TP 2 – AOS1

Bayesian linear regression, Gaussian process regression

1 Introduction

This practical session aims at applying Bayesian linear regression in a first step, and Gaussian processes in a second step. You will need several libraries for this purpose.

```
import numpy as np
import scipy as sp
import scipy.stats as spst
import matplotlib.pyplot as plt
```

2 Bayesian linear regression

2.1 Practice

$\boxed{1}$ Create a function for generating synthetic outputs according to a sample of inputs and a user-defined model (i.e. a given functional relation). Generate training and test datasets.

$\boxed{2}$ Program a function which makes predictions given training data `Xtr` and `ytr`, a noise covariance matrix Σ_n, a prior matrix Σ_p, and the set of prediction (test) instances `Xpr`. Return the associated credibility intervals via vectors `ypr_inf` and `ypr_sup`.
You may use the `np.linalg.inv` function for this purpose. Optimizing the function (using, e.g., Cholesky decomposition) is not mandatory.

```
def predGLR(Xpr, Xtr, ytr, Sign, Sigp):
    Xtr = np.concatenate([np.ones(Xtr.shape), Xtr], axis=1)
    Xpr = np.concatenate([np.ones(Xpr.shape), Xpr], axis=1)
    ...

    return [ypr, ypr_cov, ypr_inf, ypr_sup]
```

$\boxed{3}$ Use the Gaussian LR model on the generated data, for several levels of noise and several covariance priors. Represent the credibility intervals obtained using the following code.

```
fig, ax = plt.subplots()
ax.plot(Xtr, ytr, 'k+', label='test data')
ax.plot(Xpl, ypl_ave, label='estimates')
ax.fill_between(Xpl, ypl_inf.reshape(-1), ypl_sup.reshape(-1),
                 color='lightblue', label='credibility interval')
ax.legend()
```

2.2 Theory

$\boxed{4}$ Show that the ML estimates for the weights are obtained by

$$\widehat{\mathbf{w}} = \left(X^\top X\right)^{-1} X^\top \mathbf{y},$$

where X stands for the training input matrix (with training instances $\mathbf{x}_i$ stored row-wise), and $\mathbf{y}$ for the vector of associated outputs y_i.

$\boxed{5}$ We study here the distribution of the ML estimator $\widehat{\mathbf{w}}$ of the parameter vector $\mathbf{w}$.

$\boxed{5\,\mathrm{a}}$ Show that for any Gaussian random vector $\mathbf{U} \sim \mathcal{N}(\mathbf{a}, B)$, then the random vector $\mathbf{V} = \mathbf{c} + D\mathbf{U}$ is such that $\mathbf{V} \sim \mathcal{N}(\mathbf{c} + D\mathbf{a}, DBD^\top)$.

$\boxed{5\,\mathrm{b}}$ Show that the ML estimator $\widehat{\mathbf{w}}$ [1] of the parameter vector $\mathbf{w}$ is distributed as

$$\widehat{\mathbf{w}} \sim \mathcal{N}\left(\mathbf{w}, \sigma_n^2 \left(X^\top X\right)^{-1}\right).$$

3 Gaussian process regression

We consider the `scikit-learn` implementation of Gaussian process regression.

```
from sklearn.gaussian_process import GaussianProcessRegressor as GPR
from SKlearn.gaussian_process.kernels import RBF, ConstantKernel as C
from sklearn.gaussian_process.kernels import DotProduct as DP, WhiteKernel as WK
```

3.1 Practice

$\boxed{6}$ We first consider the noise-free case.

$\boxed{6\,\mathrm{a}}$ Take a function such as e.g. $f(x) = x\,\cos(x)$, with $x \in \mathbb{R}$. Generate training, prediction and plot data accordingly, without noise.

$\boxed{6\,\mathrm{b}}$ Build the GPR model; fit the model to the training data, make predictions, and display.

$\boxed{7}$ We now introduce noise in the model outputs.

```
ytr_noi = model(Xtr.ravel(), 1)
```

$\boxed{7\,\mathrm{a}}$ Train the model with the noisy data *assuming they are noise-free*, and display the results.

$\boxed{7\,\mathrm{b}}$ Display the outputs of a model which assumes the presence of noise in the training data, with a fixed amount of noise (using the `GPR` parameter `alpha`, set for instance to `alpha=1`).

[1] Note that this notation does not make a distinction between the estimator and its realization.