Linear Regression and Test-Driven Development

Applied Machine Learning in Engineering - Exercise 01

TU Berlin, Winter Term 2023/24

Prof. Dr.-Ing. Merten Stender – merten.stender@tu-berlin.de

This exercise will teach the `Python` implementation of basic linear regression applied to a small automotive engineering dataset. Furthermore, the test-driven software development paradigm is covered.

Linear Regression

Implement a function `lin_regress()` that solves a linear scalar regression problem and returns the model parameters θ_0 and θ_1.

(a) Use `numpy` to implement the normal form and solve for the model coefficients

(b) Test your implementation by passing the vectors `x=np.arange(0, 10, 1)` and a synthetic prediction vector `y_hat=x+0.1*np.random.randn(10)` to your function, receiving the coefficients, and plotting the results using a scatter plot, such as `plt.scatter(x, y_hat)`.

Use `plt.plot(x, theta_0+theta_1 * x, color='red')` to show the resulting fit.

Case study: Rolling resistance estimation

Estimate the effective rolling resistance factor of a car from measurements of vehicle speed v and engine power P_{engine}. The underlying equations for the wind's force F_{wind}, the rolling resistance force F_{roll}, and the resulting power P are given in the following

$$\begin{aligned} F_{\text{wind}} &= c_{\text{w}} A \frac{\rho_{\text{air}} v_{\text{rel}}^2}{2} \\ F_{\text{roll}} &= c_{\text{R}} M g \cos(\alpha) \\ P &= v \cdot F \end{aligned} \tag{1}$$

where the known parameters are $c_{\text{w}} = 0.4$, face area $A = 1.5\,\text{m}^2$, air density $\rho_{\text{air}} = 1.2\,\text{kg/m}^3$, gravity $g = 9.81\text{m/s}^2$ and the vehicle's mass $M = 2400\,\text{kg}$.

(a) Read the data file using `data = np.genfromtxt("driving_data.csv", delimiter=",")`, where the first column is the velocity (in m/s), and the second column carries the instantaneous engine power (in W).

(b) Re-formulate the problem such that the rolling resistance can be read from a linear fit to the existing data

(c) Report your estimate of the rolling resistance value c_{R} and check if your result is plausible.

Test-Driven Development

Red phase

Implement the R^2 error metric for computing the distance between a ground truth vector $\mathbf{y} \in \mathbb{R}^d$ and a predicted vector $\hat{\mathbf{y}} \in \mathbb{R}^d$. Use test-driven development for the implementation, i.e. first write tests and then implement the actual distance metrics.

(a) Define empty Python function `r2_dist(x, y)` in a Python file `dist_metrics.py`. Use typehints to ease readability of your code. Employ `numpy.ndarray` for representing `x` and `y`.

(b) Create a unittesting script `test_dist_metrics.py` and customize the template code provided as a separate file in `unittest_template.py`.

(c) Define a case object `Test_r2_dist` and write methods `test_exact_dist()`, `test_zero_dist()`, `test_dimensionality` `test_data_type()` for each class using the methods `assertAlmostEqual()`, `assertEqual()`, `assertRaises()`. Refer to `https://docs.python.org/3/library/unittest.html` for more details. Think of reasonable test cases that you want to check.

(d) Ensure that all tests fail (Python however should not raise any syntax error!), e.g. by returning a negative dummy value.

Green phase

(a) Implement the actual code for the distance metric using trivial indexing and looping over entries of the arrays `x` and `y`. Use only the operator `**2` for squaring a number and the Numpy methods `numpy.sum()` and `numpy.sqrt()`. Test that the numeric tests are passed.

(b) Implement a dimensionality check for the inputs `x` and `y`, and raise a `TypeError` if wrong types or wrong dimensions are supplied to the distance functions. Make use of `numpy.shape`, `numpy.ndim` and `type()` methods.

(c) Ensure that all tests are passed.

Refactoring phase

(a) Review your code and refactor. Discuss with your neighbor. Put in comments and docstrings.

(b) Check that all tests are still passed.

Evaluate

Compute the R^2 value of your fit. Potentially validate using the scikit-learn library.