

pipeline: TODO

John Letey¹, Tony E. Wong¹

ABSTRACT

TODO!!! The code is available online at <http://mussles.github.io/pipeline/> under the GNU General Public License v3.

Subject headings: methods: statistical — methods: Markov chain Monte Carlo
— TODO

¹University of Colorado at Boulder

Note: If you want to get started immediately with the `pipeline` package, start at Appendix A on page 5 or visit the online documentation at <https://mussles.github.io/pipeline>. If you are sampling with `pipeline` and having low-acceptance-rate or other issues, there is some advice in Section 3 starting on page 4.

1. Introduction

Throughout previous years of development, Markov chain Monte Carlo (MCMC) has evolved from a foregone algorithm to a more general way to fit real-life data.

2. The Algorithm

One of the most commonly used MCMC algorithms is the Metropolis-Hastings algorithm, proposed by Metropolis and Hastings, respectively, in their separate papers [Metropolis, et al. \(1953\)](#) and [Hastings \(1970\)](#). In short, the algorithm states: Given a probability density π , called the target, defined on a state space X , and computable up to a multiplying constant, $\pi(x) \propto \pi(x)$, the Metropolis-Hastings algorithm proposes a generic way to construct a Markov chain on X that is ergodic and stationary with respect to π – meaning that, if $X(t) \sim \pi(x)$, then $X(t+1) \sim \pi(x)$ – and that therefore converges in distribution to π . While there are other generic ways of delivering Markov chains associated with an arbitrary stationary distribution, see, e.g., [Barker \(1965\)](#), the Metropolis-Hastings algorithm is the workhorse of MCMC methods, both for its simplicity and its versatility, and hence the first solution to consider in intractable situations. For a psuedo-code of the algorithm, see Algorithm 1.

Although the Metropolis-Hastings algorithm sounds marvelous, it has a problem: you have to make sure to optimize the step-size that you use in the algorithm, so that your acceptance rate is good. Luckily, there exists an algorithm that fixes this complication: the Adaptive Metropolis-Hastings Algorithm, proposed by Haario, et al. in their *An Adaptive Metropolis Algorithm* [Haario, et al. \(2001\)](#). In their paper, they propose an algorithm that streamlines the task of setting the step-size, compared to the arduous process outlined in the original algorithm. For a pseudo-code of the algorithm, see Algorithm 2. Note that not only is the Adaptive Metropolis-Hastings algorithm useful in making sure that the Metropolis-Hastings algorithm has good parameters, in order to have remarkable acceptance rates, it is also useful when you have high-dimensional target distributions or when you have correlated dimensions with different variances.

Algorithm 1 The Metropolis-Hastings algorithm

- 1: Draw $X^0 \sim \mu$ where μ is the initial condition.
 - 2: Set $t \leftarrow 0$.
 - 3: **repeat**
 - 4: Draw $Y^t \sim q(y^t|x^t)$.
 - 5: Compute $\alpha = \min \left\{ \frac{\pi(y^t) q(x^t|y^t)}{\pi(x^t) q(y^t|x^t)}, 1 \right\}$
 - 6: Draw $U \sim \mathcal{U}[0, 1]$.
 - 7: **if** $U \leq \alpha$ **then**
 - 8: Set $X^{t+1} \leftarrow Y^t$.
 - 9: **else**
 - 10: Set $X^{t+1} \leftarrow X^t$.
 - 11: **end if**
 - 12: Set $t \leftarrow t + 1$.
 - 13: **until** Sufficiently many samples have been produced.
-

Algorithm 2 Adaptive Metropolis-Hastings algorithm

- Draw $X^0 \sim \mu$ where μ is the initial condition.
- Set θ^0 to an arbitrary valid value.
- Set $t \leftarrow 0$.
- repeat**
- Compute $\theta^t = \gamma^t(\theta^0, X^0, \dots, X^{t-1})$ where γ^t is a transformation that update the parameters based on past samples.
- Draw X^{t+1} using the proposal $q(\cdot|x^t, \theta^t)$ with the Metropolis-Hastings rule.
- Set $t \leftarrow t + 1$.
- until** The accept rate of the last K iterations are close to 0.234.
- Set $\theta \leftarrow \theta^t$.
- Perform Algorithm 1 with proposal $q(\cdot|x) = q(\cdot|x, \theta)$.
-

3. Discussion & Tips

REFERENCES

- Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953).
“Equations of state calculations by fast computing machines.” *J. Chem. Phys.*,
21(6): 10871092.
- Hastings, W. (1970). “Monte Carlo sampling methods using Markov chains and their
application.” *Biometrika*, 57: 97109.
- Barker, A. (1965). “Monte Carlo calculations of the radial distribution functions for a
proton electron plasma.” *Aust. J. Physics*, 18: 119133.
- Haario, H.; Saksman, E.; Tamminen, J. (2001). “An adaptive Metropolis algorithm.”
Bernoulli, 7(2): 223242.

A. Installation

B. Issues & Contributions

The development of `pipeline` is being coordinated on GitHub at <http://github.com/mussles/pipeline> and contributions are welcome. If you encounter any problems with the code, please report them at <http://github.com/mussles/pipeline/issues> and consider contributing a patch.

C. Online Documentation

To learn more about how to use `pipeline` in practice, it is best to check out the documentation on the website <https://mussles.github.io/pipeline>. This page includes the API documentation and many examples of possible work flows.