Exercises Linear Regression

1 House Price Model

Fit a linear regression model to predict house prices with the data shown in the lecture:

age	area	prize (K EUR)
10	150	300
15	120	200
10	100	250

Use your model to predict the price of a house of age 12 years and area 120 square meters.

1. First you should try to build a model using `Numpy` commands only. To invert a matrix, you can use the function `np.linalg.inv`.
2. Now use the machine learning library `sklearn`. Import the class `LinearRegression` from the `linear_model` subpackage (see below). Check out the documentation (API) of the class, it contains a small example.

```python
# imports for this problem
import numpy as np
from sklearn.linear_model import LinearRegression
```

2 Polynomial Features and Ridge Regression

In this exercise we explore multivariate polynomial basis expansion using the class `PolynomialFeatures` in the package `sklearn.preprocessing` (read the docs).

1. Generate data using `sklearn.datasets.make_friedman2` (highly non-linear regression problem): Use the default parameter and create $n = 200$ samples. Use the function `train_test_split` in the package `sklearn.model_selection` to split the data in a **training** and a **validation** set (75% for training and 25% for validation).
2. Fit several models on the training data: first a ridge regression model and, then, ridge regression models where you apply the `PolynomialFeatures` class (with different degrees, ranging from 2 to 5) as a preprocessing step. All regression models should be instances of the class `Ridge` in `sklearn.linear_model` and should use the default parameters. In order to efficiently implement the preprocessing steps, check out the **hint** below.
3. Compare the models using the validation set to measure the mean square error (MSE) of both models. Which one performs better?
4. Now generate more data with `make_friedman2` (this is the **test data**) and evaluate the models with respect to the MSE on this data. Does the measured error coincide with the results on the validation set?

Hint

Sklearn offers so called `pipelines`, which allows you to combine several preprocessing steps and the final estimator into one object. Here is an example, using a `sklearn.preprocessing.StandardScaler` and the function `sklearn.pipeline.make_pipeline` to build the pipeline.

```
pipe = make_pipeline(StandardScaler(), Ridge())
```

The object `pipe` has a `fit` and `predict` function, and automatically applies scaling to the data. Note the `scaler` also transforms data passed to the predict function, but with parameters (mean, standard deviation) learned on the training data.