Aprendizagem 2022/23

Homework III

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

I. Pen-and-paper [12v]

Consider the problem of learning a regression model from 5 univariate observations $\big((0.8),(1),(1.2),(1.4),(1.6)\big)$ with targets $(24,20,10,13,12)$.

1) [4.5v] Consider the basis function, $\phi_j(x) = x^j$, for performing a 3-order polynomial regression,

$$\hat{z}(x,\mathbf{w}) = \sum_{j=0}^{3} w_j \phi_j(x) = w_0 + w_1 x + w_2 x^2 + +w_3 x^3 \,.$$

Learn the Ridge regression (l_2 regularization) on the transformed data space using the closed form solution with $\lambda = 1$.

Hint: use `numpy` matrix operations (e.g., `linalg.pinv` for inverse) to validate your calculus.

Let us represent the design matrix Φ by computing $x,\ \ x^2$ and x^3

$$\Phi = \begin{pmatrix} 1 & 0.8 & 0.64 & 0.512 \\ 1 & 1 & 1 & 1 \\ 1 & 1.2 & 1.44 & 1.728 \\ 1 & 1.4 & 1.96 & 2.744 \\ 1 & 1.6 & 2.56 & 4.096 \end{pmatrix}$$

Using the targets and the regularization, the regression can be learnt using:

$$\mathbf{w} = (\Phi^T \cdot \Phi \ + \ \lambda \cdot I)^{-1} \cdot \Phi T \cdot \mathbf{t}$$

Applying the above formula, the polynomial regression is: $y(\mathbf{x}) \approx 7.045 + 4.64x_1 + 1.967x_2 - 1.30x_3$

2) [1.5v] Compute the training RMSE for the learnt regression.

$$\hat{\boldsymbol{t}} = \Phi\mathbf{w} = \begin{pmatrix} 11.35 \\ 12.35 \\ 13.20 \\ 13.83 \\ 14.18 \end{pmatrix}, \text{RMSE} = 6.84$$

3) [6v] Consider a multi-layer perceptron characterized by one hidden layer with 2 nodes. Using the activation function $f(x) = e^{0.1x}$ on all units, all weights initialized as 1 (including biases), and the squared error loss, perform one batch gradient descent update (with learning rate $\eta = 0.1$) for the first three observations (0.8), (1) and (1.2).

$$\mathbf{W}^{[1]} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},\ \ \mathbf{b}^{[1]} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{W}^{[2]} = \begin{pmatrix} 1 & 1 \end{pmatrix},\ \ \mathbf{b}^{[2]} = (1)$$

Let us derive the activation function: $\frac{\partial f(x)}{\partial x} = 0.1e^{0.1x}$

Let us now find the elementary rules:

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

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

$$\frac{\partial \mathbf{x}^{[i]}(\mathbf{z}^{[i]})}{\partial \mathbf{z}^{[i]}} = 0.1e^{0.1\mathbf{z}^{[i]}}$$

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

Aprendizagem 2022/23

Homework III

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

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

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

We can then infer:

$$\delta^{[2]} = \frac{\partial E}{\partial \mathbf{x}^{[2]}} \circ \frac{\partial \mathbf{x}^{[2]}}{\partial \mathbf{z}^{[2]}} = \left(\mathbf{x}^{[2]} - t\right) \circ 0.1e^{0.1\mathbf{z}^{[2]}}$$

$$\delta^{[1]} = \left(\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{x}^{[1]}}\right)^T \cdot \delta^{[2]} \circ \frac{\partial \mathbf{x}^{[1]}}{\partial \mathbf{z}^{[1]}} = \left(\mathbf{W}^{[2]}\right)^T \delta^{[2]} \circ 0.1e^{0.1\mathbf{z}^{[1]}}$$

Let us propagate the values from the three observations:

$$\mathbf{z}^{[1](1)} = \begin{pmatrix}1.8\\1.8\end{pmatrix},\quad \mathbf{x}^{[1](1)} = e^{0.1\mathbf{z}^{[1](1)}} = \begin{pmatrix}1.197\\1.197\end{pmatrix},\quad \mathbf{z}^{[2](1)} = (3.394),\quad \mathbf{x}^{[2](1)} = e^{0.1\mathbf{z}^{[2](1)}} = (1.404),$$

$$\mathbf{z}^{[1](2)} = \begin{pmatrix}2\\2\end{pmatrix},\quad \mathbf{x}^{[1](2)} = e^{0.1\mathbf{z}^{[1](2)}} = \begin{pmatrix}1.22\\1.22\end{pmatrix},\quad \mathbf{z}^{[2](2)} = (3.442),\quad \mathbf{x}^{[2](2)} = e^{0.1\mathbf{z}^{[2](2)}} = (1.411),$$

$$\mathbf{z}^{[1](3)} = \begin{pmatrix}2.2\\2.2\end{pmatrix},\quad \mathbf{x}^{[1](3)} = e^{0.1\mathbf{z}^{[1](3)}} = \begin{pmatrix}1.246\\1.246\end{pmatrix},\quad \mathbf{z}^{[2](3)} = (3.492),\quad \mathbf{x}^{[2](3)} = e^{0.1\mathbf{z}^{[2](3)}} = (1.418)$$

Let us then compute the deltas:

For first observation: $\delta^{[2](1)} = (-3.173),\quad \delta^{[1](1)} = \begin{pmatrix}-0.380\\-0.380\end{pmatrix}$

For second observation: $\delta^{[2](2)} = (-2.623),\quad \delta^{[1](2)} = \begin{pmatrix}-0.320\\-0.320\end{pmatrix}$

For third observation: $\delta^{[2](3)} = (-2.068),\quad \delta^{[1](3)} = \begin{pmatrix}-0.258\\-0.258\end{pmatrix}$

Finally, we can go to the last phase and perform the updates:

$$\frac{\partial E}{\partial \mathbf{W}^{[1]}} = \delta^{[1](1)} \frac{\partial \mathbf{z}^{[1](1)}}{\partial \mathbf{W}^{[1]}} + \delta^{[1](2)} \frac{\partial \mathbf{z}^{[1](2)}}{\partial \mathbf{W}^{[1]}} + \delta^{[1](3)} \frac{\partial \mathbf{z}^{[1](3)}}{\partial \mathbf{W}^{[1]}} = \delta^{[1](1)}\left(\mathbf{x}^{[0](1)}\right)^T + \delta^{[1](2)}\left(\mathbf{x}^{[0](2)}\right)^T + \delta^{[1](3)}\left(\mathbf{x}^{[0](3)}\right)^T$$

$$= \begin{pmatrix}-0.380\\-0.380\end{pmatrix} 0.8 + \begin{pmatrix}-0.320\\-0.320\end{pmatrix} 1 + \begin{pmatrix}-0.258\\-0.258\end{pmatrix} 1.2 = \begin{pmatrix}-0.933\\-0.933\end{pmatrix}$$

$$\mathbf{W}^{[1]} = \mathbf{W}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[1]}} = \begin{pmatrix}1\\1\end{pmatrix} - 0.1\begin{pmatrix}-0.933\\-0.933\end{pmatrix} = \begin{pmatrix}1.093\\1.093\end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[1]}} = \delta^{[1](1)} \frac{\partial \mathbf{z}^{[1](1)^T}}{\partial \mathbf{b}^{[1]}} + \delta^{[1](2)} \frac{\partial \mathbf{z}^{[1](2)^T}}{\partial \mathbf{b}^{[1]}} + \delta^{[1](3)} \frac{\partial \mathbf{z}^{[1](3)^T}}{\partial \mathbf{b}^{[1]}} = \delta^{[1](1)} + \delta^{[1](2)} + \delta^{[1](3)} = \begin{pmatrix}-0.958\\-0.958\end{pmatrix}$$

$$\mathbf{b}^{[1]} = \mathbf{b}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{b}^{[1]}} = \begin{pmatrix}1\\1\end{pmatrix} - 0.1\begin{pmatrix}-0.958\\-0.958\end{pmatrix} = \begin{pmatrix}1.096\\1.096\end{pmatrix}$$

All that is left is to update the parameters for the output layer:

$$\frac{\partial E}{\partial \boldsymbol{W}^{[2]}} = \delta^{[2](1)} \frac{\partial \mathbf{z}^{[2](1)}}{\partial \boldsymbol{W}^{[2]}} + \delta^{[2](2)} \frac{\partial \mathbf{z}^{[2](2)}}{\partial \boldsymbol{W}^{[2]}} + \delta^{[2](3)} \frac{\partial \mathbf{z}^{[2](3)}}{\partial \boldsymbol{W}^{[2]}} = \delta^{[2](1)}\left(\mathbf{x}^{[1](1)}\right)^T + \delta^{[2](2)}\left(\mathbf{x}^{[1](2)}\right)^T + \delta^{[2](3)}\left(\mathbf{x}^{[1](3)}\right)^T$$

$$= (-3.173)\begin{pmatrix}1.197\\1.197\end{pmatrix}^T + (-2.623)\begin{pmatrix}1.22\\1.22\end{pmatrix}^T + (-2.068)\begin{pmatrix}1.246\\1.246\end{pmatrix}^T = \begin{pmatrix}-9.574 & -9.574\end{pmatrix}$$

$$\mathbf{W}^{[2]} = \mathbf{W}^{[2]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[2]}} = \begin{pmatrix}1 & 1\end{pmatrix} - 0.1\begin{pmatrix}-9.574 & -9.574\end{pmatrix} = \begin{pmatrix}1.9574 & 1.9574\end{pmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}^{[2]}} = \delta^{[2](1)} \frac{\partial \mathbf{z}^{[2](1)^T}}{\partial \mathbf{b}^{[2]}} + \delta^{[2](2)} \frac{\partial \mathbf{z}^{[2](2)^T}}{\partial \mathbf{b}^{[2]}} + \delta^{[2](3)} \frac{\partial \mathbf{z}^{[2](3)^T}}{\partial \mathbf{b}^{[2]}} = \delta^{[2](1)} + \delta^{[2](2)} + \delta^{[2](3)} = (-7.863),$$

$$\mathbf{b}^{[2]} = \mathbf{b}^{[2]} - \eta \frac{\partial E}{\partial \mathbf{b}^{[2]}} = (1.7863)$$

Aprendizagem 2022/23

Homework III

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

II. Programming and critical analysis [8v]

Consider the following three regressors applied on `kin8nm.arff` data (available at the webpage):

- linear regression with Ridge regularization term of 0.1
- two MLPs – MLP_1 and MLP_2 – each with two hidden layers of size 10, hyperbolic tangent function as the activation function of all nodes, a maximum of 500 iterations, and a fixed seed (`random_state=0`). MLP_1 should be parameterized with early stopping while MLP_2 should not consider early stopping. Remaining parameters (e.g., loss function, batch size, regularization term, solver) should be set as default.

Using a 70-30 training-test split with a fixed seed (`random_state=0`):

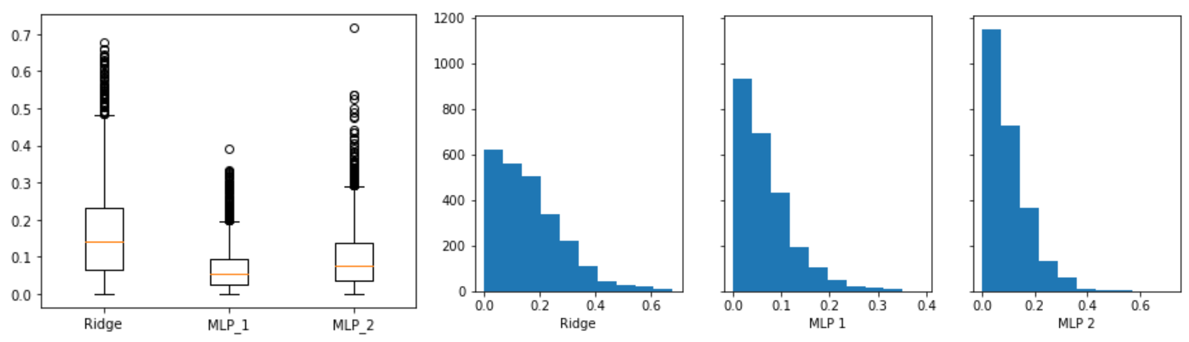
4) [4v] Compute the MAE of the three regressors: linear regression, MLP_1 and MLP_2.

MAE(RR)=0.1628, MAE(MLP_1) = 0.068, MAE(MLP_2) = 0.0978

```
from sklearn.linear_model import Ridge
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split

# 0. loading data
data = loadarff('kin8nm.arff')
df = pd.DataFrame(data[0])
df.dropna(inplace=True)
X, y = df.drop('y', axis=1), df['y']

# 1. learn the regressors
rr = Ridge(alpha=0.1)
mlp1 = MLPRegressor(hidden_layer_sizes=(10,10),activation='tanh',early_stopping=True, random_state=0,max_iter=500)
mlp2 = MLPRegressor(hidden_layer_sizes=(10,10),activation='tanh',early_stopping=False,random_state=0,max_iter=500)

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=0)

rr.fit(X_train, y_train)
mlp1.fit(X_train, y_train)
mlp2.fit(X_train, y_train)

# 2. collect the residues
y_rr, y_mlp1, y_mlp2 = rr.predict(X_test), mlp1.predict(X_test), mlp2.predict(X_test)
residues_rr = np.abs(np.array(y_rr)-np.array(y_test))
residues_mlp1 = np.abs(np.array(y_mlp1)-np.array(y_test))
residues_mlp2 = np.abs(np.array(y_mlp2)-np.array(y_test))
np.mean(residues_rr), np.mean(residues_mlp1), np.mean(residues_mlp2)
```

```
(0.162829976437694, 0.0680414073796843, 0.09780718203877478)
```

5) [1.5v] Plot the residues (in absolute value) using two visualizations: boxplots and histograms. *Hint*: consider using `boxplot` and `hist` functions from `matplotlib.pyplot` to this end



Aprendizagem 2022/23

Homework III

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

6) [1v] How many iterations were required for MLP_1 and MLP_2 to converge?

   MLP_1 = 452 iterations, MLP_2 = 77 iterations

7) [1.5v] What can be motivating the differences on the number of iterations? Hypothesize one underlying reason for the observed performance differences between the MLP regressors.

   Early stopping is oddly associated with a higher number of iterations to converge. *Why?* **1**) different training datasets due to the need to set aside validation data; **2**) loss assessed on validation set instead of training set; **3**) training with early stopping terminates when validation score is not improving by at least 10 consecutive epochs. Without early stopping, termination is based on training data, which can arbitrarily stop in the presence of a local maximum.

   Early stopping favors performance. Understandable, as it prevents both *underfitting risks* (local maximum in training data) and *overfitting risks* (optimized weights for training data in detriment of unseen data).

END