Start: 27.10.2025
Return: 10.11.2025
classroom.github.com/a/5u9Do-ya

Assignment 1
Advanced Probabilistic Machine Learning

1 Introduction

GitHub Classroom and the GitHub Repo

In this assignment, you will learn how to represent distributions and the related operations in Julia. In Part 2 and 3, you will recap parts of the first two lectures.

In Part 1, please fill out all parts marked with #TODO# in the 4 relevant source files. The annotations in the code aim to assist you on your path.

In Part 2 and 3, please provide your solutions as PDF with the respective name. You can use for example Word or Tex to generate this PDF.

Part 1: Distribution Julia Modules (4 Points)

- Relevant source files: `gaussian.jl`, `gamma.jl`, `dirichlet.jl`, `categorical.jl`

For the following lessons and assignments, we need Julia implementations of common probability distributions, that incorporate the ability to multiply and divide those distributions with each other. In this assignment, we will focus on the following four distributions:

- A **1D Gaussian distribution** over $\mathbb{R}$ characterized by the precision mean τ and precision ρ.
- A **1D Gamma distribution** over $\mathbb{R}$ characterized by the shape parameter β and the rate parameter λ.
- A **Dirichlet distribution** over the $(k-1)$-probability simplex characterized by the concentration parameters $\alpha = (\alpha_1, \ldots, \alpha_k) \in \mathbb{R}^k_{>0}$.
- A **Categorical distribution** over the values $1, \ldots, n$ characterized by the log-probabilities of $1, \ldots, n$.

The files should implement the respective distributions as parametric Julia types with the following interfaces:

- Normalized constructor: Should initialize a distribution given their parameters.
- Non-normalized constructor: Should initialize a distribution with stored normalization constant given their parameters and a normalization constant.
- Converters: Should convert the distributions from normalized to non-normalized and vice versa.
- Special constructors: Should construct a distribution for edge cases, e.g.`Gaussian1DFromMeanVariance()` should construct a Gaussian distribution from mean and variance parametrization.
- Improper uniform constructors: Should return a uniform improper distribution, e.g. `Gaussian1DUniform()`.
- `mean()`: Should compute the mean.
- `variance()`: Should compute the variance.
- `KL_divergence()`: Should compute the Kullback–Leibler divergence between two distributions of the same type.
- `is_uniform()`: Should check wether a distribution is an improper uniform distribution.
- `*`: Should use operator overloading of `Base.:*` to multiply two distributions.
- `/`: Should use operator overloading of `Base.:/` to divide two distributions.
- `distribution()`: Should convert a distribution into an object from the `Distributions.jl` library.

Part 2: Exponential Family (2 Points)

Probability density function that are within the exponential family can be easily multiplied with each other which helps to efficiently derive posterior distributions, see the APML slides of week 1.

(1) Consider two Gaussian densities $X \sim \mathcal{N}(x; \mu_1, \sigma_1^2)$ and $Y \sim \mathcal{N}(x; \mu_2, \sigma_2^2)$, $x \in \mathbb{R}$, $\mu_i \in \mathbb{R}$, $\sigma_i^2 > 0$, $i = 1, 2$. Derive the probability density of $Z \sim \mathcal{N}(x; \mu_3, \sigma_3^2) \propto \mathcal{N}(x; \mu_1, \sigma_1^2) \cdot \mathcal{N}(x; \mu_2, \sigma_2^2)$.

(2) Consider two Categorical densities $\boldsymbol{X} \sim \mathrm{Cat}(\boldsymbol{x}; \boldsymbol{\pi}_1)$ and $\boldsymbol{Y} \sim \mathrm{Cat}(\boldsymbol{x}; \boldsymbol{\pi}_2)$, $\boldsymbol{x}$ within the first K unit vectors, $\boldsymbol{\pi}_i \in [0, 1]^K$, where components add up to 1, $i = 1, 2$. Derive the probability density of $\boldsymbol{Z} \sim \mathrm{Cat}(\boldsymbol{x}; \boldsymbol{\pi}_3) \propto \mathrm{Cat}(\boldsymbol{x}; \boldsymbol{\pi}_1) \cdot \mathrm{Cat}(\boldsymbol{x}; \boldsymbol{\pi}_2)$.

(3) Consider two Dirichlet densities $\boldsymbol{X} \sim \mathrm{Dir}(\boldsymbol{x}; \boldsymbol{\alpha}_1)$ and $\boldsymbol{Y} \sim \mathrm{Dir}(\boldsymbol{x}; \boldsymbol{\alpha}_2)$, $\boldsymbol{x}$ within the first K unit vectors, $\boldsymbol{\alpha}_i \in \mathbb{R}^{+K}$, $i = 1, 2$. Derive the probability density of $\boldsymbol{Z} \sim \mathrm{Dir}(\boldsymbol{x}; \boldsymbol{\alpha}_3) \propto \mathrm{Dir}(\boldsymbol{x}; \boldsymbol{\alpha}_1) \cdot \mathrm{Dir}(\boldsymbol{x}; \boldsymbol{\alpha}_2)$.

(4) Consider two Gamma densities $X \sim \mathrm{Gam}(x; \beta_1, \lambda_1)$ and $Y \sim \mathrm{Gam}(x; \beta_2, \lambda_2)$, $x \in \mathbb{R}^+$, $\beta_i > -1$, $\lambda_i > 0$, $i = 1, 2$. Derive the probability density of $Z \sim \mathrm{Gam}(x; \beta_3, \lambda_3) \propto \mathrm{Gam}(x; \beta_1, \lambda_1) \cdot \mathrm{Gam}(x; \beta_2, \lambda_2)$.

State the result as a function of the given parameters. You do not have to compute of implement anything.

Part 3: Solving Systems of Equations (2 Points)

Assume a regular matrix M and the associated matrix $A := M^T M$. Now, consider the symmetric, positive semi-definite matrix A and a given vector $\vec{b}$. The goal is to find $\vec{x}$ such that $A \cdot \vec{x} = \vec{b}$.

(a) Compute the inverse A^{-1} of A and determine $\vec{x}$ via $\vec{x} = A^{-1} \cdot \vec{b}$. You can use standard algorithms or packages for the numerical example given in (c) below.

(b) Now, we want to obtain $\vec{x}$ using the Cholesky-Decomposition. We need to determine the lower-triangular matrix L associated to A, see the APML slides of week 2.

(i) Determine the lower triangular matrix L such that $L \cdot L^T = A$.

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

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

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

(c) Apply your general implementation of (a) and (b) for the exemplary matrix $A = \begin{pmatrix} 8 & 5 & 2 \\ 5 & 5 & 0 \\ 2 & 0 & 8 \end{pmatrix}$ and the vector $\vec{b} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$, i.e., $\vec{b}^T = (1, 2, 3)$. Which approach is more efficient (a) or (b)?