Markov Chain Monte Carlo for Inverse Problems

David Ochsner

April 24, 2020

Contents

1 Theory

1.1 Papers

1.1.1 Stuart et al: Inverse Problems: A Bayesian Perspective [3]

Theoretical Background

1.1.2 Cotter et al: MCMC for functions [1]

Implementation, MCMC in infinite dimensions

1.1.3 Schneider et al: Earth System Modeling 2.0 [2]

Example for MCMC on ODE

1.2 Small results

1.2.1 Bayes' Formula & Radon-Nikodym Derivative

Bayes' Formula is stated using the Radon-Nikodym Derivative in both [1] and [3]:

$$\frac{\mathrm{d}\mu}{\mathrm{d}\mu_0} \propto \mathrm{L}(u),$$

where $\mathrm{L}(u)$ is the likelihood.

Write the measures as $\mathrm{d}\mu = \rho(u)\mathrm{d}u$ and $\mathrm{d}\mu_0 = \rho_0(u)\mathrm{d}u$ with respect to the standard Lesbesgue measure. Then we have

$$\int f(u)\rho(u)\mathrm{d}u = \int f(u)\mathrm{d}\mu(u) = \int f(u)\frac{\mathrm{d}\mu(u)}{\mathrm{d}\mu_0(u)}\mathrm{d}\mu_0 = \int f(u)\frac{\mathrm{d}\mu(u)}{\mathrm{d}\mu_0(u)}\rho_0(u)\mathrm{d}u,$$

provided that $\mathrm{d}\mu$, $\mathrm{d}\mu_0$ and f are nice enough (which they are since we're working with Gaussians). This holds for all test functions f, so it must hold pointwise:

$$\frac{\mathrm{d}\mu(u)}{\mathrm{d}\mu_0(u)} = \frac{\rho(u)}{\rho_o(u)}.$$

Using this we recover the more familiar formulation of Bayes' formula:

$$\frac{\rho(u)}{\rho_o(u)} \propto \mathrm{L}(u).$$

1.2.2 Acceptance Probability for Metropolis-Hastings

A Markov process with transition probabilities $t(y|x)$ has a stationary distribution $\pi(x)$.

- The existence of $\pi(x)$ follows from *detailed balance*:

  $$\pi(x)t(y|x) = \pi(y)t(x|y).$$

  Detailed balance is sufficient but not necessary for the existence of a stationary distribution.

- Uniqueness of $\pi(x)$ follows from the Ergodicity of the Markov process. For a Markov processto be Ergodic it has to:
  - not return to the same state in a fixed interval
  - reach every state from every other state in finite time

The Metropolis-Hastings algorithm constructs transition probabilities $t(y|x)$ such that the two conditions above are satisfied and that $\pi(x) = P(x)$, where $P(x)$ is the distribution we want to sample from.

Rewrite detailed balance as

$$\frac{t(y|x)}{t(x|y)} = \frac{P(y)}{P(x)}.$$

Split up the transition probability into proposal $g(y|x)$ and acceptance $a(y,x)$. Then detailed balance requires

$$\frac{a(y,x)}{a(x,y)} = \frac{P(y)g(x|y)}{P(x)g(y|x)}.$$

Choose

$$a(y,x) = \min\left\{1, \frac{P(y)g(x|y)}{P(x)g(y|x)}\right\}$$

to ensure that detailed balance is always satisfied. Choose $g(y|x)$ such that ergodicity is fulfilled.

If the proposal is symmetric ($g(y|x) = g(x|y)$), then the acceptance takes the simpler form

$$a(y,x) = \min\left\{1, \frac{P(y)}{P(x)}\right\}.$$

Since the target distribution $P(x)$ only appears as a ratio, normalizing factors can be ignored.

1.2.3 Acceptance Probabilities for different MCMC Proposers

Start from Bayes' formula and rewrite the likelyhood L(u) as $\exp(-\Phi(u))$ for a positive scalar function Φ called the potential:

$$\frac{\rho(u)}{\rho_o(u)} \propto \exp(\Phi(u)).$$

Assuming our prior to be a Gaussian ($\mu_0 \sim \mathcal{N}(0, \mathcal{C})$).

(IS WRITING $\rho_0(u) \propto \exp\left(-\frac{1}{2}u^T C^{-1} u\right)$ ASSUMING FINITE DIMENSIONS? WHAT ABOUT $\rho_0(u) \propto \exp\left(-\frac{1}{2}\left\|C^{-1/2}u\right\|^2\right)$? I assume the former is not, for C to be a proper covariance operator it should be invertible. But taking the square root is probably not always well defined for infinite dimensions (so the latter one is problematic))

Then

$$\rho(u) \propto \exp\left(-\Phi(u) + \frac{1}{2}\left\|C^{-1/2}u\right\|^2\right),$$

since $u^T C^{-1} u = (C^{-1/2}u)^T (C^{-1/2}u) = \langle C^{-1/2}u, C^{-1/2}u\rangle = \left\|C^{-1/2}u\right\|^2$, where in the first equality we used C being symmetric.

This is formula (1.2) in [1] and is used in the acceptance probability for the standard random walk (see also Acceptance Probability for Metropolis-Hastings)

$\mathcal{C}^{-1/2}u$ makes problems in infinite dimensions.

Todo: Why exactly is the second term (from the prior) cancelled when doing pCN?

1.2.4 Different formulations of multivariate Gaussians

THIS WHOLE SECTION ASSUMES FINITE DIMENSIONS

Is an RV $\xi \sim \mathcal{N}(0, C)$ distributed the same as $C^{1/2}\xi_0$, with $\xi_0 \sim \mathcal{N}((,0), \mathcal{I})$?

Is $C^{1/2}\exp\left(\frac{1}{2}x^T x\right) = \exp\left(\frac{1}{2}x^T C^{-1} x\right)$?

From wikipedia: Affine transformation $Y = c + BX$ for $X \sim \mathcal{N}(\mu, \Sigma)$ is also a Gaussian $Y \sim \mathcal{N}\left(c + B\mu, B\Sigma B^T\right)$. In our case $X \sim \mathcal{N}(0, I)$, so $Y \sim \mathcal{N}\left(0, C^{1/2}\mathcal{I}C^{1/2^T}\right) = \mathcal{N}(0, C)$, since the covariance matrix is positive definite, which means it's square root is also positive definite and thus symmetric.

2 Implementation

2.1 Framework/Package Structure

The framework is designed to support an easy use case:

```
proposer = StandardRWProposer(beta=0.25, dims=1)
accepter = AnalyticAccepter(my_distribution)
rng = np.random.default_rng(42)
sampler = MCMCSampler(rw_proposer, accepter, rng)

samples = sampler.run(x_0=0, n_samples=1000)
```

There is only one source of randomness, shared among all classes and supplied by the user. This facilitates reproducability.

Tests are done with `pytest`.

2.1.1 Distributions

A class for implementing probability distributions.

```
class DistributionBase(ABC):
    @abstractmethod
    def sample(self, rng):
        """Return a point sampled from this distribution"""
        ...
```

The most important realisation is the `GaussianDistribution`, used in the proposers.

```
class GaussianDistribution(DistributionBase):
    def __init__(self, mean=0, covariance=1):
        ...

    def sample(self, rng):
        ...

    def apply_covariance(self, x):
        ...

    def apply_sqrt_covariance(self, x):
        ...
```

```python
    def apply_precision(self, x):
        ...

    def apply_sqrt_precision(self, x):
        ...
```

The design of this class is based on the implementation in muq2. The `precision` / `sqrt_precision` is implemented through a Cholesky decomposition, computed in the constructor. This makes applying them pretty fast ($\mathcal{O}(n^2)$).

At the moment the there is one class for both scalar and multivariate Gaussians. This introduces some overhead as it has to work with both `float` and `np.array`. Maybe two seperate classes would be better.

2.1.2 Proposers

Propose a new state v based on the current one u.

```python
class ProposerBase(ABC):
    @abstractmethod
    def __call__(self, u, rng):
        ...
```

1. StandardRWProposer

   Propose a new state as

   $$v = u + \sqrt{2\delta}\xi,$$

   with either $\xi \sim \mathcal{N}(0, \mathcal{I})$ or $\xi \sim \mathcal{N}(0, \mathcal{C})$ (see section 4.2 in [1]).

   This leads to a well-defined algorithm in finite dimensions. This is not the case when working on functions (as described in section 6.3 in [1])

2. pCNProposer

   Propose a new state as

   $$v = \sqrt{1-\beta^2}u + \beta\xi,$$

   with $\xi \sim \mathcal{N}(0, \mathcal{C})$ and $\beta = \frac{8\delta}{(2+\delta)^2} \in [0, 1]$ (see formula (4.8) in [1]).

   This approach leads to an improved algorithm (quicker decorrelation in finite dimensions, nicer properties for infinite dimensions)(see sections 6.2 + 6.3 in [1]).

The wikipedia-article on the Cholesky-factorization mentions the use-case of obtaining a correlated sample from an uncorrelated one by the Cholesky-factor. This is not implemented here.

2.1.3 Accepters

Given a current state u and a proposed state v, decide if the new state is accepted or rejected.

For sampling from a distribution $P(x)$, the acceptance probability for a symmetric proposal is $a = \min\{1, \frac{P(v)}{P(u)}\}$ (see 1.2.2)

```
class ProbabilisticAccepter(AccepterBase):
    def __call__(self, u, v, rng):
        """Return True if v is accepted"""
        a = self.accept_probability(u, v)
        return a > rng.random()

    @abstractmethod
    def accept_probability(self, u, v):
        ...
```

1. AnalyticAccepter

   Used when there is an analytic expression of the desired distribution.

   ```
   class AnalyticAccepter(ProbabilisticAccepter):
       def accept_probability(self, u, v):
           return self.rho(v) / self.rho(u)
   ```

2. StandardRWAccepter

   Based on formula (1.2) in [1]:

   $$a = \min\{1, \exp(I(u) - I(v))\},$$

   with

   $$I(u) = \Phi(u) + \frac{1}{2}\left\|\mathcal{C}^{-1/2}u\right\|^2$$

   .

   See also 1.2.3.

3. pCNAccepter

   Works together with the pCNProposer to achieve the simpler expression for the acceptance

$$a = \min\{1, \exp(\Phi(u) - \Phi(v))\}.$$

4. CountedAccepter

   Stores and forwards calls to an "actual" accepter. Counts calls and accepts and is used for calculating the acceptance ratio.

2.1.4 Sampler

The structure of the sampler is quite simple, since it can rely heavily on the functionality provided by the Proposers and Accepters.

```
class MCMCSampler:
    def __init__(self, proposal, acceptance, rng):
        ...

    def run(self, u_0, n_samples, burn_in=1000, sample_interval=200):
        ...

    def _step(self, u, rng):
        ...
```

2.2 Results

2.2.1 Analytic sampling from a bimodal Gaussian

2.2.2 Bayesian inverse problem for $\mathcal{G}(u) = \langle g, u \rangle$

2.2.3 Bayesian inverse problem for $\mathcal{G}(u) = g(u + \beta u^3)$

2.2.4 Geophysics example

References

[1] S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. MCMC Methods for Functions: Modifying Old Algorithms to Make Them Faster. *Statistical Science*, 28(3):424–446, August 2013. Publisher: Institute of Mathematical Statistics.

[2] Tapio Schneider, Shiwei Lan, Andrew Stuart, and João Teixeira. Earth System Modeling 2.0: A Blueprint for Models That Learn From Observations and Targeted High-Resolution Simulations. *Geophysical Research Letters*, 44(24):12,396–12,417, 2017. _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2017GL076101.

[3] A. M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, 19:451–559, May 2010.