6ELEN018W - Tutorial 8 Exercises

Exercise 1

Implement the diabetes prediction example using `sklearn` from the lecture slides.

Make sure that you type the code (not copy and paste) and understand every single line of code. Ask your tutor if any questions or in doubt about any parts of the code.

You can also study the documentation for this dataset in `sklearn` in:

`https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes`

Exercise 2

Using `sklearn` implement a neural network which is able to predict house prices. Use similar steps with the code for the diabetes problem.

The dataset is also well known (similarly with the diabetes dataset) and it is described in:

`https://www.kaggle.com/datasets/camnugent/california-housing-prices`

- The dataset is also built-in `sklearn` and it can be loaded using:

```
housing = fetch_california_housing()
X = housing.data
y = housing.target
```

Exercise 3

Implement using `sklearn` a neural network to solve the XOR problem.

If you do not remember what is the XOR problem (also described in the lecture), you can find the truth table of this boolean function at:

`https://en.wikipedia.org/wiki/Exclusive_or`

Make sure that you scale the data in an appropriate range (see the lecture slides).

Use the hyperbolic tangent function `tanh` option as an activation function for training the neural network and the `lbfgs` solver by passing the appropriate parameters to the MLPRegressor constructor:

`https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#examples-using-sklearn-neural-network-mlpregressor`

Use one hidden layer with 2 nodes.

1. Experiment with different number of hidden nodes in 1 hidden layer and log in a file how many iterations are needed for them to solve the problem (use 1 up to 20 hidden nodes).

2. Plot a Matplotlib diagram (see the previous tutorials) the number of hidden nodes (in the x-axis) vs the number of iterations that solves the problem (values in the y-axis).

3. Discuss the results of the diagram, what can you tell about the number of hidden nodes required to solve the problem with 1-hidden layer?

4. Experiment with different numbers of hidden nodes in a network using 2 hidden layers and log in a file how many iterations are needed for them (use 1 up to 20 hidden nodes).

5. Plot a Matplotlib diagram with 2 lines. In the x axis the number of hidden nodes is used for both lines. In the y-axis the two lines should correspond to the number of iterations needed for a 1-hidden layer neural network and a 2-hidden layers neural network capable to solve the problem.

   Discuss the results of the plot. What can you say about the usage of a 1-hidden layer vs 2-hidden layers in this problem?