Probabilistic Graphical Models

Approximate Inference: Advanced Topics in MCMC

Eric Xing
Lecture 18, March 24, 2014
Recap of MCMC

- Markov Chain Monte Carlo methods use adaptive proposals $Q(x'|x)$ to sample from the true distribution $P(x)$

- Metropolis-Hastings allows you to specify any proposal $Q(x'|x)$
  - But choosing a good $Q(x'|x)$ requires care

- Gibbs sampling sets the proposal $Q(x'|x)$ to the conditional distribution $P(x'|x)$
  - Acceptance rate always 1!
  - But remember that high acceptance usually entails slow exploration
  - In fact, there are better MCMC algorithms for certain models

- Knowing when to halt burn-in is an art
Auxiliary Variables

- Advanced MCMC algorithms rely on **auxiliary variables**
  - Auxiliary variables are extra r.v.s not from the original model
  - They are *random-valued intermediate quantities* that allow us to sample model r.v.s in creative ways

- Suppose x is an r.v. and v is an a.v.. Generally, we use a.v.s when:
  - $P(x|v)$ and $P(v|x)$ have simple forms
  - $P(x,v)$ is easy to navigate
Slice Sampling

- Slice sampling is an auxiliary variable MCMC algorithm
  - Key idea: uniformly sample the area under $P'(x) = aP(X)$, instead of $P(x)$
  - Never evaluate expensive $P(x)$, only evaluate cheap $P'(x)$
Slice Sampling

- When is Slice sampling useful?
  - Ex: Markov Random Fields where $P(x) = (1/a) \cdot \exp(bx)$
  - Normalizer $(1/a)$ usually intractable to evaluate!
  - Slice sampling only requires (easy) evaluation of $P'(x) = \exp(bx)$
Slice Sampling

- Slice sampling uses an a.v. $h$ (in addition to the r.v. $x$)
  - The pair $(x,h)$ is the position of the sampler in the area under $P'(x)$
- We only need to know $P'(x) = aP(x)$ for some unknown $a$
- The algorithm iterates between two steps:
  - Step 1: sample $h$ from $Q(h \mid x) = \text{Uniform}[0, P'(x)]$
  - Step 2: sample $x$ from $Q(x \mid h) \propto \begin{cases} 
1 & \text{if } P'(x) \geq h \\
0 & \text{otherwise}
\end{cases}$ (uniform dist. on all $x$ s.t. $P'(x) \geq h$)
Slice Sampling

- The algorithm iterates between two steps:
  - Step 1: sample \( h \) from \( Q(h \mid x) = \text{Uniform}[0, P'(x)] \)
  - Step 2: sample \( x \) from
    \[
    Q(x \mid h) \propto \begin{cases} 
    1 & \text{if } P'(x) \geq h \\
    0 & \text{otherwise}
    \end{cases}
    \text{ (uniform dist. on all } x \text{ s.t. } P'(x) \geq h) \]

- Step 2 requires finding the set \( \{x \text{ s.t. } P'(x) \geq h\} \)
  - Alternative 1: rejection sampling (reject whenever we get \( x \) s.t. \( P'(x) < h \))
  - Alternative 2: “Bracketing” technique (to be presented shortly)
Why does this work?

- At convergence, the samples \((x,h)\) will be uniformly distributed under the area of \(P'(x)\).
- If we marginalize out \(h\), we get samples from \(P(x) = \frac{1}{a}P'(x)\).
  - Never needed to evaluate normalizer \((1/a)!\)
Why does this work?

- How to marginalize out $h$?
  - We have samples $(x_1, h_1)$, $(x_1, h_2)$, $(x_2, h_2)$, $(x_2, h_3)$, …
  - Marginalization is just dropping $h$ from the samples
  - After dropping $h$, left with $x_1$, $x_2$, $x_3$, … which are samples from $P(x)$!
Handling difficult $Q(x|h)$

- Step 2 (sampling $Q(x|h)$) may not be easy
  - For complex distributions, cannot analytically find \{x s.t. $P'(x) \geq h$\}
  - However, we can still easily evaluate $P'(x)$ at any $x$...

- Solution: “bracketing” strategy
  1. Draw a random bracket width $w$, and place the bracket on $(x_{\text{old}}, h)$
  2. Expand the bracket until the endpoints $a$, $b$ are “above $P'(x)$”: i.e. $P'(a) < h$ and $P'(b) < h$
  3. Uniformly sample from within the bracket (reject samples $x$ s.t. $P'(x) < h$)

Satisfies detailed balance, but not as efficient because the brackets can miss other modes
How to Sample from Different Model Spaces?

- **Detailed Balance**

\[ \pi(x')T(x' | x') = \pi(x)T(x' | x) \]

- **Why we need detailed balance?**
  - Stationary distribution \( \pi(x) \)!
  - Then how can such a \( \pi(x) \) handle the following case?

2 clusters

3 clusters
Reversible Jump MCMC

- An MCMC algorithm that allows for model selection
  - Examples: choosing # clusters $K$, or even switching between two completely different models $P_1(x)$ and $P_2(x)$
RJMCMC

- Definitions:
  - \( x \) – model r.v.s (the number of \( x \)'s can change depending on the model)
  - \( u \) – auxiliary variables used to perform “dimension matching”
  - \( m \) – an indicator representing which model we are currently using
  - \( P(x|m) \) – probability distribution for r.v.s \( x \) assuming model \( m \)

- RJMCMC uses two types of proposal distribution:
  - \( j(m'|m) \) – model proposal; switches from model \( m \) to \( m' \). Must be reversible!
  - \( q(x',u'|m\rightarrow m',x,u) \) – data proposal; proposes \( (x',u') \) under the new model \( m' \), starting from \( (x,u) \) under the previous model \( m \)

- RJMCMC also requires a mapping function:
  - \( h_{m,m'}(x,u) \) – explains how \( (x,u) \) under model \( m \) maps to \( (x',u') \) under \( m' \)
The mapping function $h()$

- Properties of $h_{m,m'}(x,u)$:
  - Is deterministic (non-random)
  - Takes a vector $(x,u)$ as input, and outputs a vector $(x',u')$
    - Dimension of $x$ is usually different from $x'$ (and likewise for $u,u'$)
  - Must be bijective (one-to-one) so that its inverse is well-defined

- Simple example: switching from 2 clusters to 3 clusters
  - Let $x_1, x_2$ be the first 2 cluster centers
  - Randomly draw an a.v. $u$ to be the 3rd cluster center
  - Then
    $$h_{2,3}(x_1, x_2, u) = \begin{bmatrix} x_1' = x_1 \\ x_2' = x_2 \\ x_3' = u \end{bmatrix}$$
    - i.e. $h_{2,3}()$ maps a 2-cluster model to a 3-cluster model by setting the 3rd cluster center $x_3'$ to $u$ (dimension matching)
RJMCMC Algorithm

1. Initialize $x,u,m$

2. Repeat until convergence:
   1. Propose a new model $m'$ using $j(m'|m)$
   2. Propose a new model state $(x',u')$ using $q(x',u'|m\rightarrow m',x,u)$
   3. Compute the acceptance probability:

   $A(m',x',u'|m,x,u) = \min\left(1, \frac{P(x'|m')}{{P(x|m)}} \times \frac{j(m'|m)}{j(m|m)} \times \frac{q(x,u|m'\rightarrow m',x,u')}{q(x',u'|m\rightarrow m',x,u)} \times \left|\text{det} \frac{\partial h_{m,m'}(x,u)}{\partial(x,u)}\right|\right)$

   - Ratio of model probs.
   - Inv. ratio of model proposals
   - Inv. ratio of data proposals
   - Absolute value of the determinant of the Jacobian of $h()$
The abs-det-Jacobian term

- A “Jacobian” is a matrix of all 1st derivatives
  - Example: 2-clusters to 3-clusters; recall \( h_{2,3}(x_1, x_2, u) = \begin{bmatrix} x'_1 = x_1 \\ x'_2 = x_2 \\ x'_3 = u \end{bmatrix} \)

The Jacobian is

\[
\frac{\partial h_{2,3}(x_1, x_2, u)}{\partial (x_1, x_2, u)} = \begin{bmatrix}
\frac{\partial x'_1}{\partial x_1} & \frac{\partial x'_1}{\partial x_2} & \frac{\partial x'_1}{\partial u} \\
\frac{\partial x'_2}{\partial x_1} & \frac{\partial x'_2}{\partial x_2} & \frac{\partial x'_2}{\partial u} \\
\frac{\partial x'_3}{\partial x_1} & \frac{\partial x'_3}{\partial x_2} & \frac{\partial x'_3}{\partial u}
\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

thus

\[
\left| \frac{\partial h_{2,3}(x_1, x_2, u)}{\partial (x_1, x_2, u)} \right| = 1
\]

In general, we construct \( h() \) so that the abs-det-Jacobian term is trivial (e.g. 1)
The Jacobian term

- **Why** do we need the Jacobian?
  - It arises from a change of variables during integration!
  - Consider the detailed balance equation; take integrals on both sides:
    \[
    \int P(x)g(x',u'|x,u)A(x',u'|x,u)dxdu = \int P(x')g(x,u|x',u')A(x,u|x',u')dx'du'
    \]
  - \(g()\) combines the model proposal \(j()\) and the data proposal \(q()\)
  - For simplicity, we omit the model indicator \(m\), because the dimensionality of \((x,u)\) completely identifies which model \(m\) the system is in
  - Now perform a change of variables from \((x',u')\) to \((x,u)\) on the RHS:
    \[
    \int P(x)g(x',u'|x,u)A(x',u'|x,u)dxdu = \int P(x')g(x,u|x',u')A(x,u|x',u')dx'du' \left| \frac{\partial h_{m,m}(x,u)}{\partial (x,u)} \right| dxdu
    \]
  - The equation above holds if, for all \(x,x',u,u'\),
    \[
    P(x)g(x',u'|x,u)A(x',u'|x,u) = P(x')g(x,u|x',u')A(x,u|x',u') \left| \frac{\partial h_{m,m}(x,u)}{\partial (x,u)} \right|
    \]
The Jacobian term

- **Why** do we need the Jacobian?
  - The detailed balance condition holds if, for all $x,x',u,u'$,
    \[
    P(x)g(x',u' \mid x,u)A(x',u' \mid x,u) = P(x')g(x,u \mid x',u')A(x,u \mid x',u')\left| \det \frac{\partial h(x,u),(x',u')}{\partial (x,u)} \right|
    \]
  - We can now construct an acceptance probability that satisfies detailed balance (see previous lecture, MH algorithm):
    \[
    A(x',u' \mid x,u) = \min \left( 1, \frac{P(x')}{P(x)} \frac{g(x,u \mid x',u')}{g(x',u' \mid x,u)} \left| \det \frac{\partial h(x,u),(x',u')}{\partial (x,u)} \right| \right)
    \]
  - Restoring the model indicator $m$, we get
    \[
    A(m',x',u' \mid m,x,u) = \min \left( 1, \frac{P(x' \mid m')}{P(x \mid m)} \frac{j(m' \mid m)}{j(m \mid m')} \frac{q(x,u \mid m' \rightarrow m,x',u')}{q(x',u' \mid m' \rightarrow m',x,u)} \left| \det \frac{\partial h_{m,m'}(x,u)}{\partial (x,u)} \right| \right)
    \]
Question:

- What is our stationary distribution in our RJMCMC?
RJMCMC Example: Clustering

- Models: Let \( m = 1, 2, 3, \ldots \) denote the number of clusters
  - \( P(x, c|m) \) - probability of (observed) data \( x \) and (unknown) cluster centers \( c \), assuming \( m \) clusters
    - Can be a Gaussian mixture model or any other clustering model. For this example, we don’t need to know its exact form.

- Proposal distributions:
  - \( j(m'|m) \) – switches from \( m \) to \( m' \) clusters, where \( m' = \{m-1, m, m+1\} \)
    - \( m' = m-1 \) is used to decrease the number of clusters
    - \( m' = m+1 \) is used to increase the number of clusters
    - \( m' = m \) is used to change cluster centers \( c \)
  - \( q(x', c', u'|m \rightarrow m', x, c, u) \) – form differs depending on \( m' \) and \( m \)
  - \( h_{m,m'}(c,u) \) – again, form differs depending on \( m' \) and \( m \)
  - abs-det-Jacobian – turns out that this is always 1!
RJMCMC Example: Clustering

Starting state: m cluster centers

Remove cluster (e.g. \(c_m\))

Change cluster center (e.g. \(c_1\))

Add cluster

© Eric Xing @ CMU, 2005-2014
RJMCMC Example: Clustering

- We set \( j() \) as follows:
  \[
  j(m' | m) = \begin{cases} 
  0.5 - p & \text{if } m' = m - 1 \\
  2p & \text{if } m' = m \\
  0.5 - p & \text{if } m' = m + 1 
  \end{cases}
  \]

- For \( q() \), \( h() \) and the Jacobian, consider the 3 cases separately:
  - \( m' = m \) (change cluster center):
    - \( u, u' \) are used to change the value of some \( c_i \)
    - First, pick a cluster center \( i \) in \( \{1, \ldots, m\} \) to change assignment (at uniform)
    - Next, draw a new cluster center \( u \) according to some proposal \( q_{\text{center}}(u) \)
    - Finally, set \( c'_i = u \)
  
  Notice that reverse moves have the same probability as forward moves

"Explore cluster centers \( c \) 2p of the time, change the number of clusters 1-2p of the time"
RJMCMC Example: Clustering

- For \(q()\), \(h()\) and the Jacobian, consider the 3 cases separately:
  - \(m' = m\) (change cluster center):
    - What does the abs-det-Jacobian look like?
    - Recall that \(h_{i,m,m'=m}(c,u)\) sets \(c'_j = c_j\) for all \(j \neq i\), and \(c'_i = u\), and \(u' = c_i\)
    - Let’s say we’re changing \(c_i\), where \(i = m\)

\[
\frac{\partial h_{i=m,m'=m}(c,u)}{\partial (c,u)} = \begin{bmatrix}
\frac{\partial c'_1}{\partial c_1} & \frac{\partial c'_1}{\partial c_2} & \ldots & \frac{\partial c'_1}{\partial c_m} & \frac{\partial c'_1}{\partial u} \\
\frac{\partial c'_2}{\partial c_1} & \frac{\partial c'_2}{\partial c_2} & \ldots & \frac{\partial c'_2}{\partial c_m} & \frac{\partial c'_2}{\partial u} \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
\frac{\partial c'_m}{\partial c_1} & \frac{\partial c'_m}{\partial c_2} & \ldots & \frac{\partial c'_m}{\partial c_m} & \frac{\partial c'_m}{\partial u} \\
\frac{\partial u'}{\partial c_1} & \frac{\partial u'}{\partial c_2} & \ldots & \frac{\partial u'}{\partial c_m} & \frac{\partial u'}{\partial u} \\
\end{bmatrix} = \begin{bmatrix}
1 & 0 & \ldots & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & \ldots & 0 & 1 \\
0 & 0 & \ldots & 1 & 0 \\
\end{bmatrix} = | \frac{\partial h_{i=m,m'=m}(c,u)}{\partial (c,u)} | = | -1 | = 1
\]

In fact, the abs-det-Jacobian is 1 for any choice of \(i\)!
RJMCMC Example: Clustering

- For $q()$, $h()$ and the Jacobian, consider the 3 cases separately:
  - $m' = m-1$ (remove a cluster):
    - $u$ is empty, and $u'$ matches the cluster to be removed
    - Pick a cluster center $i$ in $\{1, \ldots, m\}$ to remove (at uniform)

\[
q(x', c', u' \mid m \rightarrow m', x, c, u) = \frac{1}{m} \quad \text{and} \quad h_{i,m,m'=m-1}(c,u) = \begin{bmatrix} c'_1 \\ c'_2 \\ \vdots \\ c'_{m-1} \\ u' \end{bmatrix} \text{ where } c'_j = c_j \text{ if } j < i, \text{ and } c'_j = c_{j+1} \text{ if } j > i, \text{ and } u' = c_i
\]
RJMCMC Example: Clustering

- For q(), h() and the Jacobian, consider the 3 cases separately:
  - \( m' = m-1 \) (remove a cluster):
    - For the Jacobian, let's assume we're removing cluster \( c_i \) where \( i = m \)
    - Thus we set \( c'_j = c_j \) for all \( j < m \), and \( u' = c_m \)

\[
\frac{\partial h_{i=m,m'=m-1}(c,u)}{\partial(c,u)} = \begin{bmatrix}
\frac{\partial c'_1}{\partial c_1} & \frac{\partial c'_1}{\partial c_2} & \cdots & \frac{\partial c'_1}{\partial c_{m-1}} & \frac{\partial c'_1}{\partial c_m} \\
\frac{\partial c'_2}{\partial c_1} & \frac{\partial c'_2}{\partial c_2} & \cdots & \frac{\partial c'_2}{\partial c_{m-1}} & \frac{\partial c'_2}{\partial c_m} \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
\frac{\partial c'_{m-1}}{\partial c_1} & \frac{\partial c'_{m-1}}{\partial c_2} & \cdots & \frac{\partial c'_{m-1}}{\partial c_{m-1}} & \frac{\partial c'_{m-1}}{\partial c_m} \\
\frac{\partial u'}{\partial c_1} & \frac{\partial u'}{\partial c_2} & \cdots & \frac{\partial u'}{\partial c_{m-1}} & \frac{\partial u'}{\partial c_m}
\end{bmatrix} = \begin{bmatrix}
1 & 0 & \cdots & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & \cdots & 0 & 1
\end{bmatrix}
\]

Therefore \[ \left| \det \frac{\partial h_{i=m,m'=m-1}(c,u)}{\partial(c,u)} \right| = 1 \]

Again, the abs-det-Jacobian is 1 for any choice of i!
RJMCMC Example: Clustering

- For $q()$, $h()$ and the Jacobian, consider the 3 cases separately:
  - $m' = m+1$ (add a cluster):
    - $u$ is the center of the cluster to be added, and $u'$ is empty
    - We draw a cluster center $u$ according to some proposal $q_{\text{center}}(u)$

$$q(x', c', u' \mid m \rightarrow m', x, c, u) = q_{\text{center}}(u)$$

and

$$h_{i, m, m'}(c, u) = \begin{bmatrix} c'_{1} \\ c'_{2} \\ \vdots \\ c'_{m} \\ c'_{m+1} \end{bmatrix}$$

where $c'_{j} = c_{j}$ for all $j \leq m$, and $c'_{m+1} = u$
RJMCMC Example: Clustering

- For $q()$, $h()$ and the Jacobian, consider the 3 cases separately:
  - $m' = m+1$ (add a cluster):
    - For the Jacobian, recall we set $c'_j = c_j$ for all $j \leq m$, and $c'_{m+1} = u$

$$\frac{\partial h_{m,m'=m+1}(c,u)}{\partial (c,u)} = \begin{bmatrix}
\frac{\partial c'_1}{\partial c_1} & \frac{\partial c'_1}{\partial c_2} & \cdots & \frac{\partial c'_1}{\partial c_m} & \frac{\partial c'_1}{\partial u} \\
\frac{\partial c'_2}{\partial c_1} & \frac{\partial c'_2}{\partial c_2} & \cdots & \frac{\partial c'_2}{\partial c_m} & \frac{\partial c'_2}{\partial u} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\frac{\partial c'_m}{\partial c_1} & \frac{\partial c'_m}{\partial c_2} & \cdots & \frac{\partial c'_m}{\partial c_m} & \frac{\partial c'_m}{\partial u} \\
\frac{\partial c'_{m+1}}{\partial c_1} & \frac{\partial c'_{m+1}}{\partial c_2} & \cdots & \frac{\partial c'_{m+1}}{\partial c_m} & \frac{\partial c'_{m+1}}{\partial u}
\end{bmatrix} = \begin{bmatrix}
1 & 0 & \cdots & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & \cdots & 0 & 1
\end{bmatrix}$$

Therefore

$$\left| \det \frac{\partial h_{m,m'=m+1}(c,u)}{\partial (c,u)} \right| = |1| = 1$$
RJMCMC Example: Clustering

- Notice the following important properties:
  - All model changes $j(m'|m)$ are all reversible
    - We can get to any number of clusters $m$
    - We can change the location of any cluster $i$
    - This ensures we converge to the stationary distribution
  - abs-det-Jacobian is always 1
    - We designed our r.v. mappings $h()$ to make this true!

- Take note:
  - For most mixture models, we can’t simply use $P(x,c|m)$. We need to introduce hidden cluster assignment variables $z$ for each data point $x$, and incorporate them into the proposals.
  - The basic principle of RJMCMC remains the same, though
Large-scale MCMC

- Modern datasets can be very large
  - Millions of data points
  - Require Gigabytes of memory
  - E.x. Yahoo web graph has ~1.4 billion nodes and 6.6 billion edges

- So far, we have not explained how to take advantage of parallelism in MCMC
  - Without parallelism, we cannot use large datasets!

- In the rest of this lecture, we will cover techniques that permit multiple CPUs/cores to be used with MCMC
Taking Multiple Chains

- Proper use of MCMC actually requires parallelism
  - To determine convergence, you need to take multiple MCMC chains
  - Chains are independent, so you can run one chain per CPU
  - Once converged, you can combine samples from all chains
Taking Multiple Chains

- Taking multiple chains doesn’t solve all issues, though
  - If burn-in is long, then all chains will take a long time to converge!
  - We need a way to take each sample faster…
Parallel Gibbs Sampling

- Recall that in MRFs, we Gibbs sample by sampling from $P(x|MB(x))$, the conditional distribution of $x$ given its Markov Blanket $MB(x)$
  - For MRFs, the Markov Blanket of $x$ is just its neighbors
  - In the MRF below, the red node's Markov Blanket consists of the blue nodes
Parallel Gibbs Sampling

- Observe that we can Gibbs sample the two green nodes simultaneously
  - Neither node is part of the other’s Markov Blanket, so their conditional distributions do not depend on each other
  - Sampling one of the green nodes doesn’t change the conditional distribution of the other node!
Parallel Gibbs Sampling

- How do we generalize this idea to the whole graph?
  - Find subsets of nodes, such that all nodes in a given subset are not in each other’s Markov Blankets, and the subsets cover the whole graph
  - The subsets should be as large as possible
    - Because we can Gibbs sample all nodes in a subset at the same time
  - At the same time, we want as few subsets as possible
    - The Markov Blankets of different subsets overlap, so they cannot be sampled at the same time. We must process the subsets sequentially.
Parallel Gibbs Sampling

- We can find these covering subsets with k-coloring algorithms (Gonzales et al., 2011)
  - A k-coloring algorithm colors a graph using k colors, such that:
    - Every node gets one color
    - No edge has two nodes of the same color
- Trees always admit a 2-coloring (e.g. below)
  - Assign one color to some node, and alternate colors as you move away

© Eric Xing @ CMU, 2005-2014
Parallel Gibbs Sampling

- Bipartite graphs are always 2-colorable
  - Color each side of the bipartite graph with opposite colors
  - e.x. Latent Dirichlet Allocation model is bipartite

- However, not all graphs have k-colorings for all $k \geq 2$
  - In the worst case, a graph with $n$ nodes can require $n$ colors
    - The full clique is one such graph
  - Determining if a graph is k-colorable for $k > 2$ is NP-complete
  - In practice, we employ heuristics to find k-colorings

- Instead of using k-colorings, why not just Gibbs sample all variables at the same time?
  - The Markov Chain may become non-ergodic, and is no longer guaranteed to converge to the stationary distribution!
Online MCMC

- In “online” algorithms, we need to process new data points one-at-a-time
  - Moreover, we have to “forget” older data points because memory is finite

- For such applications to be viable, we can only afford constant time work per new data point
  - Otherwise we will reach a point where new data can no longer be processed in a reasonable amount of time

- What MCMC techniques can we use to make an online algorithm?
Sequential Monte Carlo

- SMC is a generalization of Particle Filters
  - Recall that PFs incrementally sample $P(X_t|Y_{1:t})$, where the Xs are latent r.v.s and the Ys are observations under a state-space model
  - SMC does not assume the GM is a state-space model, or has any particular structure at all

- Suppose we have n r.v.s $x_1,\ldots,x_n$
  - SMC first draws samples from the marginal distribution $P(x_1)$, then $P(x_{1:2})$, and so on until $P(x_{1:n})$
  - Key idea: Construct proposals such that we sample from $P(x_{1:k+1})$ in constant time, given samples from $P(x_{1:k})$
  - Like other MCMC algorithms, we only require that we can evaluate $P'(x_{1:n}) = aP(x_{1:n})$ for some unknown $a$
Sequential Importance Sampling

- SIS is the foundation of Sequential Monte Carlo
  - It allows new variables to be sampled in constant time, without resampling older variables

- SIS uses proposal distributions with the following structure:

\[
q_n(x_{1:n}) = q_{n-1}(x_{1:n-1})q_n(x_n | x_{1:n-1}) \\
= q_1(x_1) \prod_{k=2}^{n} q_k(x_k | x_{1:k-1})
\]

- Notice we can propose \(x_{k+1}\) if we’ve already drawn \(x_{1:k}\), without having to redraw \(x_{1:k}\)
Sequential Importance Sampling

- In normalized importance sampling, recall how the sample weights $w^i$ are defined:
  \[ \langle f(X) \rangle_p = \sum_i f(x^i)w^i \]

  where
  \[ w^i = \sum_j r_j^i \quad \text{and} \quad r^i = \frac{P'(x^i)}{Q(x^i)} \]

- In SIS, the unnormalized weights $r$ can be rewritten as a telescoping product:
  \[
  r(x_{1:n}) = \frac{P_n'(x_{1:n})}{q_n(x_{1:n})} \\
  = \frac{P_{n-1}'(x_{1:n-1})}{q_{n-1}(x_{1:n-1})} \frac{P_n'(x_{1:n})}{P_{n-1}'(x_{1:n-1})q_n(x_n \mid x_{1:n-1})} \\
  = r_{n-1}(x_{1:n-1})\alpha_n(x_{1:n}) \\
  = r_1(x_1)\prod_{k=2}^n \alpha_k(x_{1:k})
  \]

  where
  \[ \alpha_n(x_{1:n}) = \frac{P_n'(x_{1:n})}{P_{n-1}'(x_{1:n-1})q_n(x_n \mid x_{1:n-1})} \]
Sequential Importance Sampling

\[
r(x_{1:n}) = r_1(x_1) \prod_{k=2}^{n} \alpha_k(x_{1:k}) \quad \text{where} \quad \alpha_n(x_{1:n}) = \frac{P_n'(x_{1:n})}{P'_{n-1}(x_{1:n-1}) q_n(x_n | x_{1:n-1})}
\]

- This means the unnormalized weights \( r \) can be computed incrementally
  - Compute \( \alpha_n \) and use it to update \( r(x_{1:n-1}) \) to \( r(x_{1:n}) \)
    - NB: For this update to be constant time, we also require \( P'_n(x_{1:n}) \) to be computable from \( P'_{n-1}(x_{1:n-1}) \) in constant time
  - We remember the unnormalized weights \( r \) at each iteration, and compute the normalized weights \( w \) as needed from \( r \)

- Thus, we can sample \( x \) AND compute the normalized weights \( w \) using constant time per new variable \( x_n \)
  - So SIS meets the requirements for an online inference algorithm!

- Even better, the samples don’t depend on each other
  - Assign one CPU core per sample to make the SIS algorithm parallel!
Sequential Importance Sampling

- **SIS algorithm:**
  - At time $n = 1$
    - Draw samples $x_1^i \sim q_1(x_1)$
    - Compute unnormalized weights $r_1^i = p'(x_1^i) / q_1(x_1^i)$
    - Compute normalized weights $w_1^i$ by normalizing $r_1^i$
  - At time $n \geq 2$
    - Draw samples $x_n^i \sim q_n(x_n|x_{1:n-1}^i)$
    - Compute unnormalized weights $r_n^i = r_{n-1}^i \alpha_n(x_{1:n}^i) = r_{n-1}^i \frac{p_n'(x_{1:n}^i)}{p_{n-1}'(x_{1:n-1}^i)q_n(x_n^i|x_{1:n-1}^i)}$
    - Compute normalized weights $w_n^i$ by normalizing $r_n^i$
Sequential Importance Sampling

- But we are not done yet!

- Unfortunately, SIS suffers from a severe drawback: the variance of the samples increases exponentially with $n$!
  - See eq (31) of Doucet’s SMC tutorial for an example

- Resampling at each iteration will decrease the sample variance!
  - Similar to weighted resampling from the first MC lecture!
Multinomial Resampling

- Suppose we have $m$ samples $x^1,\ldots,x^m$ with corresponding importance weights $w^1,\ldots,w^m$

- Construct a categorical distribution from these samples:
  - This distribution has $m$ categories (choices)
  - The probability of drawing category $k$ is $w^k$
  - Drawing category $k$ gets us $x^k$

- To resample, just draw $N$ times from this distribution
  - Note that $N$ can be greater/less than $m$!

- For more advanced strategies such as systematic and residual resampling, refer to page 13 of Doucet’s SMC tutorial
Why Resample?

- Apart from decreasing variance, there are other reasons...

- Resampling removes samples $x_k$ with low weights $w_k$
  - Low-weight samples come from low-probability regions of $P(x)$
  - We want to focus computation on high-probability regions of $P(x)$
  - Notice that each sample gets an equal amount of computation, regardless of its weight $w_k$
    - Resampling ensures that more computation is spent on samples $x_k$ that come from high-probability regions of $P(x)$

- Resampling prevents a small number of samples $x_k$ from dominating the empirical distribution
  - Resampling resets all weights $w_k$ to $1/N$
    - This prevents sample weights $w_k$ from growing until they reach 1
Sequential Monte Carlo

- The SMC algorithm is just SIS with resampling:
  - At time $n = 1$
    - Draw samples $x_1^i \sim q_1(x_1)$
    - Compute unnormalized weights $r_1^i = \frac{P_1'(x_1^i)}{q_1(x_1^i)}$
    - Compute normalized weights $w_1^i$ by normalizing $r_1^i$
    - Resample $w_1^i, x_1^i$ into $N$ equally-weighted particles $x_1^i$
  - At time $n \geq 2$
    - Draw samples $x_n^i \sim q_n(x_n|x_{1:n-1}^i)$
    - Compute unnormalized weights $r_n^i = r_{n-1}^i \alpha_n(x_{1:n}^i) = r_{n-1}^i \frac{P_n'(x_{1:n}^i)}{P_{n-1}'(x_{1:n-1}^i)q_n(x_n^i|x_{1:n-1}^i)}$
    - Compute normalized weights $w_n^i$ by normalizing $r_n^i$
    - Resample $w_n^i, x_{1:n}^i$ into $N$ equally-weighted particles $x_{1:n}^i$
Summary

- Slice sampling
  - Samples from area under $P(x)$

- Reverse Jump MCMC
  - Allows us to switch between different models $P(x)$

- Parallel Gibbs sampling
  - Exploit graph colorings to sample same-colored nodes in parallel

- Sequential Monte Carlo
  - Uses incremental proposal distributions
  - Provides a framework for designing online, parallel MCMC algorithms