Markov chain Monte Carlo

Construct a biased random walk that explores target distribution $P^\star(x)$. Markov steps, $x_t \sim T(x_t \leftarrow x_{t-1})$.

MCMC gives approximate, correlated samples from $P^\star(x)$. 

CS242: Lecture 7B Outline

- MCMC & Metropolis-Hastings Algorithms
- Gibbs Sampling for Graphical Models
- Moments, mixing, & convergence diagnostics
Markov Chain Monte Carlo (MCMC)

- **Goal:** Draw samples from some *target distribution*
  \[ p(x) = \frac{1}{Z} p^*(x) \quad \text{(discrete or continuous)} \]

- **Assumption:** Can evaluate up to unknown normalizer

- **Approach:** Design Markov chain (biased random walk) \( T \) which has *target distribution as unique equilibrium distribution*:
  \[ \sum_x T(x' \leftarrow x) p^*(x) = p^*(x') \quad \int_x T(x' \leftarrow x) p^*(x) \, dx = p^*(x') \]

Metropolis-Hastings Method

- **Input:** Target distribution, and user-specified transition distribution \( q(x' \mid x) \)
- **Approach:** Transform transition distribution so it has target as equilibrium
Detailed Balance

A sufficient condition for Markov Chain to have desired target distribution.

Probability of reaching state $x$ via some path, and then jumping to $x'$.

$$T(x' \leftarrow x)p^*(x) = p^*(x')T(x \leftarrow x')$$

Implications of detailed balance condition:

If $p^*(x) = p^*(x')$, then $T(x' \leftarrow x) = T(x \leftarrow x')$.

If $p^*(x') > p^*(x)$, then
$$\frac{T(x' \leftarrow x)}{T(x \leftarrow x')} = \frac{p^*(x')}{p^*(x)} > 1.$$
Detailed Balance

A sufficient condition for Markov Chain to have desired target distribution.

Detailed balance means $\rightarrow x \rightarrow x'$ and $\rightarrow x' \rightarrow x$ are equally probable:

$$T(x' \leftarrow x)p^*(x) = p^*(x')T(x \leftarrow x')$$

Detailed balance implies target is equilibrium distribution:

$$\sum_x T(x' \leftarrow x)p^*(x) = p^*(x') \left[ \sum_x T(x \leftarrow x') \right] = p^*(x')$$

Detailed balance restricts transition distribution. But we can check equilibrium distribution via isolated state pairs, without explicitly summing over all states.
The Metropolis-Hastings Algorithms

Transforms proposals into correct MCMC methods.

**Metropolis-Hastings (MH) Transition Operator** \( T(x' \leftarrow x) \)

- **Propose:** Sample from user-specified *transition distribution*
  \[ p(\tilde{x}) = q(\tilde{x} \mid x) \]
  assumed tractable

- **Evaluate:** Compute the Metropolis-Hastings *acceptance ratio*
  \[ r(\tilde{x}, x) = \frac{p^*(\tilde{x})q(x \mid \tilde{x})}{p^*(x)q(\tilde{x} \mid x)} \]
  equals one if \( q \) satisfies detailed balance for this target

- **Definite Accept:** If \( r(\tilde{x}, x) \geq 1 \), set \( x' = \tilde{x} \).
- **Possible Reject:** If \( r(\tilde{x}, x) < 1 \), sample auxiliary variable \( u \sim \text{Ber}(r(\tilde{x}, x)) \).
  \[ x' = u\tilde{x} + (1 - u)x. \]
  if reject, state is unchanged

- **Overall acceptance probability:** \( \Pr[x' = \tilde{x}] = \min(1, r(\tilde{x}, x)) \)
The MH Independence Sampler

Transforms independent proposals into correct MCMC methods.

MH Independence Sampler Transition Operator $T(x' \leftarrow x)$

- **Propose:** Sample from user-specified proposal distribution
  
  $$q(\tilde{x} \mid x) = q(\tilde{x})$$
  independent of current state

- **Evaluate:** Compute the Metropolis-Hastings acceptance ratio
  
  $$r(\tilde{x}, x) = \frac{p^*(\tilde{x})q(x)}{p^*(x)q(\tilde{x})} = \frac{w(\tilde{x})}{w(x)}$$
  $$w(x) = \frac{p^*(x)}{q(x)}$$

- **Definite Accept:** If $r(\tilde{x}, x) \geq 1$, set $x' = \tilde{x}$.
- **Possible Reject:** If $r(\tilde{x}, x) < 1$, sample auxiliary variable $u \sim \text{Ber}(r(\tilde{x}, x))$.
  
  $$x' = u\tilde{x} + (1 - u)x.$$ if reject, state is unchanged

- If $q(x) = p(x)$, always accept and recover basic Monte Carlo estimator.
Special Case: The Metropolis Algorithm

Transforms symmetric proposals into correct MCMC methods.

Metropolis Transition Operator

\[ T(x' \leftarrow x) \]

- **Propose:** Sample from user-specified symmetric transition distribution

\[ p(\tilde{x}) = q(\tilde{x} \mid x), \quad q(\tilde{x} \mid x) = q(x \mid \tilde{x}). \]

- **Evaluate:** Compute the Metropolis acceptance ratio

\[ r(\tilde{x}, x) = \frac{p^*(\tilde{x})}{p^*(x)} \]

- **Definite Accept:** If \( r(\tilde{x}, x) \geq 1 \), set \( x' = \tilde{x} \).
- **Possible Reject:** If \( r(\tilde{x}, x) < 1 \), sample auxiliary variable \( u \sim \text{Ber}(r(\tilde{x}, x)) \).

\[ x' = u\tilde{x} + (1 - u)x. \quad \text{if reject, state is unchanged} \]

- Standard greedy search would always, rather than possibly, reject.
Gaussian Random Walk Metropolis

\[ p(x) = \text{Norm}(x \mid 0, 1) \quad q(x' \mid x) = \text{Norm}(x' \mid x, \sigma^2) \]

\text{sigma}(0.1) \quad 99.8\% \text{ accepts}

\text{sigma}(1) \quad 68.4\% \text{ accepts}

\text{sigma}(100) \quad 0.5\% \text{ accepts}
A general, but often ineffective, Metropolis proposal:

\[ q(x' \mid x) = \text{Norm}(x' \mid x, \sigma^2 I) \]

- \( \sigma \) large \( \rightarrow \) many rejections
- \( \sigma \) small \( \rightarrow \) slow diffusion:
  \( \sim (L/\sigma)^2 \) iterations required
**Mixing: Metropolis Random Walk**

*Discrete target distribution is uniform over all states:*

\[ p(x) = \frac{1}{21}, \quad x \in \{0, 1, 2, \ldots, 20\} \]

\[ q(x' \leftarrow x) = \frac{1}{2}, \quad x' \in \{x - 1, x + 1\} \]
MH Algorithms Satisfy Detailed Balance

Metropolis-Hastings (MH) Transition Operator: \( T(x' \leftarrow x) \)
\[
x' \sim q(x | \tilde{x})
\]
\[
r(\tilde{x}, x) = \frac{p^*(\tilde{x})q(x | \tilde{x})}{p^*(x)q(\tilde{x} | x)}
\]
\[
\Pr[x' = \tilde{x}] = \min(1, r(\tilde{x}, x))
\]
\[
\Pr[x' = x] = 1 - \min(1, r(\tilde{x}, x))
\]

Detailed Balance Condition: \( T(x' \leftarrow x)p^*(x) = p^*(x')T(x \leftarrow x') \)

For any \( x' \neq x \):
\[
P(x) \cdot T(x' \leftarrow x) = P(x) \cdot Q(x'; x) \min \left( 1, \frac{P(x')Q(x; x')}{P(x)Q(x'; x)} \right)
\]
\[
= \min \left( P(x)Q(x'; x), P(x')Q(x; x') \right)
\]
\[
= P(x') \cdot Q(x; x') \min \left( 1, \frac{P(x)Q(x'; x)}{P(x')Q(x; x')} \right) = P(x') \cdot T(x \leftarrow x')
\]
CS242: Lecture 7B Outline

- MCMC & Metropolis-Hastings Algorithms
- Gibbs Sampling for Graphical Models
- Moments, mixing, & convergence diagnostics

Gibbs sampling

- A method with no rejections:
  - Initialize \( x \) to some value
  - Pick each variable in turn or randomly and resample

\[
P(x_i | x_j \neq i)
\]

Proof of validity:

a) check detailed balance for component update.

b) Metropolis–Hastings 'proposals'

\[
P(x_i | x_j \neq i) \Rightarrow \text{accept with prob. 1}
\]

Apply a series of these operators. Don't need to check acceptance.
Combining MCMC Transition Proposals

A sequence of operators, each with $P^*$ invariant:

\[ x_0 \sim P^*(x) \]
\[ x_1 \sim T_a(x_1 \leftarrow x_0) \quad P(x_1) = \sum x_0 T_a(x_1 \leftarrow x_0) P^*(x_0) = P^*(x_1) \]
\[ x_2 \sim T_b(x_2 \leftarrow x_1) \quad P(x_2) = \sum x_1 T_b(x_2 \leftarrow x_1) P^*(x_1) = P^*(x_2) \]
\[ x_3 \sim T_c(x_3 \leftarrow x_2) \quad P(x_3) = \sum x_1 T_c(x_3 \leftarrow x_2) P^*(x_2) = P^*(x_3) \]
\[ \ldots \quad \ldots \]

— Combination $T_c T_b T_a$ leaves $P^*$ invariant
— If they can reach any $x$, $T_c T_b T_a$ is a valid MCMC operator
— Individually $T_c$, $T_b$ and $T_a$ need not be ergodic
MCMC for Factor Graphs

\[ p(x) = \frac{1}{Z} \prod_{f \in \mathcal{F}} \psi_f(x_f) \]

\[ Z = \sum_x \prod_{f \in \mathcal{F}} \psi_f(x_f) \]

\[ \mathcal{F} \rightarrow \text{set of hyperedges linking subsets of nodes } f \subseteq \mathcal{V} \]

\[ \Gamma(s) = \{ f \in \mathcal{F} \mid s \in f \} \]

Consider a proposal which modifies one variable:

\[ \tilde{x}_s \sim q_s(\tilde{x}_s | x), \quad \tilde{x}_t = x_t \text{ for all } t \neq s. \]

Acceptance Ratio:

\[ r(\tilde{x}, x) = \frac{p^*(\tilde{x})q(x | \tilde{x})}{p^*(x)q(\tilde{x} | x)} = \frac{q_s(x_s | \tilde{x})}{q_s(\tilde{x}_s | x)} \prod_{f \in \Gamma(s)} \frac{\psi_f(\tilde{x}_f)}{\psi_f(x_f)} \]

- Computation of acceptance ratio is local:
  depends only on variable being resampled and its neighbors
- Proposal is low-dimensional: Easier to have high acceptance rate
Gibbs Sampling for Factor Graphs

\[ p(x) = \frac{1}{Z} \prod_{f \in F} \psi_f(x_f) \]

\[ Z = \sum_x \prod_{f \in F} \psi_f(x_f) \]

\[ \Gamma(s) = \{ f \in F \mid s \in f \} \]

Visiting nodes s in some fixed or random order:

1. Determine the conditional distribution of node s given values for all other nodes (equivalently, its neighbors):
   \[ q_s(\tilde{x}_s \mid x) = p(\tilde{x}_s \mid \{x_t, t \neq s\}) \propto \prod_{f \in \Gamma(s)} \psi_f(\tilde{x}_s, x_f \setminus s) \]

2. Propose a new value for node s, leave all other nodes unchanged:
   \[ \tilde{x}_s \sim q_s(\tilde{x}_s \mid x), \quad \tilde{x}_t = x_t \text{ for all } t \neq s. \]

3. Always accepting this proposal gives valid MCMC:
   \[ x' = \tilde{x} \]
Gibbs Sampling is Metropolis-Hastings

\[ p(x) = \frac{1}{Z} \prod_{f \in \mathcal{F}} \psi_f(x_f) \quad \quad Z = \sum_x \prod_{f \in \mathcal{F}} \psi_f(x_f) \]

Gibbs sampling proposal for node \( s \):

\[ q_s(\tilde{x}_s \mid x) = p(\tilde{x}_s \mid \{x_t, t \neq s \}) \propto \prod_{f \in \Gamma(s)} \psi_f(\tilde{x}_s, x_f \mid s) \]

\( \tilde{x}_s \sim q_s(\tilde{x}_s \mid x), \quad \tilde{x}_t = x_t \) for all \( t \neq s \).

Metropolis-Hastings Acceptance Ratio:

\[ r(\tilde{x}, x) = \frac{q_s(x_s \mid \tilde{x})}{q_s(\tilde{x}_s \mid x)} \prod_{f \in \Gamma(s)} \frac{\psi_f(\tilde{x}_f)}{\psi_f(x_f)} = 1 \]

Always Accept!

plugging in conditional above, using fact that \( \tilde{x}_t = x_t \) for \( t \neq s \).
Gibbs Sampling as Message Passing

Consider a pairwise MRF: (Similar result holds for factor graphs.)

\[ q_i(x_i) \propto \prod_{j \in \Gamma(i)} m_{ji}(x_i) \]

\[ \hat{x}_i \sim q_i(x_i) \quad \text{Draw single sample from marginal} \]

\[ m_{ij}(x_j) \propto \psi_{ij}(\hat{x}_i, x_j) \quad \text{Use sample to extract a “slice” of pairwise potential} \]

- For discrete variables, sample from a categorical distribution.
- For continuous variables, need to analyze model to determine conditional.

\[ p(x) = \frac{1}{Z} \prod_{(s,t) \in \mathcal{E}} \psi_{st}(x_s, x_t) \]
Gibbs Sampling Implementation

Gibbs sampling benefits from few free choices and convenient features of conditional distributions:

- Conditionals with a few discrete settings can be explicitly normalized:
  \[
  P(x_i | x_j \neq i) \propto P(x_i, x_j \neq i) = \frac{P(x_i, x_j \neq i)}{\sum_{x_i'} P(x_i', x_j \neq i)} \leftarrow \text{this sum is small and easy}
  \]

- Continuous conditionals only univariate
  ⇒ amenable to standard sampling methods.
  - Inverse CDF sampling
  - Rejection sampling
  - Slice sampling
  - …
CS242: Lecture 7B Outline

- MCMC & Metropolis-Hastings Algorithms
- Gibbs Sampling for Graphical Models
- Moments, mixing, & convergence diagnostics
MCMC Implementation & Application

• The samples aren't independent. Should we thin, only keep every $K$th sample?

• Arbitrary initialization means starting iterations are bad. Should we discard a “burn-in” period?

• Maybe we should perform multiple runs?

• How do we know if we have run for long enough?
Approximately independent samples can be obtained by \textit{thinning}. However, \textbf{all the samples can be used.}

\textbf{Use the simple Monte Carlo estimator on MCMC samples.} It is:

\begin{itemize}
\item consistent
\item unbiased if the chain has “burned in”
\end{itemize}

$$
\mathbb{E}_P[f] \approx \frac{1}{S} \sum_{s=1}^{S} f(x^{(s)})
$$

\textbf{The correct motivation to thin:} if computing $f(x^{(s)})$ is expensive

\textbf{Thinned Sampling}

\textbf{All Samples after Burn-in}
Autocovariance: Empirical covariance of values produced by MCMC method, versus iteration lag (spacing)

- Small autocovariances are necessary, but not sufficient, to demonstrate mixing to the target distribution.
- Fairly reliable for unimodal posteriors, but very misleading more generally.

Trace Plot: Value of some “interesting” summary statistic, versus MCMC iteration

For diagnostics:
- Standard software packages like R-CODA
- For opinion on thinning, multiple runs, burn in, etc.
Mixing for Gibbs Sampling

Gibbs sampling: a method with no rejections.

- Initialize \( x \) to some value.
- Pick each variable in turn or randomly and resample.

\[
P(x_i | x_j \neq i)
\]

Figure from PRML, Bishop (2006)

Proof of validity:

a) check detailed balance for component update.

b) Metropolis–Hastings 'proposals'

\[
\rightarrow \text{accept with prob.}
\]

Apply a series of these operators. Don't need to check acceptance.

Each Gibbs sampling proposal only changes one variable at a time.

One or more “rare events” may need to happen before Gibbs sampler moves between modes: SLOW mixing

Example: Slow mixing for a pair of binary variables

\[
\begin{array}{c|cc}
  z_1 & z_1 = 0 & z_1 = 1 \\
  \hline
  z_2 = 0 & 0.6 - \epsilon & \epsilon \\
  z_2 = 1 & \epsilon & 0.4 - \epsilon 
\end{array}
\]

Example: Slow mixing for a pair of binary variables
A Gibbs Sampler for Spatial MRFs

Ising and Potts Priors on Partitions

\[ p(z) = \frac{1}{Z(\beta)} \prod_{(s,t) \in E} \psi_{st}(z_s, z_t) \]

\[ \log \psi_{st}(z_s, z_t) = \begin{cases} \beta_{st} > 0 & z_s = z_t \\ 0 & \text{otherwise} \end{cases} \]

Gibbs sampler is special case of pairwise MRF sampler.

Previous Applications

- Interactive foreground segmentation
- Supervised segmentation with object category appearance models
- These are “strong” likelihoods.

Is the prior itself effective for images?

GrabCut: Rother, Kolmogorov, & Blake 2004

Verbeek & Triggs, 2007
200 Gibbs Iterations

10,000 Gibbs Iterations

Geman & Geman, PAMI 1984

128 x128 grid
8 nearest neighbor edges
K = 5 states $\beta = 2/3$

Potts potentials:
10-State Potts Models: Extended Gibbs

States sorted by size: largest in blue, smallest in red
The Ising/Potts model is not well suited to segmentation tasks

Even within the phase transition region, samples lack the size distribution and spatial coherence of real image segments.

Output of a limited MCMC run can be arbitrarily misleading!
**MCMC & Computational Resources**

(1) Best practical option: A few (> 1) initializations for as many iterations as possible

(2) Arbitrary initialization means starting iterations are bad. Should we discard a "burn-in" period?

(3) Maybe we should perform multiple runs? How do we know if we have run for long enough?