Sampling with Markov Chain Monte Carlo

A look on efficiency and why it is important

Quang Ha

April 18, 2017

Table of contents

Introduction

What is Markov Chain Monte Carlo?

- **Monte Carlo** is broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. Wiki.
- **Markov chain** are random variables having the property that, given the present, the future is conditionally independent of the past. Wiki.

$$P(X_{n+1}|X_n, X_{n-1}, \cdots, X_0) = P(X_{n+1}|X_n)$$

- **Markov Chain Monte Carlo** is a class of algorithms for sampling from a probability distribution based on constructing a Markov chain that has the desired distribution as its equilibrium distribution. Wiki.
- Born in the early 1950s in Los Alamos, NM. First implemented on MANIAC computer, built under direction of Nicholas Metropolis.

Robert et al, 2011.

How MCMC works - The landscape perspective

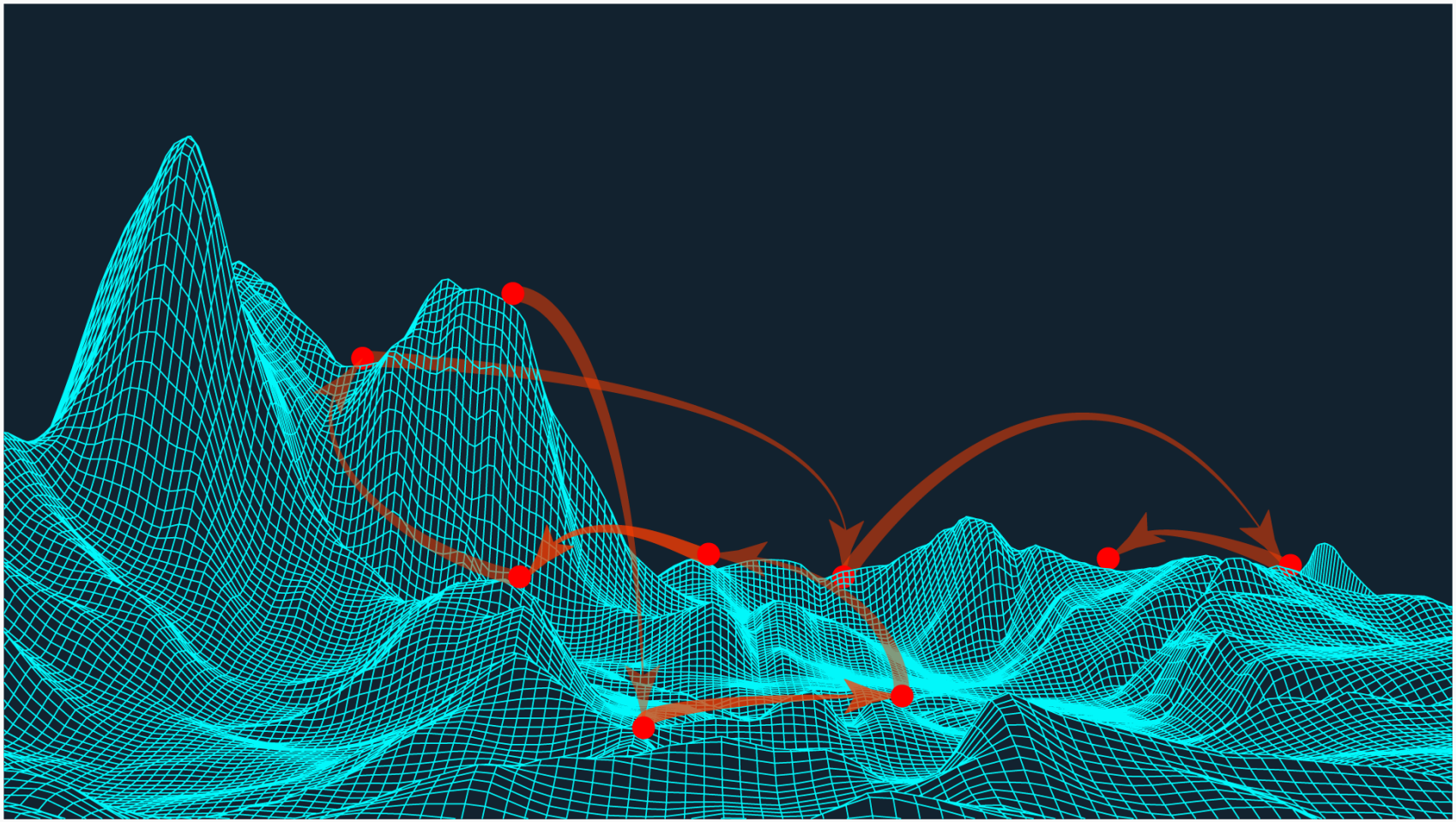


Figure 1: Exploring the landscape with random stepping.

How MCMC works - More explicit explanation

"By collecting a 'large' number of pebbles that is likely to come from the mountain, we can 'approximately' construct the landscape."

Davidson-Pilon

- Start at current position.
- **Propose moving to a new position** - somewhere near you!
- Accept/Reject the new position and ask how likely the pebble is from the mountain.
  - If it is: Move to the new position.
  - If not: Stay where you are.
- **After a large number of iterations**, return all *accepted* positions.

Reasons to use MCMC

- In most cases, MCMC are used for 'black box' system - can only obtained the output given an input.
- Preferred method in high-dimension - compared to explicit formula in $N-$dimensions.
- Sampling is easy for a uniform distribution in 1D, but not for more complicated pdfs.
- Also in high dimensions, method like *rejection-acceptance* will be very inefficient - most of the times we will miss!

Major challenges

Challenges for MCMC

- It is important to explore the entire 'landscape' - there can be useful information within the 'black box'.
- The computational cost for getting an output of the 'black box' is typically huge - hence rejection rate needs to be minimised!
- Sequences of MCMC samples are based on the assumption that the samples are derived from the pdf of interest, and theory guarantees this condition as the number of iterations approaches *infinity*.
- Unfortunately, no universal threshold exists across all problems! - *Fonnesbeck, 2014.*

Nostalgic example: HW problem...

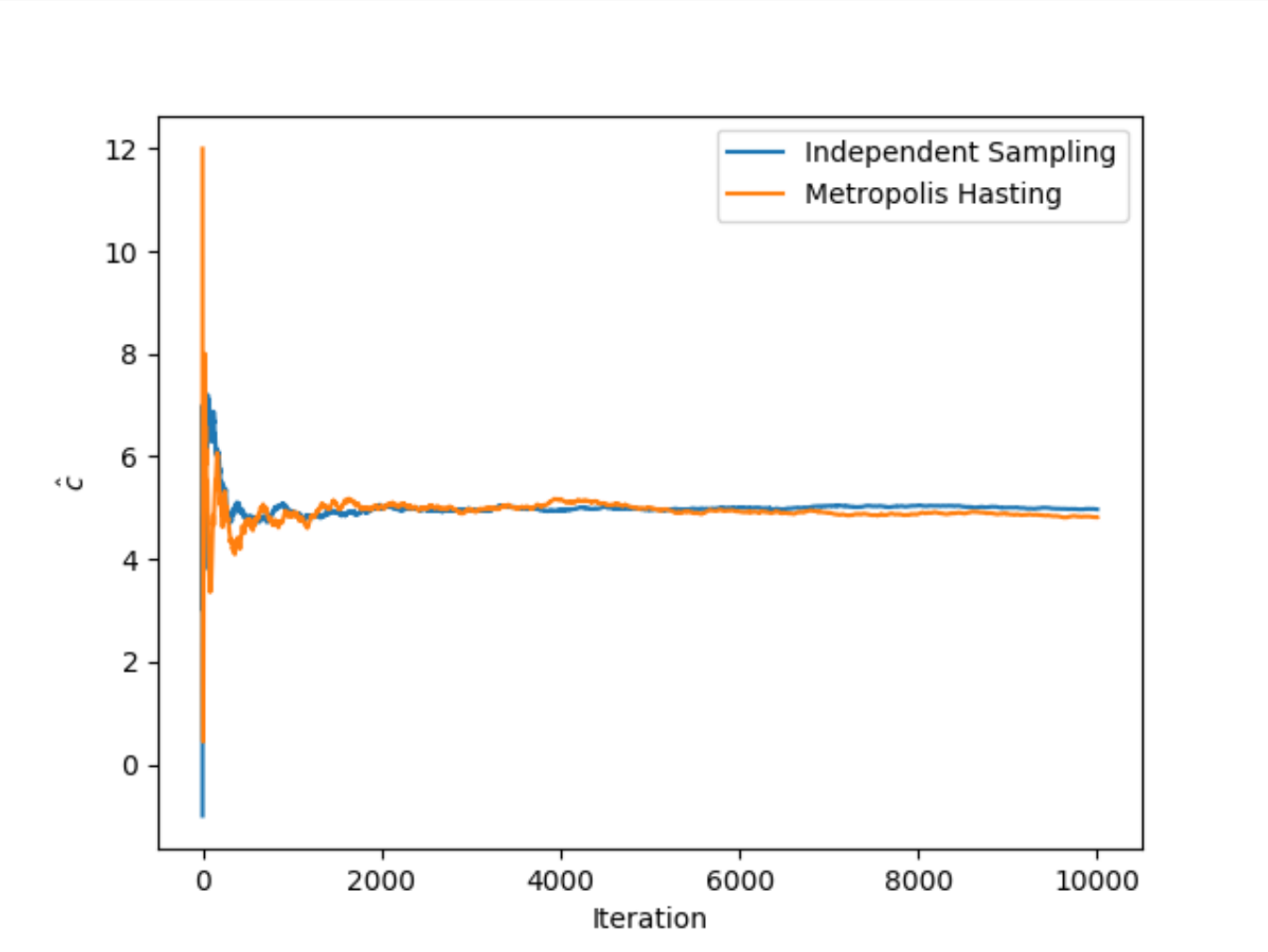


Figure 2: Example taken from homework 6 - SE/ME 714.

Rapid 'mixing'

- 'Mixing' indicates how well MCMC samples move through the entire 'landscape' of the desired pdf.
- In ideal Monte Carlo methods, our samples are expected to be independent. However, this is not the case for Markov chains.
- Poor mixing is caused by *inappropiate proposed movings* or *highly-correlated variables.*
- Frequently, lack of convergence is caused by poor mixing.

Poor mixing examples

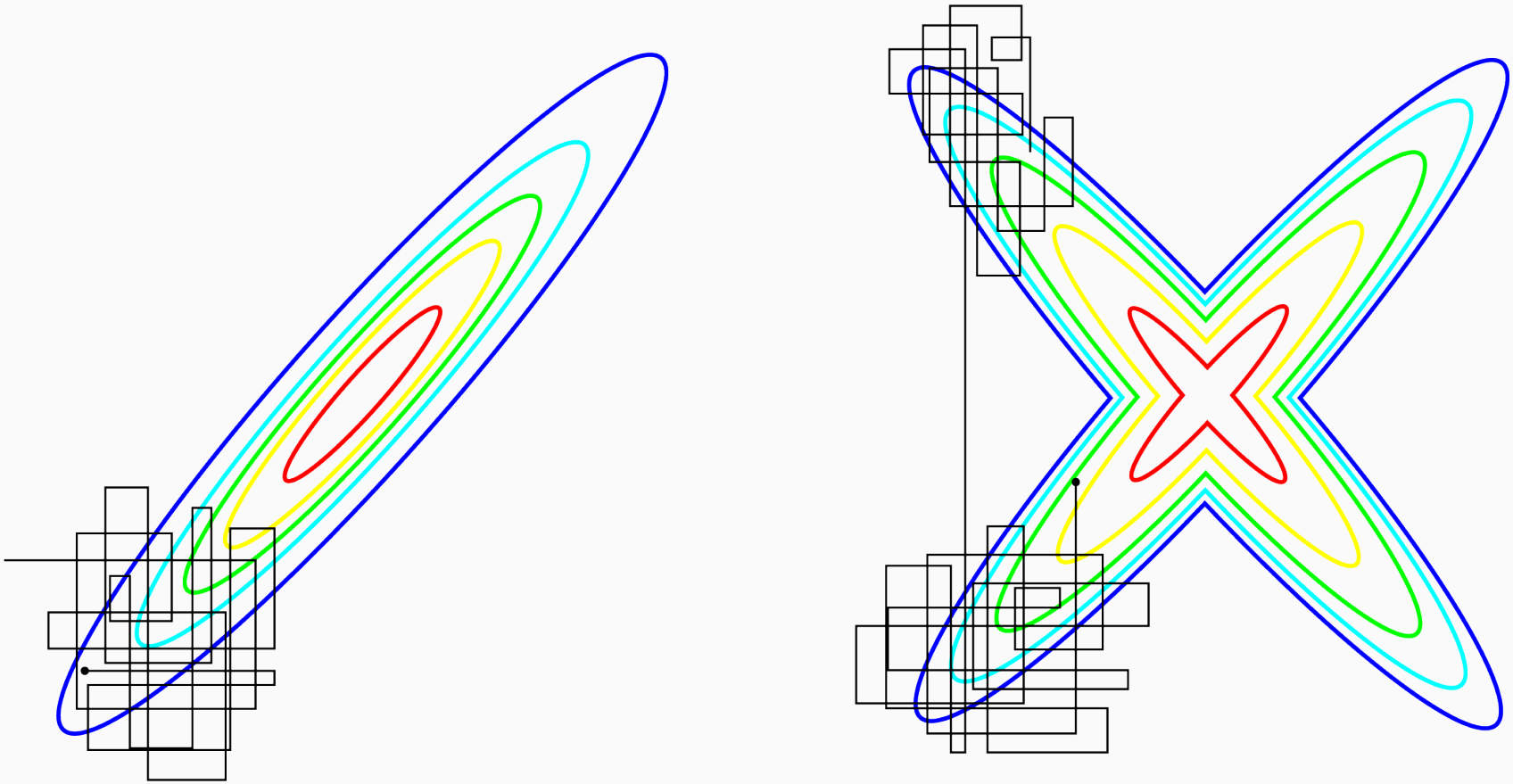


Figure 3: Example of poor mixing.

Strategies for proposing moves

Propose moving to a new position

- Designing a good strategy for moving/stepping is hard, and it's depends on the desired pdf.
- The proposal move is important to design a rapid mixing MCMC. It is normally built using known 'randomness' algorithms.
- Known strategies include:
  - **Single-component update**: Gibbs sampling, Component-wise Metropolis Hasting, etc.
  - **Adaptive direction sampling**: Hit-and-run, Adaptive direction, etc.
  - **Sample from an 'easier' distribution**: Importance sampling, etc.
  - **Re-sampling from the previous samples**
  - **Adopt physical system dynamic**: Hybrid/Hamiltonian MCMC, etc.

  ... and many more...

Single-component update

- Not many multi-variate distribution to sample from, but we do have several 1D pdfs...
- Main drawback: very poor mixing if variables are highly correlated.
- Solution: re-parameterise into independent variables.
- Further problem(!): not as easy for more complicated problems!

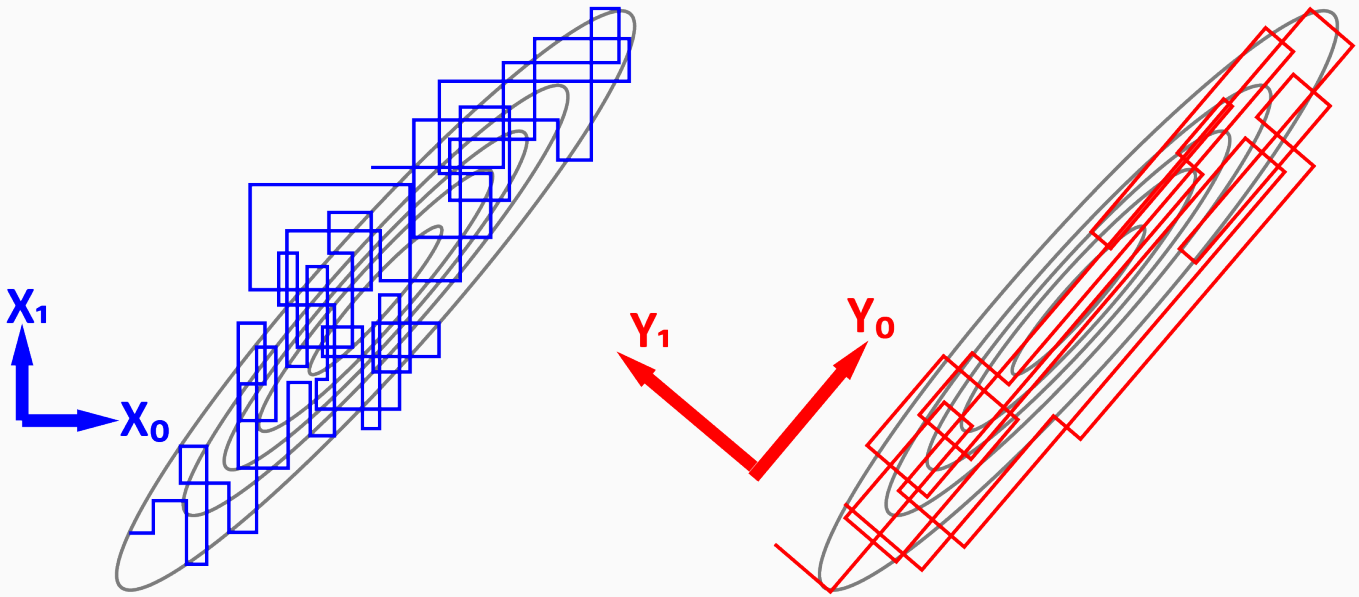


Figure 4: Re-parameterisation for single-component update.

Adaptive direction sampling

- Upgrade from single-component update: choose *direction* and *step size* at random!
- Pros: improve mixing when choosing 'good' directions.
- Cons: choosing 'good' directions.

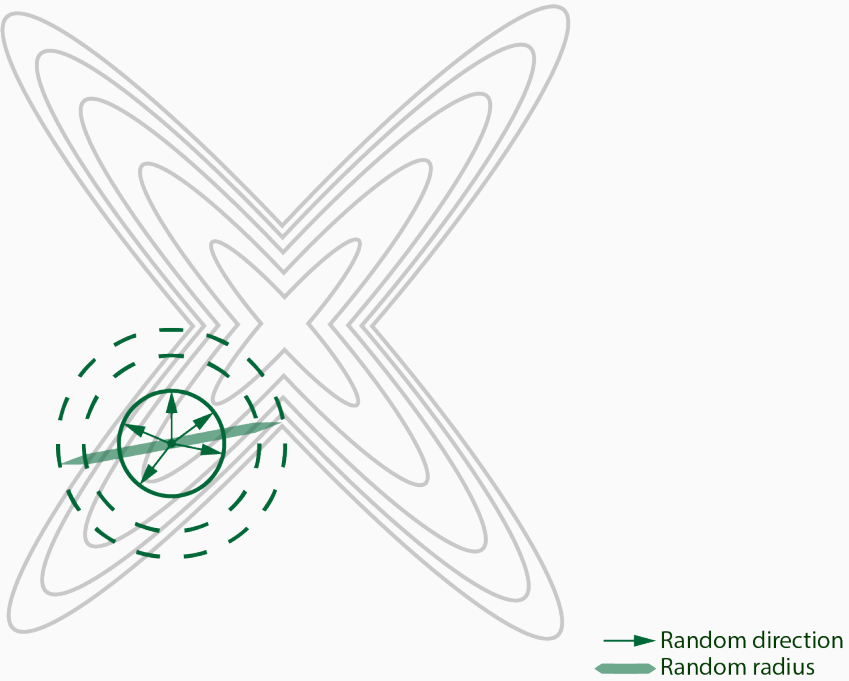


Figure 5: Adaptive direction sampling

Resampling from previous samples

- Re-sampling from previous steps to allow easier movement.
- Arrive at a distribution from re-sampled points that is close to the desired stationary distribution.

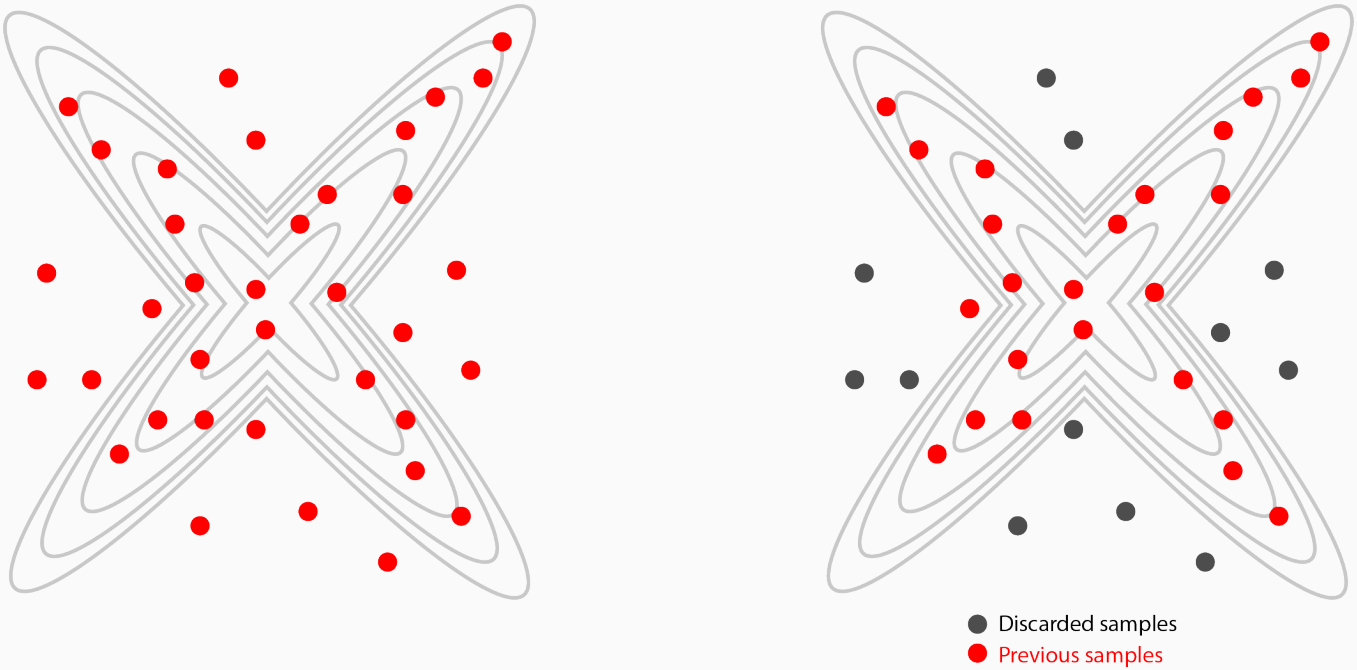


Figure 6: Re-sampling from the past steps.

Sampling from 'easier' distribution

- Use samples from a different distributions (usually ones we are familiar with.)
- Then apply weights to correct over-generation and under-generation.
- Cons: Does not work well in high dimensions.

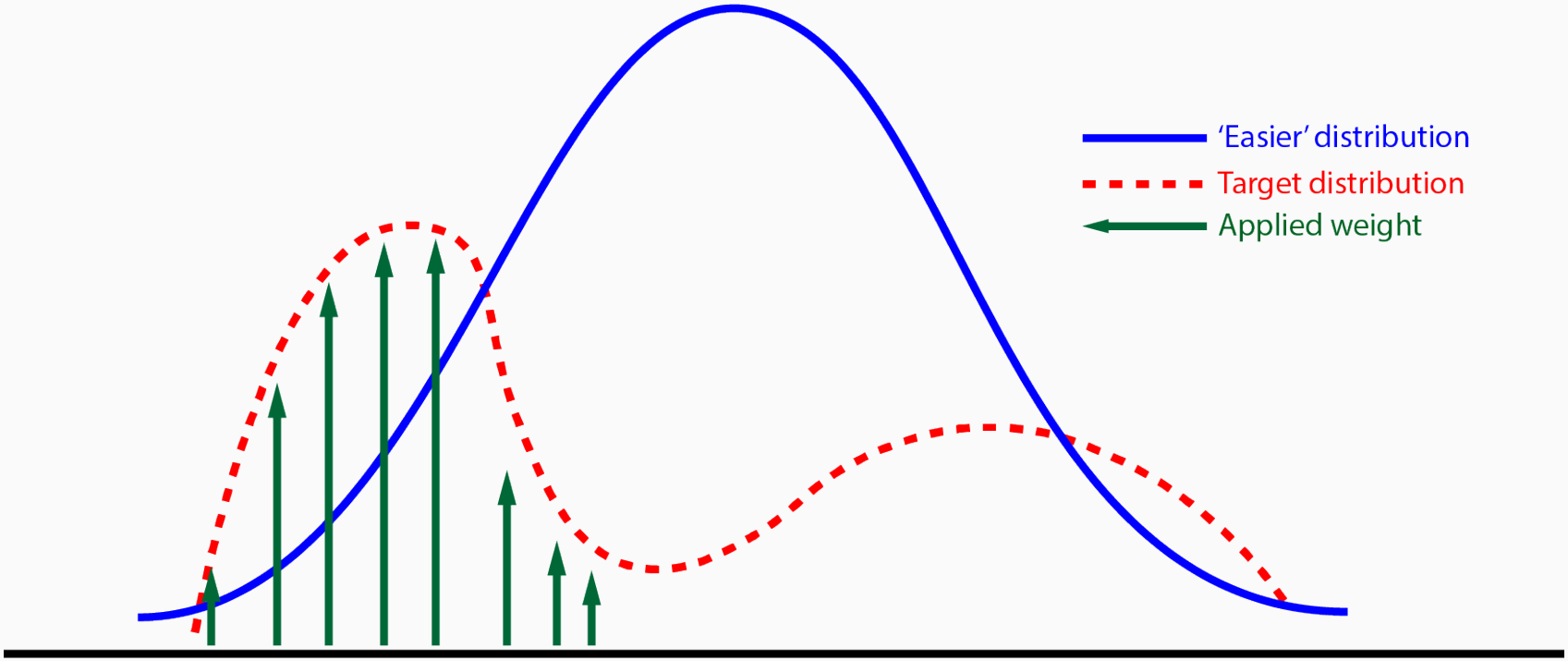
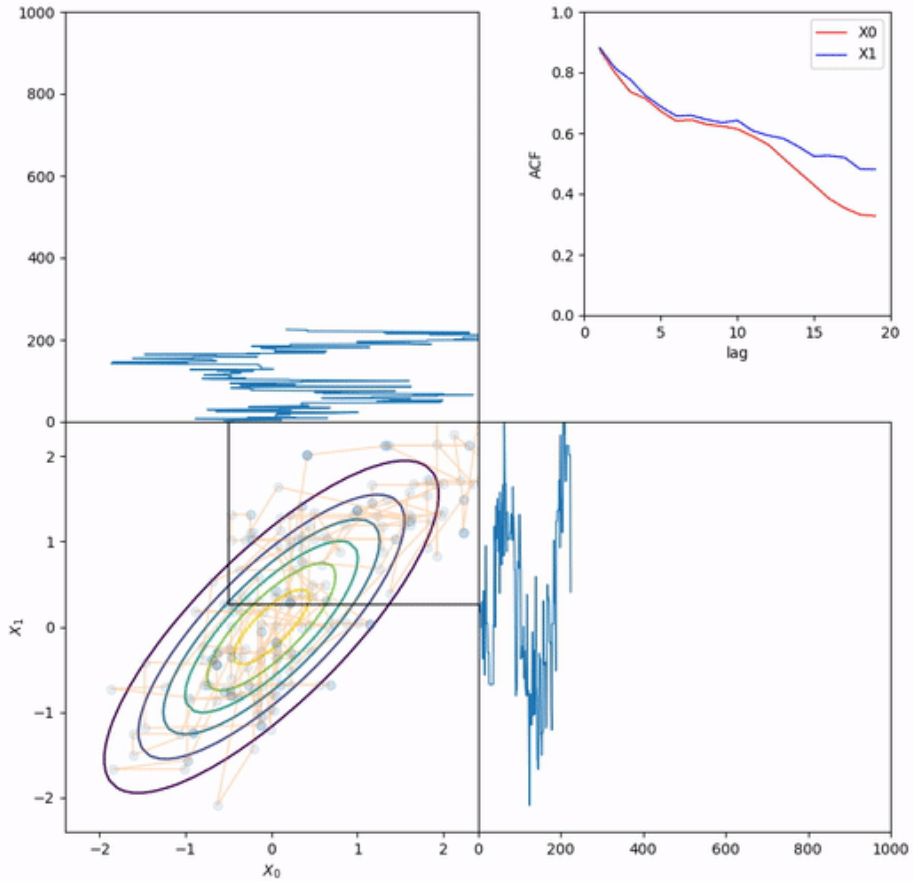


Figure 7: Transform the target distribution.

(E) Visualising MCMC

Code available: `https://github.com/quang-ha/mcmc-sampling`



Summary

Summary

- It is important for MCMC sampling strategies to achieve rapid mixing, which improves convergence and reduce overall computational cost.
- There are several strategies to improve the efficiency of MCMC sampling. The choice of strategy needs careful consideration based on the problem.
- There are no global method to check for convergence in MCMC - different attempts have been proposed such as checking autocorrelation function, Gelman-Rubin etc.