Pract_regression_professor

September 19, 2019

1 Parametric ML and Bayesian regression

```
Notebook version: 1.2 (Sep 28, 2018)

Authors: Miguel Lázaro Gredilla
         Jerónimo Arenas García (jarenas@tsc.uc3m.es)
         Jesús Cid Sueiro (jesus.cid@uc3m.es)

Changes: v.1.0 - First version. Python version
         v.1.1 - Python 3 compatibility. ML section.
         v.1.2 - Revised content. 2D visualization removed.

Pending changes:
```

[1]:
```
# Import some libraries that will be necessary for working with data and␣
 ↪displaying plots

# To visualize plots in the notebook
%matplotlib inline

import matplotlib
import matplotlib.pyplot as plt
import matplotlib.cm as cm

import numpy as np
import scipy.io       # To read matlab files
from scipy import spatial
import pylab
pylab.rcParams['figure.figsize'] = 8, 5
```

1.1 1. Introduction

In this exercise the student will review several key concepts of Maximum Likelihood and Bayesian regression. To do so, we will assume the regression model

$$s = f(\mathbf{x}) + \varepsilon$$

where s is the output corresponding to input $\mathbf{x}$, $f(\mathbf{x})$ is an unobservable latent function, and ε is white zero-mean Gaussian noise, i.e.,

$$\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2).$$

In addition, we will assume that the latent function is *linear in the parameters*

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{z}$$

where $\mathbf{z} = T(\mathbf{x})$ is a possibly non-linear transformation of the input. Along this notebook, we will explore different types of transformations.

Also, we will assume an a priori distribution for $\mathbf{w}$ given by

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_p^2 \, \mathbf{I})$$

1.1.1 Practical considerations

- Though sometimes unavoidable, it is recommended not to use explicit matrix inversion whenever possible. For instance, if an operation like $\mathbf{A}^{-1}\mathbf{b}$ must be performed, it is preferable to code it using python numpy.linalg.lstsq function (see http://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.lstsq.html), which provides the LS solution to the overdetermined system $\mathbf{A}\mathbf{w} = \mathbf{b}$.
- Sometimes, the computation of $\log|\mathbf{A}|$ (where $\mathbf{A}$ is a positive definite matrix) can overflow available precision, producing incorrect results. A numerically more stable alternative, providing the same result is $2\sum_i \log([\mathbf{L}]_{ii})$, where $\mathbf{L}$ is the Cholesky decomposition of $\mathbf{A}$ (i.e., $\mathbf{A} = \mathbf{L}^\top \mathbf{L}$), and $[\mathbf{L}]_{ii}$ is the ith element of the diagonal of $\mathbf{L}$.
- Non-degenerate covariance matrices, such as the ones in this exercise, are always positive definite. It may happen, as a consequence of chained rounding errors, that a matrix which was mathematically expected to be positive definite, turns out not to be so. This implies its Cholesky decomposition will not be available. A quick way to palliate this problem is by adding a small number (such as 10^{-6}) to the diagonal of such matrix.

1.1.2 Reproducibility of computations

To guarantee the exact reproducibility of the experiments, it may be useful to start your code initializing the seed of the random numbers generator, so that you can compare your results with the ones given in this notebook.

[2]:
```
np.random.seed(3)
```

1.2 2. Data generation with a linear model

During this section, we will assume affine transformation

$$\mathbf{z} = T(\mathbf{x}) = (1, \mathbf{x}^\top)^\top$$

.

The a priori distribution of $\mathbf{w}$ is assumed to be

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_p^2 \, \mathbf{I})$$