Course: Numerical Analysis for Machine Learning

Prof. E. Miglio - Jume 16th 2022

Duration of the exam: 2.5 hours.

Exercise 1 We consider a database containing the temperature in four corners of a room as a function of time. Each row of the matrix A refers to a time and each column represents the temperature in a corner of the room.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv('https://openmv.net/file/room-temperature.csv')
labels = ['FrontLeft', 'FrontRight', 'BackLeft', 'BackRight']
A = data.to_numpy()[:,1:].astype(np.float64)
A -= 273.15 #from Kelvin to Celsius
```

1. How many samples and features are there in the dataset ?

2. Plot the values over time of the temperature in the four corners of the room. From the figure, what can you tell about the potential for dimensionality reduction of the dataset?

3. Perform PCA on the dataset by means of the SVD decomposition. Then, plot the trend of the fraction of the "explained variance": $\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{q} \sigma_i^2}$;

4. Print the four principal directions.

5. By using the principal directions computed above, perform dimensionality reduction using only the first k principal directions. Repeat the exercise for $k = 1, 2, 3$.

6. Plot the values over time of the temperature in the four corners of the room, obtained from the reconstructions of rank 1, 2 and 3 obtained above.

7. Comment on the results of points 3,4 and 6.

Exercise 2

Consider the Boston dataset contained in scikit-learn:

```
import numpy as np
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split

# Load the Boston housing dataset.
X_boston, y_boston = load_boston(return_X_y=True)

# Split into 60% training, 20% validation and 20% test.
X_boston_tr, X_rest, y_boston_tr, y_rest = \
  train_test_split(X_boston, y_boston, test_size=0.4, random_state=0)
X_boston_val, X_boston_te, y_boston_val, y_boston_te = \
  train_test_split(X_rest, y_rest, test_size=0.5, random_state=0)
```

1. Implement the analytical solution of ridge regression $(X^T X + \alpha I)w = X^T y$ using scipy.linalg.solve. Compute the solution on the training data. Make sure that the gradient at the solution is zero (up to machine precision).

2. Train the models for several possible values of alpha (`alphas = np.logspace(-3, 3, 20)`). Plot the mean squared error on the test set as a function of alpha. Use the validation data to find the best alpha and display it on the graph.

3. Consider the scaled version of the Boston dataset

```
from sklearn.preprocessing import StandardScaler

X = StandardScaler().fit_transform(X_boston)
y = y_boston - y_boston.mean()
y /= np.std(y_boston)
```

Write a function that computes the stochastic gradient of ridge regression

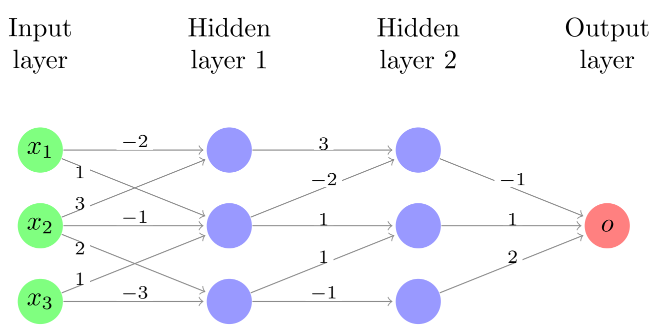
$$J(\mathbf{w}) = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \frac{\alpha}{2}\|\mathbf{w}\|^2. \tag{1}$$

Check that the mean of the stochastic gradient gives the gradient.

4. Write a function that implements stochastic gradient descent. Implement two rules for sampling the index: cyclic, and at random. Compare the convergence of both algorithms using the value of alpha found at point 2. What is the role of the step size?

Exercise 3

Consider the following network where on each edge (i, j) the value of $\frac{\partial y(j)}{\partial y(i)}$ is given; $y(k)$ denotes the activation of node k.



The output o is equal to 0.1 and the loss function is $L = -log(o)$. Compute the value of $\frac{\partial L}{\partial x_i}$ for each input x_i using the backpropagation method.