array([[-0.98652085],
       [-0.9981635 ],
       [-0.98652085]])
```

Backpropagation

derived activation function

```python
def derived_actication_function(x):
    return (1 / (np.cosh(0.5 * x - 2) ** 2)) * 0.5
```

```python
delta3_1 = np.multiply((X3_1 - t1), derived_actication_function(Z3_1))
delta3_1
```

```
array([[ 0.00677537],
       [-0.31773455],
       [ 0.00677537]])
```

```python
delta3_2 = np.multiply((X3_2 - t2), derived_actication_function(Z3_2))
delta3_2
```

```
array([[-2.65961421e-02],
       [ 3.36962051e-06],
       [ 1.80462886e-04]])
```

```python
delta2_1 = np.multiply(np.matmul(W3.T, delta3_1), derived_actication_function(Z2_1))
delta2_1
```

```
array([[-0.37448246],
       [-0.10155772]])
```

```python
delta2_2 = np.multiply(np.matmul(W3.T, delta3_2), derived_actication_function(Z2_2))
delta2_2
```

```
array([[-1.15092599e-05],
       [-1.72899298e-04]])
```

```
delta1_1 = np.multiply(np.matmul(W2.T, delta2_1), derived_actication_function(Z1_1))
delta1_1
```

```
array([[-0.18719036],
       [-0.33587187],
       [-0.18719036]])
```

```
delta1_2 = np.multiply(np.matmul(W2.T, delta2_2), derived_actication_function(Z1_2))
delta1_2
```

```
array([[-1.66619254e-05],
       [-1.97816249e-05],
       [-1.66619254e-05]])
```

Updates dos valores

Nível 3

```
dE_dW3 = (delta3_1 * X2_1.T) + (delta3_2 * X2_2.T)
dE_dW3
```

```
array([[ 0.02963673,  0.022516  ],
       [-0.14313723,  0.18314544],
       [ 0.0028718 , -0.00408474]])
```

```
W3_new = W3 - (learning_rate * dE_dW3)
W3_new
```

```
array([[0.99703633, 0.9977484 ],
       [3.01431372, 0.98168546],
       [0.99971282, 1.00040847]])
```

```
dE_db3 = (delta3_1 + delta3_2)
dE_db3
```

```
array([[-0.01982077],
       [-0.31773118],
       [ 0.00695584]])
```

```
b3_new = b3 - (learning_rate * dE_db3)
b3_new
```

```
array([[1.00198208],
       [1.03177312],
       [0.99930442]])
```

Nível 2

```
dE_dW2 = (delta2_1 * X1_1.T) + (delta2_2 * X1_2.T)
dE_dW2
```

```
array([[-0.17304435, -0.28519324, -0.17304435],
       [-0.04677506, -0.07718926, -0.04677506]])
```

```
W2_new = W2 - (learning_rate * dE_dW2)
W2_new
```

```
array([[1.01730444, 4.02851932, 1.01730444],
       [1.00467751, 1.00771893, 1.00467751]])
```

```
dE_db2 = (delta2_1 + delta2_2)
dE_db2
```

```
array([[-0.37449397],
       [-0.10173062]])
```

```
b2_new = b2 - (learning_rate * dE_db2)
b2_new
```

```
array([[1.0374494 ],
       [1.01017306]])
```

Nível 1

```
dE_dW1 = (delta1_1 * X0_1.T) + (delta1_2 * X0_2.T)
dE_dW1
```

```
array([[-0.18720702, -0.18719036, -0.18719036, -0.1871737 ],
       [-0.33589165, -0.33587187, -0.33587187, -0.33585209],
       [-0.18720702, -0.18719036, -0.18719036, -0.1871737 ]])
```

```
W1_new = W1 - (learning_rate * dE_dW1)
W1_new
```

```
array([[1.0187207 , 1.01871904, 1.01871904, 1.01871737],
       [1.03358917, 1.03358719, 2.03358719, 1.03358521],
       [1.0187207 , 1.01871904, 1.01871904, 1.01871737]])
```

```
dE_db1 = (delta1_1 + delta1_2)
dE_db1
```

```
array([[-0.18720702],
       [-0.33589165],
       [-0.18720702]])
```

```
b1_new = b1 - (learning_rate * dE_db1)
b1_new
```

```
array([[1.0187207 ],
       [1.03358917],
       [1.0187207 ]])
```

Aprendizagem 2023/24

Homework III – Group 35

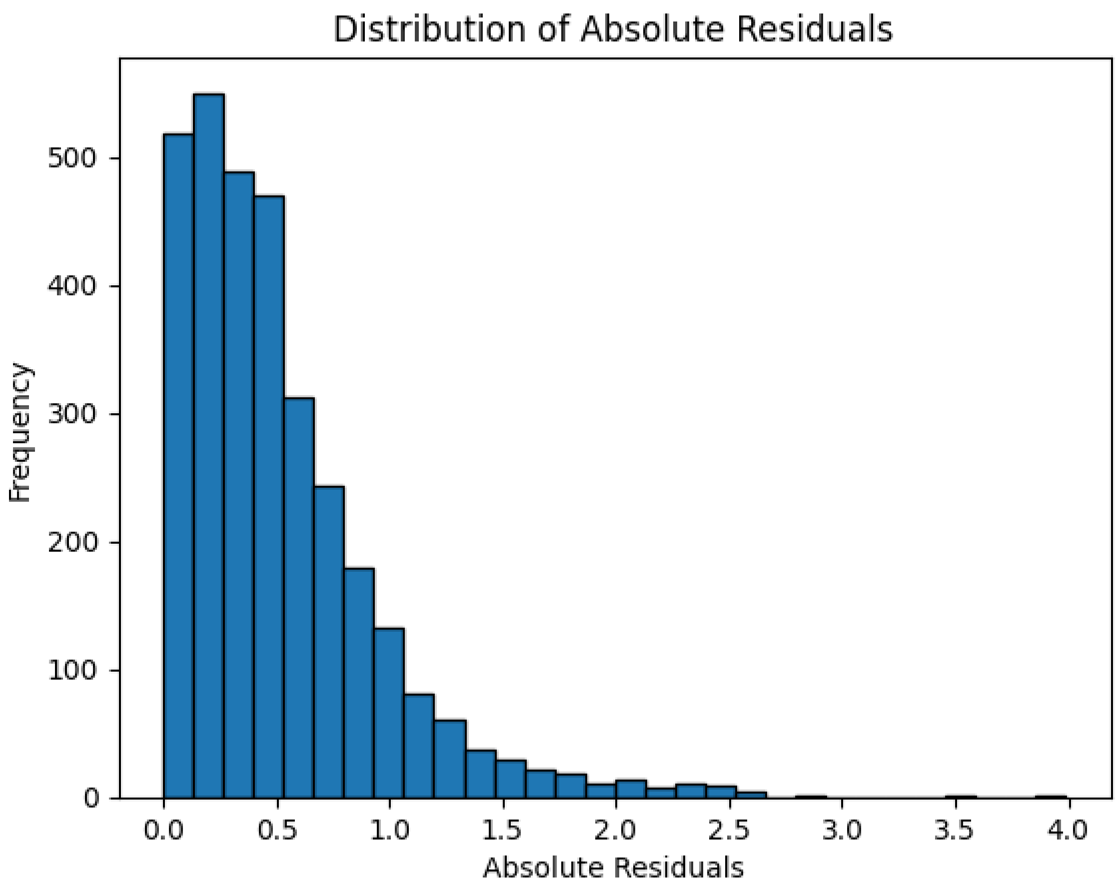
(ist1100032, ist1100070)

II. Programming and critical analysis

1)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error

file_path = "winequality-red.csv"

df = pd.read_csv(file_path, delimiter=';')

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
num_runs = 10
random_states = range(1, num_runs + 1)

residuals = []

mae_original = []
mae_values_rounded_bounded = []

for random_state in random_states:
    mlp = MLPRegressor(hidden_layer_sizes=(10, 10), activation='relu', early_stopping=True, validation_fraction=0.2, random_state=random_state)

    mlp.fit(X_train, y_train)

    y_pred = mlp.predict(X_test)

    residuals.extend(np.abs(y_test - y_pred))

    # For question 2
    # Calculate MAE for original predictions
    mae_original.append(mean_absolute_error(y_test, y_pred))
    # Round and bound the estimates & calculate the MAE
    y_pred_rounded = np.round(y_pred)
    y_pred_rounded_bounded = np.clip(y_pred_rounded, 1, 10)
    mae_rounded_bounded = mean_absolute_error(y_test, y_pred_rounded_bounded)
    mae_values_rounded_bounded.append(mae_rounded_bounded)

plt.hist(residuals, bins=30, edgecolor='k')
plt.xlabel('Absolute Residuals')
plt.ylabel('Frequency')
plt.title('Distribution of Absolute Residuals')
plt.show()
```



2)

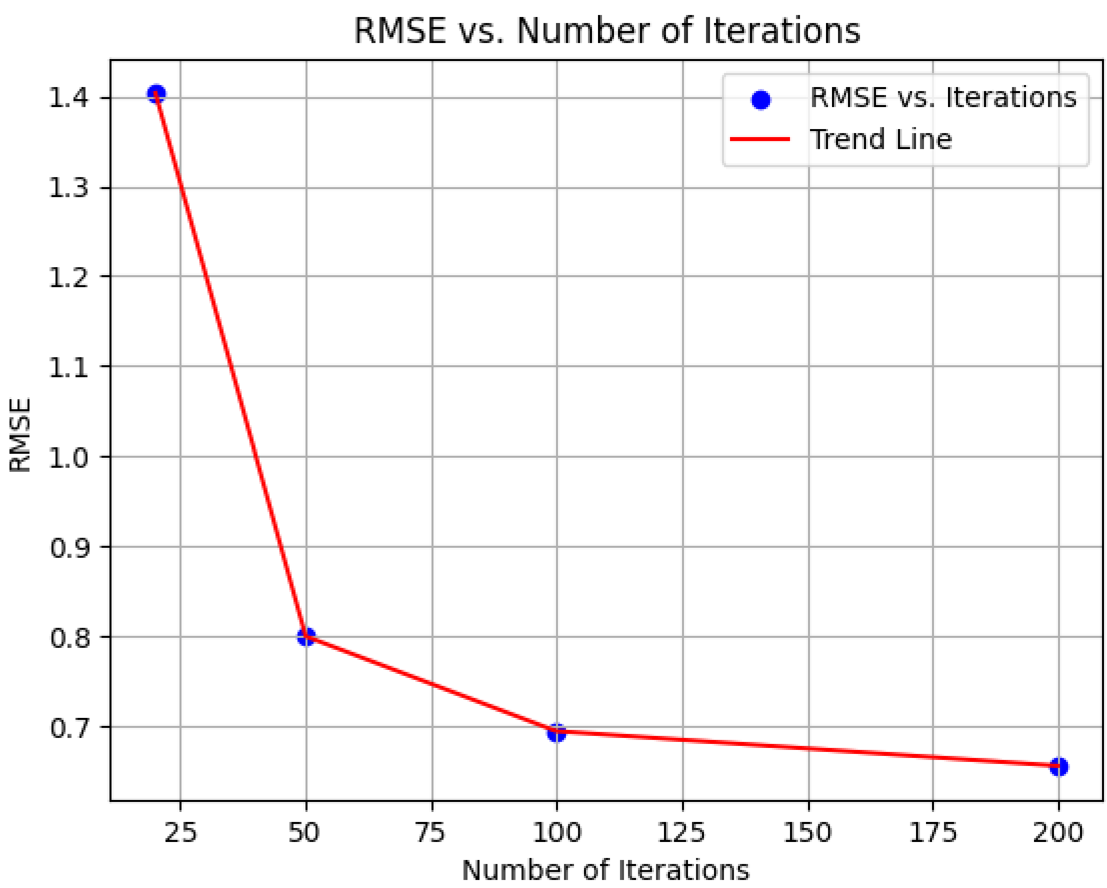
```
# Calculate the average MAE for original predictions and rounded predictions
avg_mae_original = np.mean(mae_original)
avg_mae_rounded_bounded = np.mean(mae_values_rounded_bounded)

print(f"Average MAE without rounding and bounding: {avg_mae_original:.4f}")
print(f"Average MAE with rounding and bounding: {avg_mae_rounded_bounded:.4f}")
```

```
Average MAE without rounding and bounding: 0.5097
Average MAE with rounding and bounding: 0.4388
```

We can observe the Mean Absolute Error with rounding and bounding applied to the predictions is lower than the original prediction, sugesting this is a good improvement to apply.

3)

```
iteration_settings = [20, 50, 100, 200]

rmse_results = []

for iterations in iteration_settings:
    aux_rmse_results = []
    for random_state in random_states:
        mlp = MLPRegressor(hidden_layer_sizes=(10, 10), activation='relu', max_iter=iterations, validation_fraction=0.2, random_state=random_state)

        mlp.fit(X_train, y_train)

        y_pred = mlp.predict(X_test)

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

    rmse_results.append(np.mean(aux_rmse_results))


plt.scatter(iteration_settings, rmse_results, marker='o', color='b', label='RMSE vs. Iterations')
plt.plot(iteration_settings, rmse_results, linestyle='-', color='r', label='Trend Line')
plt.xlabel('Number of Iterations')
plt.ylabel('RMSE')
plt.title('RMSE vs. Number of Iterations')
plt.grid(True)
plt.legend()
plt.show()
```



4) The results show a clear trend as the number of iterations increases, the RMSE generally decreases, indicating better model performance.
Early stopping helps prevent overfitting by stopping training when the validation error starts to increase, preventing the model from over-optimizing on the training data, leading to better generalization. It can also save trining time which is most relevant in bigger models.
On the other hand, early stopping can worsen performance when a model stops learning before it has converged to the optimal solution (global minimum of the loss function), particularly relevant if the chosen number of iterations is too small, leading to underfitting, where the model lacks the complexity to capture the underlying patterns in the data effectively. If the validation set is not representative of the test data or is too small, early stopping may also stop training prematurely or continue training unnecessarily, leading to suboptimal performance.

END