Machine Learning Project 1: Overfitting, Underfitting and Metaparameters

1 Goal of the Project

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 complexity of these models. You are expected to handle out a report of max. 4 pages, including plots. **Additional pages will not be read.** 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 with your report on Webcampus.

2 Training and Test Datasets

The Diabetes dataset is widely known and used in machine learning. It contains patient information in order to predict their progression of diabete. To learn more about this regression dataset, see `https://scikit-learn.org/stable/datasets/index.html#diabetes-dataset`. Use the function called "load_diabetes" to fetch the dataset (`https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html`). The dataset can be split in a training set and a test set using the function train_test_split 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 test 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 ordinary least squares (OLS) in its analytical form (i.e. no iterations are needed to train the model).

The goal of this first step is to make you aware of what a model can be. Please describe in your report the code that you wrote for this first step, as well as the performance of your model.

Your first task is to implement an OLS model using the training set. Describe your implementation, show the weights that you obtain and comment on the performance when predicting 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) using the training set. 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 OLS model. Can you say something about over/underfitting given these two models (OLS and Lasso)?

5 Comparing Models of Increasing Complexity

While the training set allows you to train a model, the test set can be used to compare several models of increasing complexity.

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

Tip: plots can be made with the `plot` function from `matplotlib.pyplot`. Here is an example of use:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4])
y1 = x
y2 = x**2

plt.plot(x, y1, 'r')
plt.plot(x, y2, 'b')

plt.xlabel('x')
plt.ylabel('y')

plt.show()
```

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 OLS model from the first task (in terms of over/underfitting, interpretability and training/test scores).