The TURING Language

From Bayesian Inference to Probabilistic Programming

徐锴 Kai Xu
2020 年 8 月 23 日 Julia 中文社区夏季会议
University of Edinburgh & The TURING Language Team

Table of contents

A Brief Introduction to Bayesian Inference

数据、建模和机器学习

我们先简要介绍一下 Turing.jl 的应用场景

- 假定我们给定来一些数据 data $\mathcal{D}$
- 为了解决一些数据分析的任务，我们引入了一个带参数 θ 的模型
- 我们的首要任务就是让模型从数据中学习：就是给定 $\mathcal{D}$，来找 θ

  优化方法 给定 $\mathcal{D}$，找到最好的 θ^*

  贝叶斯推断 给定 $\mathcal{D}$，描述整个 $P(\theta \mid \mathcal{D})$ 分布
- 做分析任务，如某个预测 $f(\theta)$，的时候，使用学习到的模型（参数）

  优化方法 用最好的参数 $f(\theta^*)$

  贝叶斯推断 用整个分布做“加权平均” $\int P(\theta \mid \mathcal{D}) f(\theta) \mathrm{d}\theta$
- 相对于优化方法，贝叶斯推断能够考虑模型不确定性 model uncertainty

贝叶斯定理 The Bayes' theorem

The famous Bayes' rule states

$$P(\theta \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \theta)P(\theta)}{P(\mathcal{D})}$$

$$\begin{aligned}
\mathcal{D} &\quad \text{数据 data} \\
\theta &\quad \text{模型参数 model parameters} \\
P(\theta) &\quad \text{先验 prior} \\
P(\mathcal{D} \mid \theta) &\quad \text{似然 likelihood} \\
P(\theta \mid \mathcal{D}) &\quad \text{后验 posterior} \\
P(\mathcal{D}) &\quad \text{边缘似然 marginal likelihood}
\end{aligned}$$

计算后验 $\approx$ 学习模型

生成建模 Generative modelling

建模 = 描述数据生成的过程

- 似然函数 $P(\mathcal{D} \mid \theta)$ 描述了给定模型参数 θ 数据如何生成
- 先验 $P(\theta)$ 描述了模型参数如何生成
- 用 $P(\theta)$ 生成模型参数 $\rightarrow$ 用 $P(\mathcal{D} \mid \theta)$ 生成数据
- 建模 = 描述先验 & 似然函数

丢硬币

- 数据 $\mathcal{D} = \{0, 1, 0, 1, 0\}$ (0 代表反面，1 代表正面)
- 模型 = 伯努利分布 Bernoulli distribution
  - 模型参数 θ 是硬币投出正面的概率
  - $P(x \mid \theta) = \mathcal{B}er(x; \theta) = \theta^x (1 - \theta)^{1-x}$
  - $P(\mathcal{D} \mid \theta) = \prod_{x \in \mathcal{D}} P(x \mid \theta)$
  - 给定模型参数 θ 的情况下，生成数据 = 从伯努利分布中采样
  - 如何先验 $P(\theta)$ 呢?

贝叶斯推断 Bayesian inference

假定模型已经确定，即先验 $P(\theta)$ 和似然函数 $P(\mathcal{D} \mid \theta)$ 已经确定

- 学习，或推断 *inference* = 求解 $P(\theta \mid \mathcal{D})$
- 已知 $P(\theta)$ 和 $P(\mathcal{D} \mid \theta)$ 求 $P(\theta \mid \mathcal{D}) \to$ 贝叶斯定理
- 边缘似然 $P(\mathcal{D})$ 怎么办?
    1. 求积分 $P(\mathcal{D}) = \int P(\mathcal{D} \mid \theta)P(\theta)\mathrm{d}\theta$
    2. 忽略它 $P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta)P(\theta)$ – 因为 $P(\mathcal{D})$ 是一个不关于 θ 的量

如果想系统学习贝叶斯推断，推荐阅读 David MacKay 的《信息论、推理与学习算法》*Information Theory, Inference and Learning Algorithms* [2].

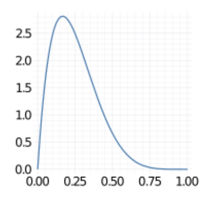
共轭先验 Conjugate priors

并不是所有积分 $P(\mathcal{D}) = \int P(\mathcal{D} \mid \theta)P(\theta)\mathrm{d}\theta$ 都好求

- 当先验和似然共轭的时候，好求！

还是丢硬币

- 似然 = 伯努利分布 Bernoulli distribution
- 共轭先验是贝塔分布 Beta distribution
  - 支撑集 support 是 $(0, 1)$ 和伯努利的参数匹配
  - $P(\theta) = \mathcal{B}eta(\theta; \alpha, \beta) = \theta^{\alpha-1}(1-x)^{\beta-1}/B(\alpha, \beta)$, $B(\alpha, \beta)$ 是贝塔函数
- 定义 t 和 h 分别是数据 $\mathcal{D}$ 里的 0 和 1
- 贝塔先验 $\mathcal{B}eta(\theta; \alpha, \beta)$ + 伯努利似然
  $\rightarrow$ 贝塔后验 $\mathcal{B}eta(\theta; \alpha + h, \beta + t)$
  算积分算出来的 $\uparrow$



重要的事重复一遍：并不是所有积分都好求 → 限制了建模灵活性

马尔可夫链蒙特卡洛 Markov chain Monte Carlo (MCMC)

只关注 $P(\theta \mid \mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})} \propto P(\mathcal{D} \mid \theta)P(\theta)$

- 做预测 = 求积分 $\int P(\theta \mid \mathcal{D}) f(\theta) \mathrm{d}\theta$
  - $f(\theta)$ 表示用模型参数 θ 的预测
  - 积分 = 考虑参数在后验下的所有可能预测
  - 通常可以用蒙特卡洛 Monte Carlo 估计，即 $\int P(\theta \mid \mathcal{D}) f(\theta) \mathrm{d}\theta \approx \frac{1}{N} \sum_i f(\theta_i), \quad \theta_i \sim P(\theta \mid \mathcal{D})$
- 贝叶斯推断 ≈ 能够从 $P(\theta \mid \mathcal{D})$ 采样

MCMC 是一种从分布 $p(x)$ 中采样的方法

- 条件：不需知道 $p(x)$ 的完整形式，只需知道 $p(x) = \frac{\bar{p}(x)}{Z}$ 中的分子
  - $\bar{p}(x)$ 是没有正则化的概率 unnormalized probability
- 说也巧，先验 $P(\theta)$ 和似然 $P(\mathcal{D} \mid \theta)$ 是我们知道的
- 联合概率 joint probability 一般指乘积 $P(\mathcal{D} \mid \theta)P(\theta) = P(\mathcal{D}, \theta)$

Probabilistic Programming using Turing.jl

概率编程 Probabilistic programming

为什么要做概率编程?

- 生成建模很麻烦，要写好多概率分布
- 推断更麻烦
    - 积分算不来
    - MCMC 也不好实现
- 模型不对 $\rightarrow$ 所有麻烦从头再来 – the *iterative* nature of modelling

概率编程框架 = 有概率语义的编程（即建模）+ 自动贝叶斯推断

The Turing language

Turing [1] 是基于 Julia 的概率编程语言。

- 像写数学公式一样的建模
  - Distributions.jl 提供来丰富的概率分布
  - 在 Turing 里可以直接写 $x \sim \mathcal{N}ormal(0, 1)$ 来定义随机变量
- 任何概率图模型 probabilistic graphical model (PGM) 都能用 Turing 表达
  - 支持控制流程 control flow
- 可能拥有是所有概率编程框架中最丰富的推断算法库
- 跑得快!

⋆ 基于我们都喜爱的 Julia ⋆

让我们一起来看看 TURING.JL 吧！

Demo Time

团队和贡献者 The TURING Language team & contributors

欢迎参与: `https://github.com/TuringLang/Turing.jl`

Thanks everyone for making TURING this far!

Core team Hong Ge, Kai Xu, Martin Trapp, Will Tebbutt, Mohamed Tarek, Tor Erlend Fjelde, Cameron Pfiffer, David Widmann, Qingliang Zhuo, Philipp Gabler, Miles Lucas, Zoubin Ghahramani

Collaborators Emile Mathieu, Yee Whye Teh

GSoC & GSoD students Sharan Yalburgi, Arthur Lui, Saranjeet Kaur

GitHub contributors Andreas Noack, Seth Axen, Elizaveta Semenova, Christopher Rackauckas, . . . and more at `https://github.com/TuringLang/Turing.jl/graphs/contributors`

Questions?

谢谢！

References

H. Ge, K. Xu, and Z. Ghahramani.
Turing: A language for flexible probabilistic inference.
In *International Conference on Artificial Intelligence and Statistics*, pages 1682–1690, Mar. 2018.

D. J. MacKay.
Information theory, inference and learning algorithms.
Cambridge university press, 2003.