Start: 27.5.2024
Return: 10.6.2024
Use github link shown in moodle

Assignment 4
Introduction to Probabilistic Machine Learning

1 Introduction

1.1 GitHub Classroom and the GitHub Repo

In this assignment, you will . . .

Part 1: Bayesian Linear Regression (3 Points)

- Relevant source files: `regression.jl`
- Supporting library files from previous assignments: `distributions.jl, functions.jl, example.jl`
- Test files: `test_regression.jl`

In this task, we'd like to implement the Bayesian linear regression, as introduced in the lecture. As a closed-form solution was provided for the problem at hand, we'd like to implement this closed form solution instead of the lengthy factor graph approach.

The only task consists of filling out the function `bayesian_linear_regression` in the file `regression.jl`. The function should take the following arguments:

- ϕ: The matrix input values x, already transformed into the feature space.
- `y`: The target values.
- `prior`: The prior distribution of the weights, using the struct defined in `distributions.jl`.
- `sigma`: The noise parameter, as used in the likelihood definition in the lecture.

The function should return the posterior distribution of the weights, using the same struct as the prior distribution. The function should also return the posterior predictive distribution, which is a Gaussian distribution with mean and variance. Both are introduced at the end of Unit 7.

Part 2: Cholesky Decomposition Implementation (1 Point)

Consider a symmetric, positive semi-definite matrix A. Implement the Cholesky decomposition, to obtain a lower triangular matrix L such that $L \cdot L^T = A$. See, for instance, https://de.wikipedia.org/wiki/Cholesky-Zerlegung (note, here L corresponds to G).

Use your implementation and compute L for the exemplary matrix $A = \begin{pmatrix} 8 & 5 & 2 \\ 5 & 5 & 0 \\ 2 & 0 & 8 \end{pmatrix}$.

An empty function for your implementation is provided along with a test file which implements the matrix above. Please note that we expect your function to check for the validity of the input matrix A. Please explain your implementation in the solution section.

Part 3: Solving Matrix Inverse Problems with Back-Substitution (1 Point)

Consider a regular matrix A and a given vector $\vec{b}$. The goal is to find $\vec{x}$ such that $A \cdot \vec{x} = \vec{b}$. Instead of computing the inverse of A, we want to use the lower-triangular matrix L associated to A, see Part 2 above.

Implement back-substitution as described in Unit 6, slide 25. Compute $\vec{x}$ for the exemplary matrix A, see Part 2, and the vector $\vec{b} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$, i.e., $\vec{b}^T = (1, 2, 3)$.

(i) First, given L and $\vec{b}$, compute $\vec{y}$ (cf. $\vec{y} = L \backslash \vec{b}$), i.e., such that $L \cdot \vec{y} = \vec{b}$.

(ii) Second, given L^T and $\vec{y}$, cf. (i), compute $\vec{x}$ (cf. $\vec{x} = L^T \backslash \vec{y}$), i.e., such that $L^T \cdot \vec{x} = \vec{y}$.

(iii) Finally, check whether $A \cdot \vec{x} = \vec{b}$ holds.

Hint: In step (ii), the recursion used in (i), see Unit 6, slide 25, has to be slightly adapted as L^T is of different structure compared to L.