**I. Pen-and-paper**



=



𝑧̂



Forward Propagation:

;

;

;

;

;

;

Back Propagation:

=

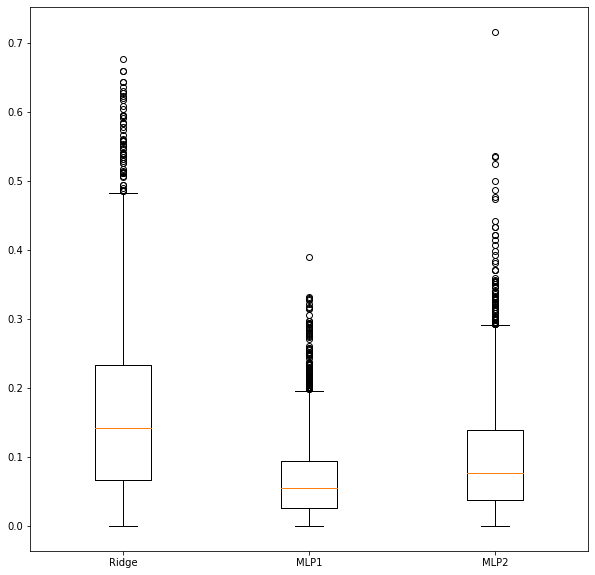
**II. Programming and critical analysis**

1. MAE(Ridge): 0.162829976437694

MAE(MPL1): 0.0680414073796843

MAE(MPL2): 0.0978071820387748

1. Chart, bar chart, histogram

   Description automatically generated
2. MLP1 iterations: 452

MLP2 iterations: 77

1. The regressor with early stopping will separate 10% of the input data for validation. This validation data will decide when the convergence algorithm should stop (in this case, it stops when there are *10* iterations with a validation score less than *1e-4*, default values of *n\_iter\_no\_change* and *tol* attributes respectively). The regressor without early stopping only uses the training loss on the entire input data for the stopping criterion. For this reason, early stopping regressor may have more iterations than the non-early stopping regressor, as in this case. Additionally, we can notice that MLP1’s MAE is smaller than MLP2’s MAE, and this means that the goal of decrease overfitting with early stopping was achieved.

**III. APPENDIX**

import pandas as pd

from scipy.io.arff import loadarff

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

from sklearn import metrics

import matplotlib.pyplot as plt

from sklearn.linear\_model import Ridge

from sklearn.neural\_network import MLPRegressor

data = loadarff('kin8nm.arff')

df = pd.DataFrame(data[0])

target = df['y']

df = df.drop('y', axis=1)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(df.values, target, test\_size = 0.3,\ random\_state=0)

rr = Ridge(alpha=0.1)

mlp1 = MLPRegressor(hidden\_layer\_sizes=(10,10), activation='tanh', random\_state=0,\ max\_iter=500, early\_stopping=True)

mlp2 = MLPRegressor(hidden\_layer\_sizes=(10,10), activation='tanh', random\_state=0,\ max\_iter=500, early\_stopping=False)

#4

rr.fit(x\_train, y\_train)

mlp1.fit(x\_train, y\_train)

mlp2.fit(x\_train, y\_train)

y\_pred\_rr = rr.predict(x\_test)

y\_pred\_mlp1 = mlp1.predict(x\_test)

y\_pred\_mlp2 = mlp2.predict(x\_test)

print("MAE(Ridge): ", metrics.mean\_absolute\_error(y\_test, y\_pred\_rr))

print("MAE(MPL1): ", metrics.mean\_absolute\_error(y\_test, y\_pred\_mlp1))

print("MAE(MPL2): ", metrics.mean\_absolute\_error(y\_test, y\_pred\_mlp2))

#5.1

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

boxplot = plt.boxplot([abs(y\_test - y\_pred\_rr), abs(y\_test - y\_pred\_mlp1), \

abs(y\_test - y\_pred\_mlp2)], labels=['Ridge', 'MLP1', 'MLP2'])

#5.2

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

hist = plt.hist([abs(y\_test - y\_pred\_rr), abs(y\_test - y\_pred\_mlp1), \

abs(y\_test - y\_pred\_mlp2)])

plt.legend(labels=['Ridge', 'MLP1', 'MLP2'])

# 6)

print("MLP1 iterations: ", mlp1.n\_iter\_)

print("MLP2 iterations: ", mlp2.n\_iter\_)

**END**