Exercise 3: Non-parametric Regression

1. k-NN Regression

Besides for classification tasks the k-NN algorithm can be used for regression. The k-NN algorithm is a conceptually simple local approach and can be applied to nonlinear problems.

(a) In a first step we will implement the basic k-NN regression algorithm. Therefore open the script `Ex3_1ab.m` and complete the code of the `knn_regression` function with the help of the comments and the following description of the algorithm:

- Choose a value for the number of nearest neighbors k and the point x_* you want to obtain a prediction for.
- Determine the k closest training inputs $\boldsymbol{x}_{nn}$ according to some distance function. In our case the euclidean distance is used:

$$d\left(\boldsymbol{x}_*, \boldsymbol{x}\right) = \sqrt{\sum_{j=1}^{n} \left(x_{*_j} - x_j\right)^2} \tag{1}$$

- Estimate $f(x_*)$ using the average of all corresponding training outputs $\boldsymbol{y}_*$:

$$\hat{f}(x_*) = \frac{1}{k} \sum_{y_i \in \boldsymbol{y}_*} y_i \tag{2}$$

(b) Use the function `generate_nonlin_data_1D` to create a training data set of size $N = 50$. Plot the results for $k = 1$ and for $k = 10$. Compare and interpret the results.

(c) In this exercise we want to use Leave-One-Out Cross-Validation (LOOCV) with our k-NN function to estimate the generalization error. Therefore open the script `Ex_1c` and implement a loop over all data points. In each iteration i split the data set into training data $\{(x_1, y_1), \ldots, (x_N, y_N)\} \setminus \{(x_i, y_i)\}$ and test data (x_i, y_i) and calculate the mean squared error (MSE_i) for the test data point (x_i, y_i). Finally calculate the mean value of all MSE_i. Then calculate the $\mathrm{MSE}_{\mathrm{LOOCV}}$ for each $k = 1, 2, \ldots, 10$ and plot the results in one graph. Which k would you choose? Plot the model prediction together with the training data, similar to (b), with your chosen k and comment on the result.

2. Gaussian Process Regression

In the next exercise Gaussian Process Regression will be examined. Therefore open the script `Ex3_2abc.m`.

(a) Complete the function `SqExpKernel` at the end of the script by implementing the squared exponential kernel:

$$k_{\mathrm{SE}}(x, x') = \theta_{\mathrm{f}}^2 \exp\left(-\frac{(x - x')^2}{2\theta_{\mathrm{l}}^2}\right)$$

Plot the kernel function for fixed $x = 0$ and varying $x' \in [-5, 5]$ for all combinations of $\theta_{\mathrm{f}} \in \{0.5, 2\}$ and $\theta_{\mathrm{l}} \in \{0.5, 2\}$.

(b) Now we want to sample from a Gaussian Process (GP). A GP is fully defined by its covariance function and its mean function. We will use our implemented squared exponential kernel as covariance function and assume constant zero mean for the mean function. Construct the covariance matrix using the squared exponential kernel for $N = 100$, $x \in [0, 10]$. Use the `mvnrnd` function to draw 20 random samples from a multivariate normal distribution and plot them in one graph.

(c) From the results in the lecture we know that the conditional distribution for a new test point x_* given the training data is again Gaussian distributed:

$$p(f_*|y) = \mathcal{N}(f_*|\mu_*, \Sigma_*)$$

with

$$\begin{aligned}\mu_* &= \bar{f}_* k_*^T K_y^{-1} y \\ \Sigma_* &= Var(f_*) = k_{**} - k_*^T K_y^{-1} k_*\end{aligned}$$

Implement the prediction by completing the function `gprPred` in the script with the help of the comments and the lecture slides. Generate non-linear data using the `GenerateNonlinData` function. Use your `gprPred` function to fit the data in the range $x \in [-0.2, 1.2]$ with a noise standard deviation of $\sigma_n = 1$ and the kernel parameters $\theta_{\mathrm{f}} = 5$ and $\theta_{\mathrm{l}} = 0.1$. Plot the prediction together with the training data. Add prediction intervals to your plot by adding and subtracting the predicted standard deviation from the estimated mean. Feel free to change the parameters and comment on your findings.

(d) Finally, we want to apply GPR to a real world problem using the MATLAB Machine-Learning Toolbox. The toolbox offers many options and pre-implemented kernels. Moreover the hyperparameters can be determined automatically by optimizing the log likelihood function. Open the `Ex3_2d` script and load the `carsmall` data set.

(i) Fit a `gprModel` using the `fitrgp` function with an `ardsquaredexponential` kernel and otherwise with the standard configuration (maximization of log marginal likelihood using a quasi-Newton optimizer, see the Matlab documentation of `fitrgp` for more details). Make predictions with the training data set as input using the `predict` function. Plot the training data vs. the predictions and calculate the corresponding MSE.

(ii) Predict the MPG for constant Displacement = 200, Weight = 3000 and varying Horsepower $\in [0, 300]$ using the same model. Plot the MPG depending on the Horsepower together with the corresponding 95 % confidence interval.

(iii) Now change the kernel function to the standard `squaredexponential` function. Compare and discuss the results and give reasons for your conclusions.

(iv) Estimate a linear regression model using the `fitlm` function. Compare and discuss the results with your GPR model and comment on the differences.