**I. Pen-and-paper**



W = (XT.X + YI)^-1 . XT.Z

|  |
| --- |
| [[24] |
| [20] |
| [10] |
| [13] |
| [12]] |

1 , 0.8 , 0.8\*\*2 , 0.8\*\*3

1 , 1 , 1 , 1

X = 1 , 1.2 , 1.2\*\*2 , 1.2\*\*3 Z =

1 , 1.4 , 1.4\*\*2 , 1.4\*\*3

1 , 1.6 , 1.6\*\*2 , 1.6\*\*3

|  |
| --- |
| [[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]] |

X^T =

|  |
| --- |
| [[ 5. 6. 7.6 10.08 ] |
| [ 6. 7.6 10.08 13.8784 ] |
| [ 7.6 10.08 13.8784 19.68 ] |
| [10.08 13.8784 19.68 28.55488]] |

X^T.X =

|  |
| --- |
| [[ 7. 6. 7.6 10.08 ] |
| [ 6. 9.6 10.08 13.8784 ] |
| [ 7.6 10.08 15.8784 19.68 ] |
| [10.08 13.8784 19.68 30.55488]] |

X^T.X + 2\*I =

|  |
| --- |
| [[ 0.34168753 -0.1214259 -0.07490231 -0.00932537] |
| [-0.1214259 0.3892078 -0.09667718 -0.07445624] |
| [-0.07490231 -0.09667718 0.37257788 -0.17135047] |
| [-0.00932537 -0.07445624 -0.17135047 0.17998796]] |

(X^T.X + 2\*I)^-1 =

|  |
| --- |
| [[ 0.19183474 0.13603395 0.07200288 -0.00070608 -0.08254055] |
| [ 0.08994535 0.09664848 0.07774793 0.02966982 -0.05115977] |
| [-0.00152564 0.02964793 0.04950363 0.04981662 0.02236208] |
| [-0.08640083 -0.07514413 -0.03439835 0.04447593 0.17011812]] |

(X^T.X + 2\*I)^-1.X^T =

|  |
| --- |
| [[ 7.0450759 ] |
| [ 4.64092765] |
| [ 1.96734046] |
| [-1.30088142]] |

(X^T.X + 2\*I)^-1.X^T.Z = = W

1. Z^i = W0 + W1\*Zi + W2\*Zi^2 + W3\*Zi^3

X = 0.8 | 1 | 1.2 | 1.4 | 1.6

Z = 24 | 20 | 10 | 13 | 12

~Z^ = 11.35 | 12.35 | 13.20 | 13.83 | 14.18

~Dif^2 = 160 | 58.48 | 10.24 | 0.69 | 4.75

RMSE = SQRT( SUM( Dif^2 ) / 5 ) ~ 6.843294892

1. X 🡪 NET1 🡪 A1 🡪 NET2 🡪 A2 🡪 L

X = A0 = [[0.8],

[ 1 ],

[1.2]]

W1 = [[1,1,1], W2 = [[1,1], B1 = [[1], B2 = [[1],

[1,1,1]] [1,1], [1]] [1],

[1,1]] [1,]]

NET1 = W1.X + B1 = [[4], A1i = e^(0.1\*NET1i) = [[1.491824698],

[4]] [1.491824698]]

NET2 = W2.A1 + B2 = [[3.98365], A2i = e^(0.1\*NET2i) = [[1.489387557],

[3.98365], [1.489387557],

[3.98365]] [1.489387557]]

dL/dA2 = A2 – Z

dAi/dNETi = 0.1\*e^(0.1NETi)

dNETi/dWi = A(i-1)

dNETi/dBi = 1

dNETi/dA(i-1) = Wi

|  |
| --- |
| [[-3.352702608], |
| [-2.756947585], |
| [-1.267560028]] |

δ2 = dL/dA2.dA2/dNET2 = ( A2 – Z ).0.1e^(0.1NET2) =

|  |
| --- |
| [[-0.856409118], |
| [-0.856409118]] |

δ1 = dNETi/dA1\*δ2.dA1/dNET1 = W2^T\*δ2.0.1e^(0.1NET1) =

|  |  |  |
| --- | --- | --- |
| [[-0.685127295, | -0.856409118, | -1.03], |
| [-0.685127295, | -0.856409118, | -1.03]] |

dL/dW1 = δ1\*A0 =

|  |  |  |
| --- | --- | --- |
| [[1.068512729, | 1.085640912, | 1.103], |
| [1.068512729, | 1.085640912, | 1.103]] |

W1 = W1 – 0.1.dL/dW1 =

|  |
| --- |
| [[1.856409118], |
| [1.856409118]] |

B1 = B1 – 0.1. δ1 =

|  |  |
| --- | --- |
| [[-5.001644554, | -5.001644554], |
| [-4.112882497, | -4.112882497], |
| [-1.890977355, | -1.890977355]] |

dL/dW2 = δ2\*A1 =

|  |  |
| --- | --- |
| [[1.500164455, | 1.500164455], |
| [1.41128825, | 1.41128825], |
| [1.189097735, | 1.189097735]] |

W2 = W2 – 0.1.dL/dW2 =

|  |
| --- |
| [[1.335270261], |
| [1.275694758], |
| [1.126756003]] |

B2 = B2 – 0.1. δ2 =

**II. Programming and critical analysis**

1. LR: 0.162829976437694

MPL1: 0.06804140737968428

MPL2: 0.0978071820387748

Chart, bar chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

1. M PL1\_iterations: 452

MPL2\_iterations: 77

**III. APPENDIX**

4)

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from scipy import stats

from scipy.io.arff import loadarff

from sklearn.linear\_model import Ridge

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import mean\_absolute\_error

from sklearn.neural\_network import MLPRegressor

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix

from sklearn.model\_selection import cross\_val\_score, StratifiedKFold

data = loadarff('kin8nm.arff')

df = pd.DataFrame(data[0])

X = df.drop('y', axis=1)

y = df['y']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, train\_size=0.7,random\_state=0)

LR = Ridge(alpha=0.1)

MPL1 = MLPRegressor(hidden\_layer\_sizes=(10,10),activation='tanh',max\_iter=500,random\_state=0,early\_stopping=True)

MPL2 = MLPRegressor(hidden\_layer\_sizes=(10,10),activation='tanh',max\_iter=500,random\_state=0,early\_stopping=False)

LR.fit(X\_train,y\_train)

MPL1.fit(X\_train,y\_train)

MPL2.fit(X\_train,y\_train)

y\_pred1 = LR.predict(X\_test)

y\_pred2 = MPL1.predict(X\_test)

y\_pred3 = MPL2.predict(X\_test)

print("LR:",mean\_absolute\_error(y\_test,y\_pred1))

print("MPL1:",mean\_absolute\_error(y\_test,y\_pred2))

print("MPL2:",mean\_absolute\_error(y\_test,y\_pred3))

5)

plt.boxplot([abs(y\_test.values - y\_pred1), abs(y\_test.values - y\_pred2), abs(y\_test.values - y\_pred3)])

plt.title('Boxplot')

plt.show()

plt.hist([abs(y\_test.values - y\_pred1), abs(y\_test.values - y\_pred2), abs(y\_test.values - y\_pred3)], label = ["LR", "MLP1", "MLP2"])

plt.title('Histogram')

plt.legend()

plt.show()

6)

print("MPL1\_iterations:", MPL1.n\_iter\_)

print("MPL2\_iterations:", MPL2.n\_iter\_)

**END**