Machine Learning Project 1: Overfitting, Underfitting and Metaparameters

1 Goal of the project and modalities

In the first part of the course, you learned about basic models, over/underfitting and metaparameters. In this project, you will use linear models to solve a regression task and select the right complexity for these models.

You are expected to handle out a **PDF** report of max. 4 pages, including plots. This report can be written in English or in French. On the implementation side, you will use Python 3 with the `scikit-learn` framework (`http://scikit-learn.org`), which provides many tools for machine learning and has a detailed documentation. Please check the instructions in `https://scikit-learn.org/stable/install.html` to install scikit-learn. **Your code should be submitted along with your report on Webcampus (by either providing a notebook or a .py file).**

Please, carefully respect the aforementioned instructions. Due to the high number of reports to evaluate, failure to comply with at least one of the previous instructions will result in a grade of 0/20 !

2 Training and testing datasets

The Diabetes dataset is widely known and used in machine learning. It contains information about patients in order to predict their progression of diabete. To learn more about this regression dataset, see `https://scikit-learn.org/1.5/datasets/toy_dataset.html#diabetes-dataset`. Use the function `load_diabetes()` to fetch the dataset (`https://scikit-learn.org/1.5/modules/generated/sklearn.datasets.load_diabetes.html`). The dataset can be split in a training set and a test set using the `train_test_split()` function of scikit-learn: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`.

3 Writing your own model

Once you have loaded the training and testing sets, you can train models. The first step of this project is to write your own model in Python. In order to do that, we ask you to carefully read the course material and to come up with an implementation of linear regression that will minimize the Mean Squared Error

(MSE) using an analytical solution (i.e. this first implementation should not optimize the MSE through an iterative process). Please describe in your report how you solved this first step, as well as the performance of your model.

Your first task is to implement a linear regression model (using the analytical solution) and to train it. Describe your implementation, show the weights that you obtain and comment the performance when predicting on the test set. In order to get a performance score, use the coefficient of determination R^2 (`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html`).

4 Lasso models from scikit-learn

More complex linear models than the one you wrote in the previous section exist. Scikit-learn provides an implementation of Lasso, a type of linear models that tries to minimizes the error AND the number of non-zero weights. Thanks to that, Lasso models are generally more interpretable, as less features are used. The scikit-learn implementation of Lasso can be found here: `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html`. The only metaparameter that you will have to deal with is alpha, representing the balance between the importance given to the error versus the number of non-zero weights.

Your second task is to train a Lasso model (with alpha = 0.5). Show the weights in your report (using the attribute `coef_` of your Lasso model, see `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html`) and comment on the performance on the test set. Compare the weights used by this Lasso model and its performance with those of your previous model. Can you say something about over/underfitting given these two models (the one from the first task and Lasso)?

5 Comparing models of increasing complexity

In this part of the project, you will try different models and compare them together.

Your third task is to train Lasso models with 100 alpha ranging from 0.1 to 1 (meaning that 100 models will be trained). Train each Lasso model and compute the training performance (coefficient of determination R^2 from the function `score()` of `scikit-learn`). Show a plot of the training and test scores for each alpha in your report and comment your results. Does the plot correspond to the theory? Can you spot over/underfitting cases in the plot?

6 Choose your model

Using the results from the previous section, you will now have to select your final Lasso model.

For this final task, choose the best value of alpha based on the corresponding plot of the third task. Justify your choice theoretically. Comment the model very shortly (in terms of over/underfitting, interpretability and training/test scores). Finally, compare this Lasso model to your own implementation from the first task (in terms of over/underfitting, interpretability and training/test scores).