Statistical Computing
Chap. 4.2: Variational Inference and Pyro

LIU, Ran

Department of Statistics,
Beijing Normal University

May 5, 2024

北京師範大學
BEIJING NORMAL UNIVERSITY

Summary

Variational Inference

Pyro

变分贝叶斯推断

- 变分贝叶斯推断常用于推断潜变量给定观察值（和参数）的条件分布，即潜变量的后验分布。
- 后验可以写作:

$$p(z \mid x) = \frac{p(z, x)}{\int p(z, x) dz}$$

- 由于直接计算后验分布通常不可行（涉及到不易解析求解的积分），我们需要使用近似推断方法。
- 目标是找到一个近似分布 $q(z)$，使得它能够尽可能接近真实后验分布，通常通过最小化两者之间的某种距离来实现。

Kullback-Leibler 散度

- 我们通过 Kullback-Leibler (KL) 散度来衡量两个分布之间的接近程度:

$$KL(q\|p) = \int q(z) \log \frac{q(z)}{p(z \mid x)} dz$$

- KL 散度的直观解释:
  - 如果 q 高且 p 高，则我们满意（KL 散度低）。
  - 如果 q 高且 p 低，则代价高（KL 散度高）。
  - 如果 q 低，则我们不关心（KL 散度也低）。

证据下界 (ELBO)

- 为了进行变分贝叶斯推断，我们希望最小化 KL 散度。但直接最小化 KL 散度往往不可行。
- 我们可以最小化一个与 KL 散度等价的函数——证据下界（ELBO）

$$\text{ELBO} = \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q(z)]$$

- 最大化 ELBO 等价于最小化 KL 散度。

最大化 ELBO 等价于最小化 KL 散度。我们从 KL 散度的定义开始：

$$\mathrm{KL}(q\|p) = \int q(z) \log \frac{q(z)}{p(z|x)} dz$$

ELBO 定义为：

$$\mathrm{ELBO}(q) = \mathbb{E}_q[\log p(x,z)] - \mathbb{E}_q[\log q(z)]$$

使用贝叶斯定理重写 $p(x,z)$：

$$\mathrm{ELBO}(q) = \mathbb{E}_q[\log p(z|x) + \log p(x)] - \mathbb{E}_q[\log q(z)]$$

简化并与 KL 散度联系：

$$\mathrm{ELBO}(q) = -\mathrm{KL}(q\|p) + \log p(x)$$

因此，由 $p(x)$ 固定可知，最大化 ELBO 就是最小化 KL 散度。

均场变分推断

- 在均场变分推断中，我们假设对潜变量的变分分布具有因式分解形式:

$$q(z_1, \ldots, z_m) = \prod_{j=1}^{m} q(z_j)$$

- 这种设置假设了后验各个隐变量相互独立，有一定的限制。

均场变分推断的优化

- 通常使用坐标上升优化方法（即依次优化每个潜变量的变分近似，同时保持其他变量固定）。
- 对于每个潜变量 z_j，更新规则为:

$$q^*(z_j) \propto \exp\left\{\mathbb{E}_{-j}[\log p(z_j \mid z_{-j}, x)]\right\}$$

- 这种更新过程通常收敛到局部最大值。

潜变量更新规则的推导

在均场近似中，变分分布假设为：

$$q(z) = \prod_{j=1}^{m} q(z_j)$$

我们要最大化的 ELBO 是：

$$\text{ELBO}(q) = \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q(z)]$$

对于单个潜变量 z_j，ELBO 可简化为：

$$\mathbb{E}_{q_{-j}}[\mathbb{E}_{q_j}[\log p(x, z)]] - \mathbb{E}_{q_j}[\log q(z_j)]$$

最优的 $q(z_j)$ 与条件概率成比例：

$$q^*(z_j) \propto \exp(\mathbb{E}_{q_{-j}}[\log p(x, z)|z_j])$$

这表明 $q(z_j)$ 应逼近 $p(z_j|z_{-j}, x)$。

变分推断算法

- 选择一个参数化的分布家族 $q(z;\phi)$ 作为近似分布，其中 ϕ 表示分布参数。
- 初始化参数 ϕ。
- 使用梯度上升算法迭代更新参数 ϕ 以最大化 ELBO：

$$\phi^{(new)} = \phi^{(old)} + \eta \nabla_\phi \text{ELBO}(\phi)$$

- 其中，η 是学习率，$\nabla_\phi \text{ELBO}(\phi)$ 是 ELBO 相对于参数 ϕ 的梯度。

挑战与未来方向

- 模型复杂性：在处理具有高维参数空间或复杂依赖结构的模型时，变分推断的计算复杂度和精度可能成问题。
- 局部最优：由于 ELBO 的优化通常涉及非凸优化问题，优化过程可能陷入局部最优解。
- 分布家族的选择：选择的 $q(z)$ 分布家族可能过于简单，无法捕捉后验分布的所有特性，如多峰性。
- 未来研究方向：研究更灵活的变分分布家族，如正态化流（Normalizing Flows）和变分自编码器（VAEs），以及改进优化算法，提高计算效率和近似精度。

Summary

Variational Inference

Pyro

Pyro

Pyro 是一种用 Python 编写的通用概率编程语言 (probabilistic programming language)，并在后端由 PyTorch 提供支持。

其他概率编程语言：Pymc3, Stan...

* 参考：Tutorial on deep probabilitic modeling with pyro

Pyro

首先设定所有 latent(包含 parameter) 的 prior, 以及观测值由 latent 怎么采样得到。(相当于我们定义了目标函数)

```
def model(is_cont_africa, ruggedness, log_gdp=None):
    a = pyro.sample("a", dist.Normal(0., 10.))
    b_a = pyro.sample("bA", dist.Normal(0., 1.))
    b_r = pyro.sample("bR", dist.Normal(0., 1.))
    b_ar = pyro.sample("bAR", dist.Normal(0., 1.))
    sigma = pyro.sample("sigma", dist.Uniform(0., 10.))
    mean = a + b_a * is_cont_africa + b_r * ruggedness + b_ar *
     is_cont_africa * ruggedness

    with pyro.plate("data", len(ruggedness)):
        return pyro.sample("obs", dist.Normal(mean, sigma), obs
    =log_gdp)

pyro.render_model(model, model_args=(is_cont_africa, ruggedness
    , log_gdp), render_distributions=True)
```

接下来，pyro 一般使用变分推断，比如用一个正态分布去近似这个后验分布，去不断学习这个正态分布的 mean 和 var. (相当于我们定义了 prediction 的结构)

```
pyro.clear_param_store()

auto_guide = pyro.infer.autoguide.AutoNormal(model)
pyro.infer.MCMC(auto_guide)
adam = pyro.optim.Adam({"lr": 0.02})
elbo = pyro.infer.Trace_ELBO()
svi = pyro.infer.SVI(model, auto_guide, adam, elbo)
losses = []
for step in range(1000):
    loss = svi.step(is_cont_africa, ruggedness, log_gdp)
    losses.append(loss)
    if step % 100 == 0:
        logging.info("Elbo loss: {}".format(loss))
```


* 参考 [知乎]：变分推断 - 深度概率编程 - Pyro 第一个完整例子

如果将 SVI 改成 NUTS 的话:

```
pyro.clear_param_store()

auto_guide = pyro.infer.autoguide.AutoNormal(model)
nuts_kernel = pyro.infer.NUTS(model)
mcmc = pyro.infer.MCMC(nuts_kernel, num_samples=1000,
    warmup_steps=200)
mcmc.run(is_cont_africa, ruggedness, log_gdp)

samples = mcmc.get_samples()

# Access the desired samples or statistics from the MCMC run
# For example:
# samples["parameter_name"]

# You can also access the posterior predictive distribution
    using
# mcmc.predictive(is_cont_africa, ruggedness, log_gdp)
```