Aprendizagem 2022/23

Homework III

Deadline 28/10/2022 23:59 via Fenix as PDF

I. Pen-and-paper [12v]

1)

X=

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

X^T =

	A_1	A_2	A_3	A_4	A_5
1	1	1	1	1	1
2	0.8	1	1.2	1.4	1.6
3	0.64	1	1.44	1.96	2.56
4	0.512	1	1.728	2.744	4.096

Z =

24
20
10
13
12

$$w = (X^T.X + \lambda I)^{-1}.X^T.Z$$

$X^T.X$ =

	C_1	C_2	C_3	C_4
1	5	6	7.6	10.08
2	6	7.6	10.08	13.8784
3	7.6	10.08	13.8784	19.68
4	10.08	13.8784	19.68	28.55488

$X^T.X$ + 2I=

	C_1	C_2	C_3	C_4
1	7	6	7.6	10.08
2	6	9.6	10.08	13.8784
3	7.6	10.08	15.8784	19.68
4	10.08	13.8784	19.68	30.55488

$(X^T.X \ + \ 2I)^{-1} =$

	B_1	B_2	B_3	B_4
1	0.34168752983722529376	-0.12142590407576881366	-0.074902305488221549419	-0.0093253732832817855968
2	-0.12142590407576881366	0.38920780478603230441	-0.096677178802746364369	-0.074456244175093230302
3	-0.074902305488221549434	-0.096677178802746364405	0.37257788488219200307	-0.17135046764589585224
4	-0.0093253732832817855828	-0.074456244175093230283	-0.17135046764589585229	0.17998795953793059107

$(X^T.X \ + \ 2I)^{-1}.X^T$ =

	C_1	C_2	C_3	C_4	C_5
1	0.19183473994310818608	0.136033946989953155	0.07200288000975277152	-0.00070607891509049004	-0.08254054770217415536
2	0.08994534830165162072	0.09664847773242389	0.07774793425695407288	0.02966981815483769384	-0.05115977029432972264
3	-0.00152564164051442808	0.02964793294532825	0.04950362608673128648	0.04981661533669168104	0.02236207824820643336
4	-0.0864008326333092474	-0.075144125566340265	-0.0343983456219396058	0.044475929257713399	0.1701181211304394182

$(X^T.X \ + \ 2I)^{-1}.X^T.Z$ = w =

	C_1
1	7.04507590020892093
2	4.640927648938590735
3	1.967340458757000628
4	-1.30088141683007606

f(x) = 7.04 + 4.64x + 1.97$x^2 - 1.3x^3$

2)

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

$$= \sqrt{\frac{1}{5}[(24 \ - \ 11.35)^2 + (20 \ - \ 12.35)^2 + (10 \ - \ 13.2)^2 + (13 \ - \ 13.83)^2 + (12 \ - \ 14.18)^2]} \ = \ 6.84$$

3)

$w_1 = [{}^1_1]$ $b_1 = [{}^1_1]$

$w_2 = [1\ 1]$ $b_2 = [1]$

$f(x) = e^{0.1x}$

$$\delta^{[2]} = \frac{dE}{dx^{[2]}} \circ \frac{dx^{[2]}}{dz^{[2]}} \mid \frac{dE}{dx^{[2]}} = x^{[2]} - t \mid \frac{dx^{[k]}}{dz^{[k]}} = 0.1e^{0.1z^{[k]}}$$

$$\delta^{[1]} = \frac{dz^{[2]}}{dx^{[1]}}. \delta^{[2]} \circ \frac{dx^{[1]}}{dz^{[1]}} \mid \frac{dz^{[k]}}{dx^{[k-1]}} = w^{[k]^T}$$

for observation x1 = 0.8:

$z_1 = w_1x_1 + b_1 = [{}^{1.8}_{1.8}]$ $v_1 = f(z_1) = [{}^{1.197}_{1.197}]$

$z_2 = w_2v_1 + b_2 = 3.394$ $v_2 = f(z_2) = 1.404$

$\delta^{[2]}$ = $(v_2 - t_1). 0. 1. e^{0.1z_2}$ = -3.173

$\delta^{[1]} = w^{[2]^T}. \delta^{[2]}. 0. 1. e^{0.1z_1} = [{}^{-0.380}_{-0.380}]$

for observation x2 = 1:

$z_1 = w_1x_2 + b_1 = [{}^{2}_{2}]$ $v_1 = f(z_1) = [{}^{1.221}_{1.221}]$

$z_2 = w_2v_1 + b_2 = 3.443$ $v_2 = f(z_2) = 1.411$

$\delta^{[2]}$ = $(v_2 - t_2). 0. 1. e^{0.1z_2}$ = -2.623

$\delta^{[1]} = w^{[2]^T}. \delta^{[2]}. 0. 1. e^{0.1z_1} = [{}^{-0.320}_{-0.320}]$

for observation x3 = 1.2:

$z_1 = w_1 x_3 \; + \; b_1 = \; [\begin{smallmatrix} 2.2 \\ 2.2 \end{smallmatrix}] \quad v_1 = f(z_1) \; = \; [\begin{smallmatrix} 1.246 \\ 1.246 \end{smallmatrix}]$

$z_2 = \; w_2 v_1 \; + \; b_2 = 3.492 \quad v_2 = f(z_2) \; = \; 1.418$

$\delta^{[2]}$ = $(v_2 - \; t_3). 0. 1. e^{0.1z_2}$ =-1.217

$\delta^{[1]} = w^{[2]^T} . \delta^{[2]} . 0. 1. e^{0.1z_1} = \; [\begin{smallmatrix} -0.152 \\ -0.152 \end{smallmatrix}]$

atualizações:

$w^1_{new} = \mathbf{W}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[1]}} = \; w_1 - \; n(\delta^1_1 x^0_1 + \delta^1_2 x^0_2 + \delta^1_3 x^0_3) \; = [\begin{smallmatrix} 1.081 \\ 1.081 \end{smallmatrix}]$

$w^2_{new} = \mathbf{W}^{[2]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[2]}} = w_2 - \; n(\delta^2_1 x^0_1 + \delta^2_2 x^0_2 + \delta^2_3 x^0_3) \; = \; [1.852 \;\; 1.852]$

$b^1_{new} = \; b_1 - \; n(\delta^1_1 + \delta^1_2 + \delta^1_3) \; = [\begin{smallmatrix} 1.085 \\ 1.085 \end{smallmatrix}]$

$b^2_{new} = \; b_1 - \; n(\delta^2_1 + \delta^2_2 + \delta^2_3) \; = \; 1.701$

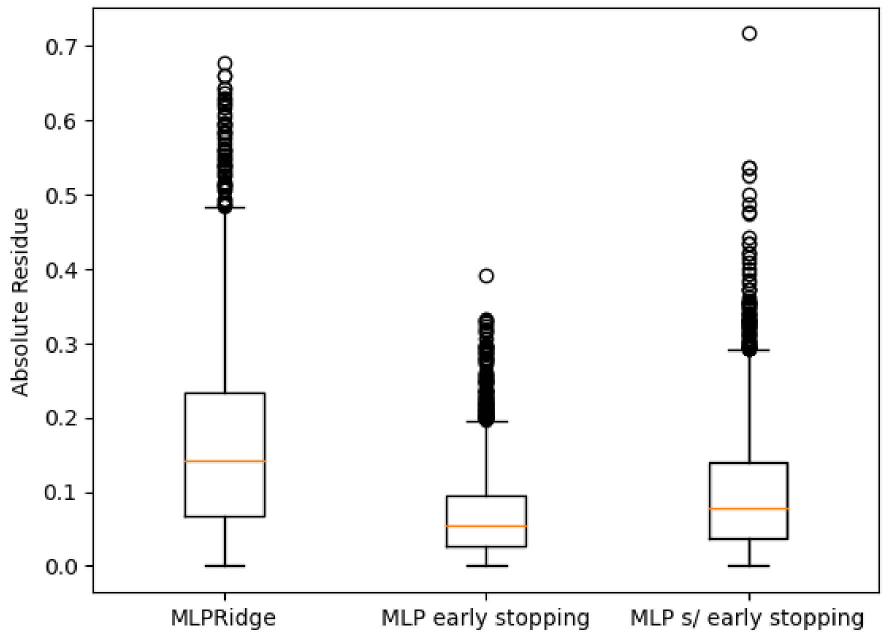
4)

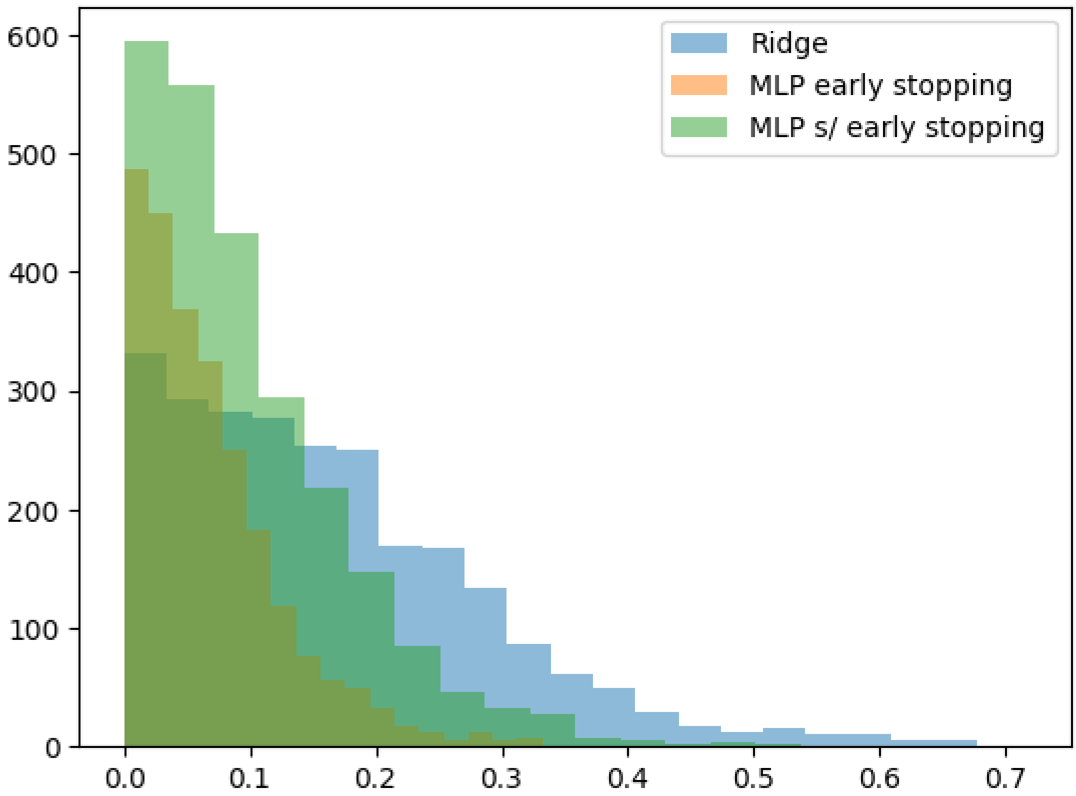
```
# Import Wall
import pandas as pd
import numpy as np
from sklearn import metrics, datasets, tree
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt
from scipy.io.arff import loadarff
import seaborn as sns
from scipy import stats
from sklearn.model_selection import train_test_split


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

# split data into training and testing sets
X = df.drop('y', axis=1).values
y = df['y'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0)

# linear regression with Ridge regularization term of 0.1 and random_state=0
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=0.1, random_state=0)
ridge.fit(X_train, y_train)
ridge_pred = ridge.predict(X_test)
print("Ridge Regression")
print("Mean Absolute Error: ", metrics.mean_absolute_error(y_test, ridge_pred))

# MLPRegressor with 10 hidden layers and hyperbolic tangent activation function
with early stopping with a maximum of 500 iterations
from sklearn.neural_network import MLPRegressor
mlp = MLPRegressor(hidden_layer_sizes=(10,10), activation='tanh', max_iter=500,
early_stopping=True,random_state=0)
```

```python
mlp.fit(X_train, y_train)
mlp_pred = mlp.predict(X_test)
print("MLPRegressor with early stopping")
print("Mean Absolute Error: ", metrics.mean_absolute_error(y_test, mlp_pred))

# MLPRegressor with 10 hidden layers and hyperbolic tangent activation function
mlp2 = MLPRegressor(hidden_layer_sizes=(10,10), activation='tanh', max_iter=500,
random_state=0)
mlp2.fit(X_train, y_train)
mlp2_pred = mlp2.predict(X_test)
print("MLPRegressor with no early stopping")
print("Mean Absolute Error: ", metrics.mean_absolute_error(y_test, mlp2_pred))

#b)
#plot the residue in absolute value using boxplot
plt.boxplot([abs(y_test - ridge_pred), abs(y_test - mlp_pred), abs(y_test -
mlp2_pred)])
plt.xticks([1, 2, 3], ['MLPRidge', 'MLP early stopping', 'MLP s/ early stopping'])
plt.ylabel('Absolute Residue')
plt.show()

#plot the residue in absolute value using histograms
plt.hist(abs(y_test - ridge_pred), bins=20, alpha=0.5, label='Ridge')
plt.hist(abs(y_test - mlp_pred), bins=20, alpha=0.5, label='MLP early stopping')
plt.hist(abs(y_test - mlp2_pred), bins=20, alpha=0.5, label='MLP s/ early
stopping')
plt.legend(loc='upper right')
plt.show()



#c)
#how many iterations did it take to converge?
print("Iterations for MLP to converge: ", mlp.n_iter_)
print("Iterations for MLP2 to converge: ", mlp2.n_iter_)
```

```
Ridge Regression
Mean Absolute Error:  0.162829976437694
MLPRegressor with early stopping
Mean Absolute Error:  0.0680414073796843
MLPRegressor with no early stopping
Mean Absolute Error:  0.0978071820387748
```

5)





6)

```
Iterations for MLP with early stopping to converge:  452
Iterations for MLP2 without early stopping to converge:  77
```

7)

The reason underlying the observed performance differences between the MLPs is that the MLP with early stopping stops training when the validation score does not improve by at least tol for two consecutive epochs. This means that the model could stop training before it has converged.