Monte Carlo Estimation

One reason to sample a distribution is to approximate expected values under that distribution…

Expected value of function $f(x)$ w.r.t. distribution $p(x)$ given by,

$$\mathbb{E}_p[f(x)] = \int p(x) f(x)\, dx \equiv \mu$$

- Doesn't always have a closed-form for arbitrary functions
- Suppose we have iid samples: $\{x_i\}_{i=1}^N \sim p(x)$
- *Monte Carlo* estimate of expected value,

$$\hat{\mu}_N = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \approx \mathbb{E}_p[f(x)]$$

Samples must be independent!

Markov chain Monte Carlo methods

- The approximations of expectation that we have looked at so far have assumed that the samples are independent draws.
- This sounds good, but in high dimensions, we do not know how to get good independent samples from the distribution.
- MCMC methods drop this requirement.
- Basic intuition
  - If you have finally found a region of high probability, stick around for a bit, enjoy yourself, grab some more samples.

Markov chain Monte Carlo methods

- Samples are conditioned on the previous one (this is the Markov chain).
- MCMC is often a good hammer for complex, high dimensional, problems.
- Main downside is that it is not “plug-and-play”
  - Doing well requires taking advantage to the structure of your problem
  - MCMC tends to be expensive (but take heart---there may not be any other solution, and at least your problem is being solved).
  - If there are faster solutions, you can incorporate that (and MCMC becomes a way to improve/select these good guesses).

Metropolis Algorithm

We want samples $z^{(1)}, z^{(2)}, \ldots.$

Again, write $p(z) = \tilde{p}(z)/Z$

Assume that $q\left(z \middle| z^{(prev)}\right)$ can be sampled easily

Also assume that $q(\quad)$ is symmetric, i.e., $q\left(z_A \middle| z_B\right) = q\left(z_B \middle| z_A\right)$

For example, $q\left(z \middle| z^{(prev)}\right) \sim \mathbb{N}\left(z; z^{(prev)}, \sigma^2\right)$

Metropolis Algorithm

While not_bored

{

Sample $q\left(z \middle| z^{(prev)}\right)$

Accept with probability $A\left(z, z^{(prev)}\right) = \min\left(1, \frac{\tilde{p}(z)}{\tilde{p}\left(z^{(prev)}\right)}\right)$

If accept, emit z, otherwise, emit $z^{(prev)}$.

}

Always emit one or the other

If things get better, always accept. If they get worse, sometimes accept.

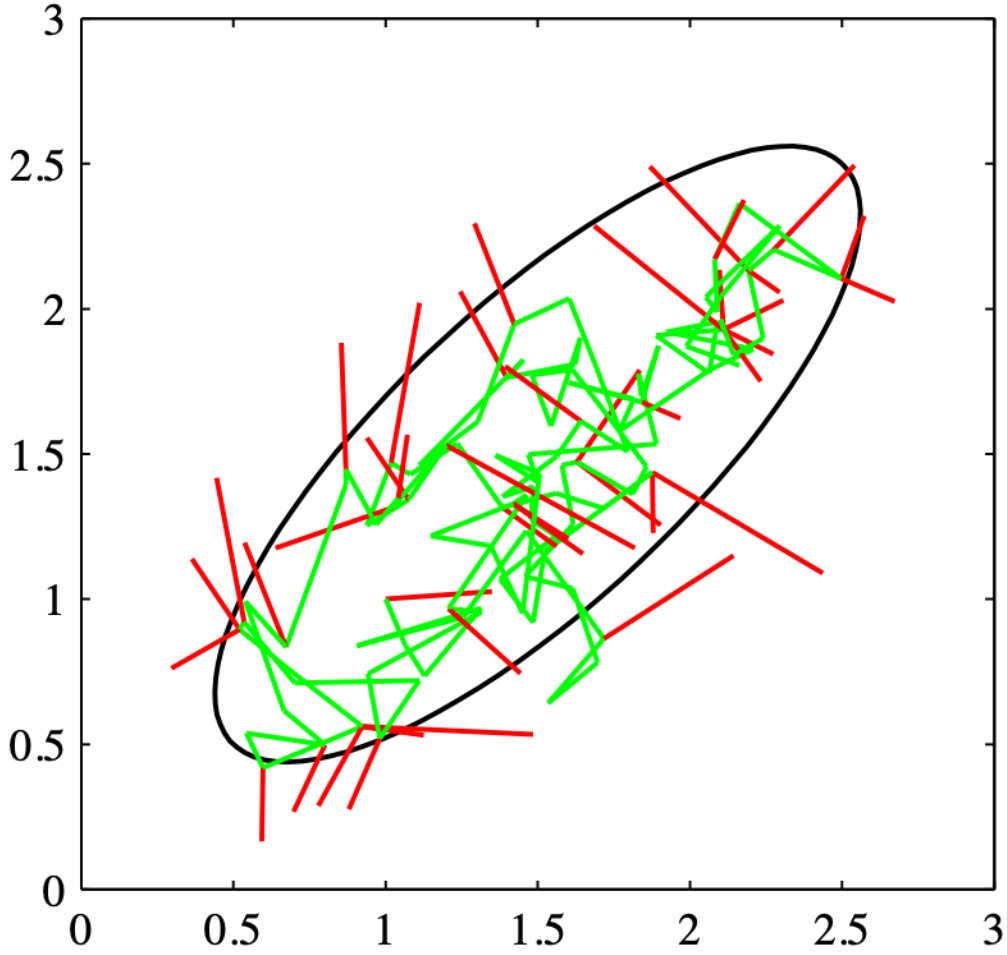
Metropolis Algorithm

Note that

$$A\left(z, z^{(prev)}\right) = \min\left(1, \frac{\tilde{p}\left(z\right)}{\tilde{p}\left(z^{(prev)}\right)}\right) = \min\left(1, \frac{p\left(z\right)}{p\left(z^{(prev)}\right)}\right)$$

So we do not need to normalize $p\left(z\right)$

Metropolis Example



Green follows accepted proposals
Red are rejected moves.

Markov chain view

Denote an initial probability distribution by $p\left(z^{(1)}\right)$

Define transition probabilities by:

$$T\left(z^{(prev)}, z\right) = p\left(z \middle| z^{(prev)}\right) \qquad \text{(a probability distribution)}$$

T can change over time, but for now, assume that it
it is always the same (homogeneous chain)

A given chain evolves from a sample of $p\left(z^{(1)}\right)$, and is
an instance from an essemble of chains.

Markov Chain Monte Carlo (MCMC)

- Stochastic 1st order Markov process with transition kernel:

$$T(z^{(t)} \mid z^{(t-1)})$$

$$\ldots \longrightarrow z^{(t-1)} \longrightarrow z^{(t)} \longrightarrow z^{(t+1)} \longrightarrow \ldots$$

- Each $z^{(t)}$ full N-dimensional state vector
- MCMC samples $\ldots, z^{(t-1)}, z^{(t)}, z^{(t+1)}, \ldots$ **not independent**
- New superscript notation indicates dependence:

$$\{z^{(\ell)}\}_{\ell=1}^{L} \qquad \{z^{(t)}\}_{t=1}^{T}$$

Independent **Dependent**

Key Question: How many MCMC samples T are needed to draw L independent samples from p(x)?

Stationary Markov chains

- Recall that our goal is to have our Markov chain emit samples from our **target distribution** *p(z)*.
- This implies that the distribution being sampled at time *t+1* would be the same as that of time *t* (**stationary**).
- If our stationary (target) distribution is *p()*, then if we imagine an ensemble of chains, they are in each state with (long-run) probability *p()*.
  - On average, a switch from s1 to s2 happens as often as going from s2 to s1, otherwise, the percentage of states would not be stable.

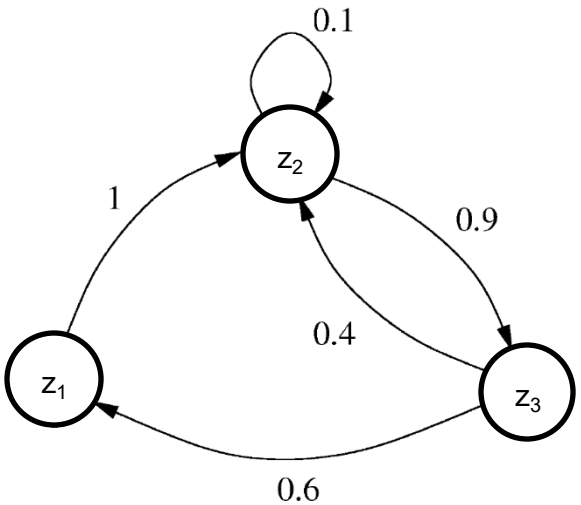
Markov Chain Monte Carlo (MCMC)

- Stochastic 1st order Markov process with transition kernel:

$$T(z^{(t)} \mid z^{(t-1)})$$

$$\ldots \longrightarrow z^{(t-1)} \longrightarrow z^{(t)} \longrightarrow z^{(t+1)} \longrightarrow \ldots$$

E.g. Let, $T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}$



[Source: Andrieu et al.]

- Initial state distribution: $\mu(z^{(1)}) = (0.5, 0.2, 0.3)$
- Repeated transitions converge to target

$$\mu(z^{(1)}) \cdot T \cdot T \cdot \ldots T = (0.2, 0.4, 0.4) = p(z)$$

True for <u>any</u> initial state distribution

How can we formalize this?

Ergodic chains

- Different starting probabilities will give different chains
- We want our chains to converge (in the limit) to the same stationary state, regardless of starting distribution.
- Such chains are called ergodic, and the common stationary state is called the equilibrium state.
- Ergodic chains have a unique equilibrium.

When do our chains converge?

- Important theorem tells us that for finite state spaces* our chains converge to equilibrium under two relatively weak conditions.
  - (1) Irreducible
    - We can get from any state to any other state
  - (2) Aperiodic
    - The chain does not get trapped in cycles
- These are true for detailed balance (there exists a stationary state) with T>0 (you can get there).
  - Detailed balance is sufficient, but not necessary for convergence—it is a stronger property than (1) & (2)

*Infinite or uncountable state spaces introduces additional complexities, but the main thrust is similar.

MCMC so far

- Under reasonable conditions (ergodicity) ensembles of chains over discretized states converge to an equilibrium state (stationary distribution)

- Easiest way to prove (or check) that this is the case is to show **detailed balance** and use T>0 (sufficient but not necessary)

- There is a nice analogy with powers of stochastic matrices, which converge to an operator based on the largest magnitude eigenvector (with |eigenvalue|=1)

- In theory, to use MCMC for sampling a distribution, we simply need to ensure that our target distribution is the equilibrium state.

- In practice we do not know even know if we have visited the best place yet. (The ensemble metaphor runs into trouble if you have a small number of chains compared to the number of states).

MCMC Theory vs. Practice

- The time it takes to get reasonably close to equilibrium (where samples come from the target distribution) is called "burn in" time.
  - I.E., how long does it take to forget the starting state.
  - There is no general way to know when this has occurred.

- The average time it takes to visit a state is called "hit time".

- What if we really want independent samples?
  - In theory we can take every N^{th} sample (some theories about how long to wait exist, but it depends on the algorithm and distribution).

MCMC for ML in practice

- We use MCMC for machine learning problems with very complex distributions over high dimensional spaces.

- Variables can be either discrete or continuous (often both)

- Despite the gloomy worst case scenario, MCMC is often a good way to find good solutions (either by MAP or integration).
  - Key reason is that there is generally structure in our distributions.
  - We need to exploit this knowledge in our proposal distributions.
  - Instead of getting hung up about whether you actually have convergence
    - Enjoy that fact that what you are doing is principled and can improve any answer (with respect to your model) that you can get by other means
      - Your model should be able to tell you which proposed solution are good.

Beyond the Metropolis Method

Metropolis requires the proposal to be symmetric,

$$q(z' \mid z) = q(z \mid z')$$

This often results in a chain that takes a long time to converge to a stationary distribution (long burn in time)

Example The most common proposal (Gaussian),

$$q(z' \mid z) = \mathcal{N}(z' \mid z, \sigma^2 I)$$

exhibits random walk dynamics that are inefficient

Metropolis-Hastings relaxes this symmetry requirement...

Metropolis-Hastings MCMC method

While not_bored

{

Sample $q\left(z \middle| z^{(prev)}\right)$

Accept with probability $A\left(z, z^{(prev)}\right) = \min\left(1, \frac{\tilde{p}\left(z\right) q\left(z^{(prev)} \middle| z\right)}{\tilde{p}\left(z^{(prev)}\right) q\left(z \middle| z^{(prev)}\right)}\right)$

If accept, emit z, otherwise, emit $z^{(prev)}$.

}

Metropolis-Hastings comments

- Again it does not matter if we use unnormalized probabilities in the M-H acceptance ratio A(z,z')

- It should be clear that the Metropolis method (where q() is symmetric) is a special case of M-H

- q(z'|z) can be anything, but you need to specify the reverse move q(z|z'), which can be tricky

MCMC So Far…

Metropolis Algorithm

- Sample RV from proposal $z \sim q(z \mid z^{(\text{prev})})$
- Proposal must be *symmetric* $q(z \mid z^{(\text{prev})}) = q(z^{(\text{prev})} \mid z)$
- Accept with probability $\min\left\{1, \widetilde{p}(z) \div \widetilde{p}(z^{(\text{prev})})\right\}$

Metropolis-Hastings Algorithm

- Proposal does not have to be symmetric
- Accept with probability

$$\min\left\{1, \frac{\widetilde{p}(z)q(z^{(\text{prev})} \mid z)}{\widetilde{p}(z^{(\text{prev})})q(z \mid z^{(\text{prev})})}\right\}$$

Both methods require choosing proposal, which can be hard

Combined samplers

Different samplers fail in different ways, so combine them…

1. Initialise $x^{(0)}$.
2. For $i = 0$ to $N - 1$
   - Sample $u \sim \mathcal{U}_{[0,1]}$.
   - If $u < \nu$

     Apply the MH algorithm with a global proposal.
   - else

     Apply the MH algorithm with a random walk proposal.

…can also combine with Gibbs proposals

Mixing MCMC Kernels

Can do this more generally….

Consider a set of MCMC kernels $T_1, T_2, \ldots, T_K$ all having target distribution p(x) then the mixture:

$$T = \sum_{k=1}^{K} \pi_k T_k$$

Mixing weights

Is a valid MCMC kernel with target distribution p(x)

Mixture MCMC Transition kernel given by:

1. Sample $k \sim \pi$
2. Sample $x^{(t+1)} \sim T_k(x \mid x^{(t)})$

Inference (and related) Tasks

- Simulation: $x \sim p(x) = \frac{1}{Z} f(x)$
- Compute expectations: $\mathbb{E}[\phi(x)] = \int p(x)\phi(x)\, dx$
- Optimization: $x^* = \arg\max_x f(x)$
- Compute normalizer: $Z = \int f(x)\, dx$

Inference (and related) Tasks

- Simulation: $x \sim p(x) = \frac{1}{Z} f(x)$
- Compute expectations: $\mathbb{E}[\phi(x)] = \int p(x)\phi(x)\, dx$
- Optimization: $x^* = \arg\max_x f(x)$
- Compute normalizer: $Z = \int f(x)\, dx$

Simulated Annealing

- Analogy with physical systems
- Relevant for optimization (not integration)
- Powers of probability distributions emphasize the peaks
- If we are looking for a maximum within a lot of distracting peaks, this can help.

Simulated Annealing

- Define a temperature T, and a cooling schedule (black magic part)
- Lower temperatures correspond to emphasized maximal peaks.
  - Hence we exponentiate by (1/T).
- The terminology makes sense because the number of states accessible to a physical system decreases with temperature.

Simulated Annealing

1. Initialise $x^{(0)}$ and set $T_0 = 1$.
2. For $i = 0$ to $N - 1$
   - Sample $u \sim \mathcal{U}_{[0,1]}$.
   - Sample $x^\star \sim q(x^\star | x^{(i)})$.
   - If $u < \mathcal{A}(x^{(i)}, x^\star) = \min\left\{1, \frac{p^{\frac{1}{T_i}}(x^\star) q(x^{(i)}|x^\star)}{p^{\frac{1}{T_i}}(x^{(i)}) q(x^\star|x^{(i)})}\right\}$

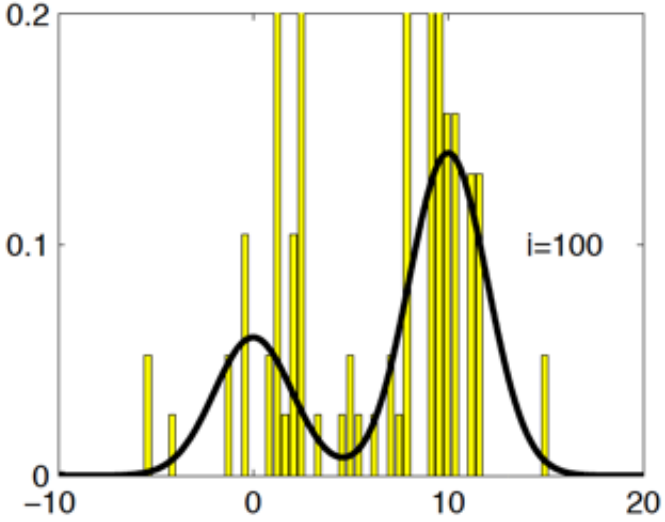
     $$x^{(i+1)} = x^\star$$

     else

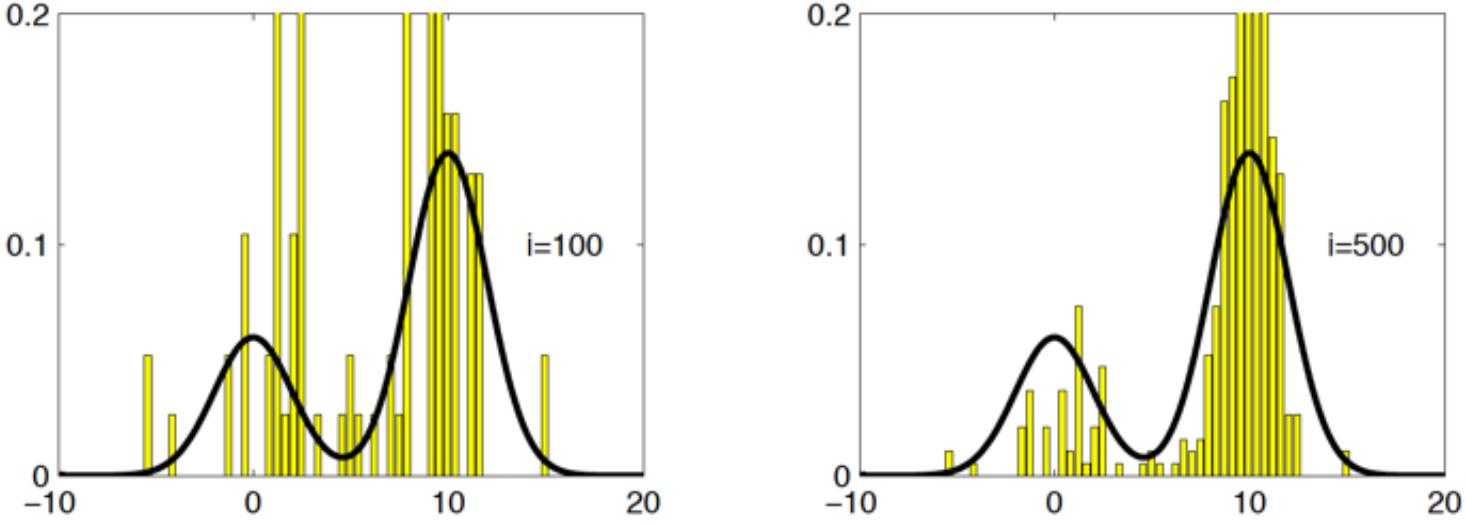
     $$x^{(i+1)} = x^{(i)}$$
   - Set T_{i+1} according to a chosen cooling schedule.

Basically M-H but we are *annealing* target distribution with temperature T

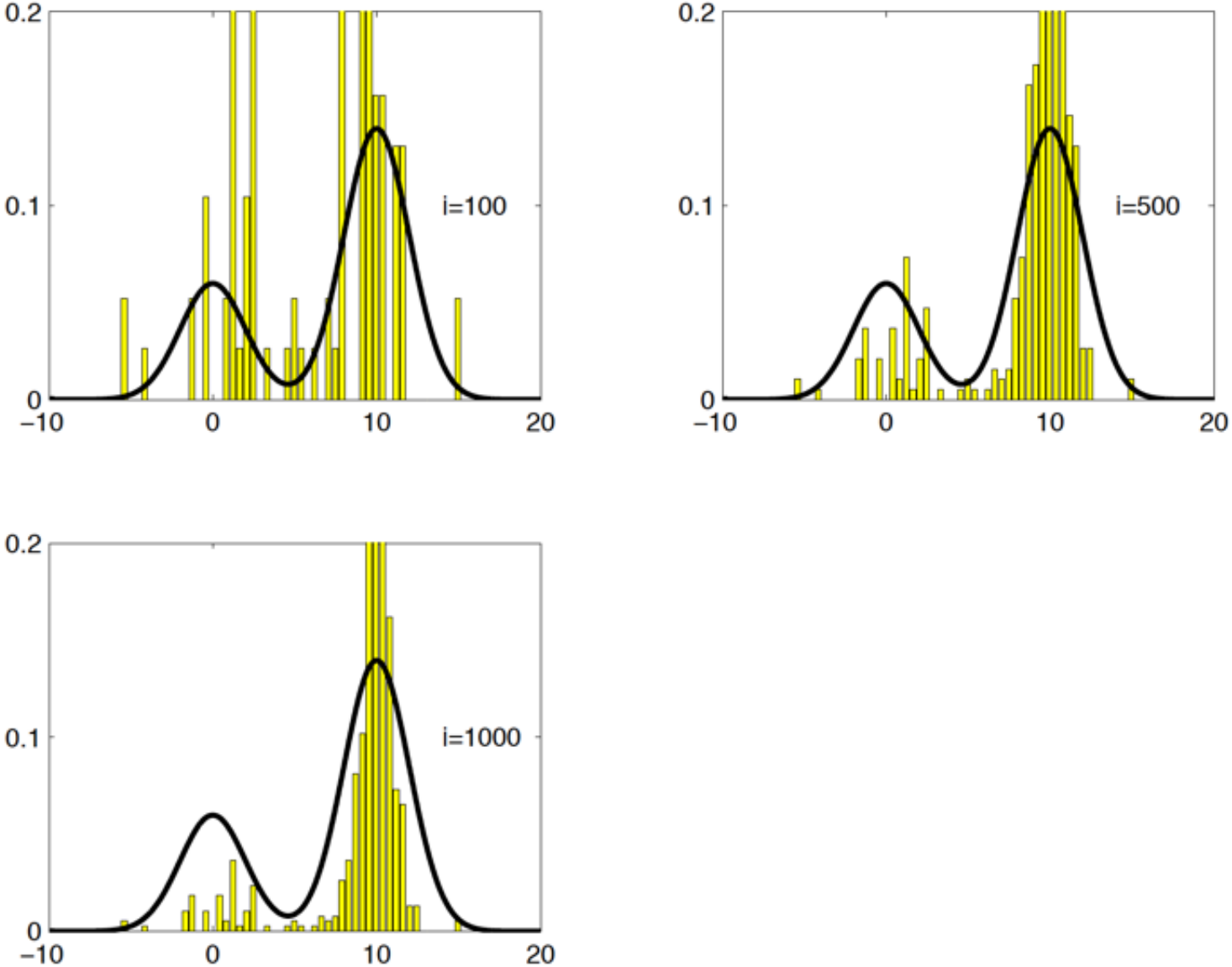
(From Andrieu et al)



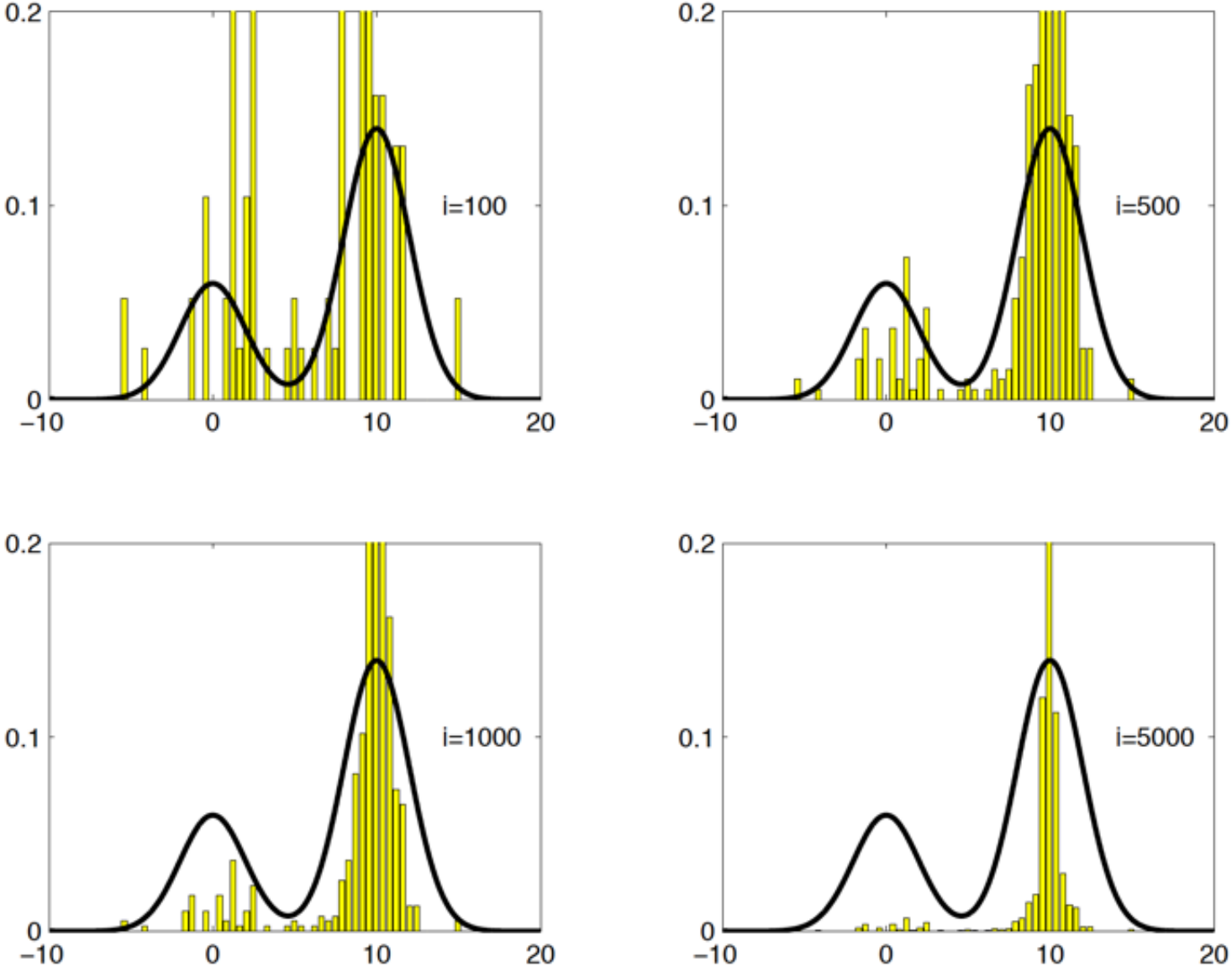
(From Andrieu et al)



(From Andrieu et al)



(From Andrieu et al)



(From Andrieu et al)

Simulated Annealing

Let *annealing distribution* at temp τ be given by:

$$p_\tau(x) \propto (f(x))^{1/\tau}$$

As $\tau \to 0$ we have:

$$\lim_{\tau \to 0} p_\tau(x) = \delta(x^*) \qquad \text{where} \qquad x^* = \arg\max_x f(x)$$

Simulated Annealing (SA) for Global Optimization:
Annealing schedule $\tau_0 \geq \ldots \geq \tau_t \geq \ldots \geq 0$

1. Sample $x^{(t)}$ from MCMC kernel T_t with target $p_{\tau_t}(x)$
2. Set τ_{t+1} according to annealing schedule

SA for Convergence: $\tau_0 \geq \ldots \geq 1$ Final temperature = 1

MCMC Summary

- Markov chain induced by MCMC transition kernel T(z,z')
- Converges to stationary distribution iff chain is **ergodic**
  - Chain is ergodic if it is **irreducible** (can get from any z to any z') and **aperiodic** (doesn't get trapped in cycles)
- Easier to prove **detailed balance,** which implies ergodicity

$$p(z)T(z, z') = p(z')T(z', z)$$

- Metropolis algorithm samples from symmetric proposal q(z'|z) and accepts sample z' with probability,

$$A = \min\left(1, \frac{\widetilde{p}(z')}{\widetilde{p}(z)}\right)$$

MCMC Summary

- Metropolis-Hastings allows non-symmetric proposal q(z'|z) and accepts sample z' with probability,

$$A = \min\left(1, \frac{\widetilde{p}(z')}{\widetilde{p}(z)} \frac{q(z \mid z')}{q(z' \mid z)}\right)$$

- Gibbs sampler on random vector $z = (z_1, \ldots, z_d)^T$ successively samples from *complete conditionals*,

$$\begin{aligned}
z_1^{\text{new}} &\sim p(z_1 \mid z_2^{\text{old}}, \ldots, z_d^{\text{old}}) \\
z_2^{\text{new}} &\sim p(z_2 \mid z_1^{\text{new}}, z_3^{\text{old}}, \ldots, z_d^{\text{old}}) \\
&\ldots \\
z_d^{\text{new}} &\sim p(z_d \mid z_1^{\text{new}}, \ldots, z_{d-1}^{\text{new}})
\end{aligned}$$

- Gibbs is instance of M-H that *always accepts*

MCMC Summary

- Simulated annealing adjusts target distribution at each stage with temperature T

$$(p(z))^{\frac{1}{T_i}}$$

- For decreasing temperatures $\lim T_i \to 0$ support of target approaches set of global maximizers
  - Convenient to use for global maximization
  - Can prove that this will find the global maximum in the limit (need to wait for the heat death of the universe, however…)
- For increasing temp ending at $\lim T_i \to 1$ approaches $p(x)$
  - Helps avoid getting stuck in local optima