INF264 - Homework 4

Natacha Galmiche

1 Neural networks for a regression task using scikit-learn

In this section, we will use scikit-learn in order to manipulate simple feedforward neural networks. More specifically, we will first visualize and preprocess the diabetes dataset. Then, build different MLPs (Multi-Layer Perceptron) regressors that consist of a few dense layers, each model defined with a different set of hyper-parameters. Then, we will select the best model based on their performance on the diabetes dataset. Finally, we will evaluate its performance on unseen data and visualize its predictions compare to the expected values.

Note: There is no template for this exercise, but you are free to get inspiration from previous exercises / homework solutions.

1. Load the diabetes dataset using the `sklearn.datasets.load_diabetes` function. Alternatively, you can find the dataset following this link: `https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html`.
2. Visualize the dataset. You can use scatterplots for example.
3. Preprocess the data:
   (a) Select relevant features.
   (b) Split the data into a training, validation and test dataset.
   (c) Scale the data (you can use `MinMaxScaler` and/or `StandardScaler` from `sklearn.preprocessing`).
4. Build simple MLPs using sklearn:
   (a) Use the `MLPRegressor` class from `sklearn.neural_network`
   (b) Define different sets of hyperparameters. You can for example play with the `hidden_layer_sizes`, `momentum`, `solver` and `learning_rate_init` hyperparameters.
5. Train all your models using a loss function suitable for a regression task
6. Select the best model using an appropriate metric and KFold cross validation.
7. Evaluate the best model using the same metric. Can you really tell how good your model is performing based on the metric here?
8. Visualizing the performance of your model.:
   (a) Plot the model predictions next to the expected values, both for the train-validation dataset and the test dataset. You can use different colors to distinguish between the expected values and the predicted ones. Comment. Can you see if your model is doing great? Can you see which cases are hard to predict?
   (b) Plot the features/targets in a similar fashion than in question 2. More specifically, define multiple scatterplots, where the y-axis is always the target value and each x-axis is one of the selected features. The data is taken from the test dataset and the N best predictions are colored in green, the N worst in red and all the others in blue. Comment. Can you see which cases are hard to predict now?