Approaches to inference

• Exact inference algorithms
  • The elimination algorithm
  • Message-passing algorithm (sum-product, belief propagation)
  • Junction tree algorithm

• Approximate inference techniques
  • Variational algorithms
    • Loopy belief propagation
    • Mean field approximation
  • Stochastic simulation / sampling methods
  • Markov chain Monte Carlo methods
How to represent a joint distribution?

• Closed form representation

\[
(x_1, \ldots, x_p)^T \sim \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp \left( -\frac{1}{2} x - \mu \right)^T \Sigma^{-1} (x - \mu)
\]

\[
E_p(f(x)) = \int f(x)p(x)dx
\]

• Sample-based representation

Collect samples \( X^{(m)} \sim P(x) \) if we draw a lot of samples we can use the law of large numbers to get that \( E_p(f(x)) = \sum_m f(X^{(m)})/|m| \)
Monte Carlo Methods

• Draw random samples from the desired distribution
• Yield a stochastic representation of a complex distribution
  • marginals and other expectations can be approximated using sample-based averages
  • $E_p(f(x)) = \Sigma_{m} f(X^{(m)})/|m|$
• Asymptotically exact and easy to apply to arbitrary models
• Challenges:
  • how to draw samples from a given dist. (not all distributions can be trivially sampled)?
  • how to make better use of the samples (not all sample are useful, or equally useful, see an example later)?
  • how to know we've sampled enough?
Monte Carlo Methods

• Direct Sampling
  • We have seen it.
  • Very difficult to populate a high-dimensional state space

• Rejection Sampling
  • Create samples like direct sampling, only count samples which is consistent with given evidences.

• Likelihood weighting, ...
  • Sample variables and calculate evidence weight. Only create the samples which support the evidences.

• Markov chain Monte Carlo (MCMC)
  • Metropolis-Hasting
  • Gibbs
Rejection sampling

• Suppose we wish to sample from dist. \( \Pi(X) = \Pi'(X)/Z. \)
  - \( \Pi(X) \) is difficult to sample, but \( \Pi'(X) \) is easy to evaluate
  - Sample from a simpler distribution \( Q(X) \)
  - Rejection sampling \( x^* \sim Q(X), \) accept \( x^* \) w.p. \( \Pi'(x^*)/kQ(x^*) \)

\[
p(x) = \frac{[\Pi'(x)/kQ(x)]Q(x)}{\int[\Pi'(x)/kQ(x)]Q(x)dx} \\
= \frac{\Pi'(x)}{\int \Pi'(x)dx} = \Pi(x)
\]

• Correctness:

• Pitfall: We gained a sample but what did we pay?
Unnormalized Importance Sampling

• Suppose sampling from $P(\cdot)$ is hard.
• Suppose we can sample from a "simpler" proposal distribution $Q(\cdot)$ instead.
• If $Q$ dominates $P$ (i.e., $Q(x) > 0$ whenever $P(x) > 0$), we can sample from $Q$ and reweight:

\[
\langle f(X) \rangle = \int f(x) P(x) dx \\
= \int f(x) \frac{P(x)}{Q(x)} Q(x) dx \\
\approx \frac{1}{M} \sum_m f(x^m) \frac{P(x^m)}{Q(x^m)} \quad \text{where } x^m \sim Q(X) \\
= \frac{1}{M} \sum_m f(x^m) w^m
\]
Normalized importance sampling

• Suppose we can only evaluate $P'(x) = a P(x)$

Let $r(X) = \frac{P'(x)}{Q(x)} \Rightarrow \langle r(X) \rangle_Q = \int \frac{P'(x)}{Q(x)} Q(x) \, dx = \int P'(x) \, dx = \alpha$

\[
\langle f(X) \rangle_p = \int f(x) P(x) \, dx = \frac{1}{\alpha} \int f(x) \frac{P'(x)}{Q(x)} Q(x) \, dx
\]
\[
= \frac{\int f(x) r(x) Q(x) \, dx}{\int r(x) Q(x) \, dx}
\]
\[
\approx \frac{\sum_m f(x^m) r^m}{\sum_m r^m} \quad \text{where } x^m \sim Q(X)
\]
\[
= \sum_m f(x^m) w^m \quad \text{where } w^m = \frac{r^m}{\sum_m r^m}
\]
Weighted resampling

• Problem of importance sampling: performance depends on how well Q matches P
  • If $P(x)f(x)$ is strongly varying and has a significant proportion of its mass concentrated in a small region, $r_m$ will be dominated by a few samples

• Solution: use a heavy tail Q and weighted resampling

$$w^m = \frac{P(x^m)/Q(x^m)}{\sum_l P(x^l)/Q(x^l)} = \frac{r^m}{\sum_m r^m}$$
Limitations of Monte Carlo

• Direct sampling
  • Hard to get rare events in high-dimensional spaces
  • Infeasible for MRFs unless we know the normalizer $Z$

• Rejection sampling, Importance sampling
  • We need a good proposal $Q(x)$ that is not very different than $P(x)$

• How about we use an adaptive proposal?
Markov Chain Monte Carlo

- MCMC algorithms feature adaptive proposals
  - Instead of $Q(x')$ use $Q(x'|x)$ where $x'$ is the new state being sampled and $x$ is the previous sample
  - As $x$ changes $Q(x'|x)$ can also change
Metropolis-Hastings

• Draw a sample $x'$ from $Q(x'|x)$ where $x$ is the previous sample
• The new sample $x'$ is accepted or rejected with some probability $A(x'|x)$
  
  • Acceptance prob: $A(x'|x) = \min\left(1, \frac{P(x')Q(x|x')}{P(x)Q(x'|x)}\right)$

• $A(x'|x)$ is like a ration of importance sampling weights
  • $P(x')/Q(x'|x)$ is the importance weight for $x'$, $P(x)/Q(x|x')$ is the importance weight for $x$
  • We divide the importance weight for $x'$ by that of $x$
  • Notice that we only need to compute $P(x')/P(x)$ rather than $P(x')$ or $P(x)$

• $A(x'|x)$ ensures that after sufficiently many draws, our samples come from the true distribution.
Metropolis-Hastings

1. Initialize starting state $x^{(0)}$, set $t = 0$

2. **Burn-in:** while samples have “not converged”
   - $x = x^{(t)}$
   - $t = t + 1,$
   - sample $x^* \sim Q(x^*|x)$ \hspace{1mm} // draw from proposal
   - sample $u \sim \text{Uniform}(0,1)$ \hspace{1mm} // draw acceptance threshold
   - if $u < A(x^*|x) = \min\left(1, \frac{P(x^*)Q(x| x^*)}{P(x)Q(x^*|x)}\right)$
     - $x^{(t)} = x^*$ \hspace{1mm} // transition
   - else
     - $x^{(t)} = x$ \hspace{1mm} // stay in current state

- **Take samples from $P(x) =$** \hspace{1mm} : Reset $t=0$, for $t = 1:N$
  - $x(t+1) \xleftarrow{} \text{Draw sample } (x(t))$
Example of MH

• Let $Q(x' \mid x)$ be a Gaussian centered on $x$
• We are trying to sample from a bimodal $P(x)$
Example of MH

- Let $Q(x'|x)$ be a Gaussian centered on $x$
- We are trying to sample from a bimodal $P(x)$

$$A(x'|x) = \min \left( 1, \frac{P(x')Q(x|x')}{P(x)Q(x'|x)} \right)$$

Initialize $x^{(0)}$
Draw, accept $x^1$
Example of MH

- Let $Q(x' | x)$ be a Gaussian centered on $x$
- We are trying to sample from a bimodal $P(x)$

$$A(x' | x) = \min \left( 1, \frac{P(x')Q(x | x')}{P(x)Q(x' | x)} \right)$$
Example of MH

• Let $Q(x' | x)$ be a Gaussian centered on $x$
• We are trying to sample from a bimodal $P(x)$
Example of MH

• Let $Q(x' | x)$ be a Gaussian centered on $x$
• We are trying to sample from a bimodal $P(x)$

\[
A(x' | x) = \min \left( 1, \frac{P(x')Q(x | x')}{P(x)Q(x' | x)} \right)
\]
Example of MH

• Let $Q(x'|x)$ be a Gaussian centered on $x$
• We are trying to sample from a bimodal $P(x)$
Example of MH

- Let $Q(x'|x)$ be a Gaussian centered on $x$
- We are trying to sample from a bimodal $P(x)$
Example of MH

• Let $Q(x'|x)$ be a Gaussian centered on $x$
• We are trying to sample from a bimodal $P(x)$
Some theoretical aspects of MCMC

• The MH algorithm has a burn-in period
  • Initial samples are not truly from P

• Why are the MH samples guaranteed to be from P(x)?
  • The proposal Q(x’|x) keeps changing with the value of x; how do we know the samples will eventually come from P(x)?

• Why Markov Chain?
Markov Chains

• A Markov Chain is a sequence of random variables $x_1, x_2, \ldots, x_N$ with the Markov Property

$$P(x^{(n)} = x | x^{(1)}, \ldots, x^{(n-1)}) = P(x^{(n)} = x | x^{(n-1)})$$

• The right hand side is the transition kernel. Next state depends only on preceding state

• Let’s assume the kernel is fixed with time.
MC Concepts

- Probability distributions over states: $\pi^{(t)}(x)$ is a distribution over the state of the system $x$, at time $t$
  - When dealing with MCs, we don't think of the system as being in one state, but as having a distribution over states
  - For graphical models, remember that $x$ represents all variables
- Transitions: recall that states transition from $x^{(t)}$ to $x^{(t+1)}$ according to the transition kernel $T(x' \mid x)$. We can also transition entire distributions:
  $$\pi^{(t+1)}(x') = \sum_x \pi^{(t)}(x) T(x' \mid x)$$
  - At time $t$, state $x$ has probability mass $\pi^{(t)}(x)$. The transition probability redistributes this mass to other states $x'$.
- Stationary distributions: $\pi(x)$ is stationary if it does not change under the transition kernel:
  $$\pi(x') = \sum_x \pi(x) T(x' \mid x) \quad \text{for all } x'$$
MC Concepts

• Stationary distributions are of great importance in MCMC. Some notions
  • **Irreducible**: an MC is irreducible if you can get from any state $x$ to any other state $x'$ with probability $x > 0$ in a finite number of steps
  • **Aperiodic**: an MC is aperiodic if you can return to any state $x$ at any time
  • **Ergodic (or regular)**: an MC is ergodic if it is irreducible and aperiodic

• Ergodicity is important: it implies you can reach the stationary distribution no matter the initial distribution.
MC Concepts

- Reversible (detailed balance): an MC is reversible if there exists a distribution $\pi(x)$ such that the detailed balance condition holds

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

- Reversible MCs always have a stationary distribution

\[
\pi(x')T(x \mid x') = \pi(x)T(x' \mid x) \\
\sum_x \pi(x')T(x \mid x') = \sum_x \pi(x)T(x' \mid x) \\
\pi(x') = \sum_x \pi(x)T(x' \mid x) \\
\text{The last line is the definition of a stationary distribution!}
\]
Why does MH work?

• We draw a sample $x'$ according to $Q(x'|x)$ and then accept/reject according to $A(x'|x)$. Hence the transition kernel is:

$$T(x'|x) = Q(x'|x)A(x'|x)$$

• We can prove that MH satisfies detailed balance.

Recall that

$$A(x'|x) = \min\left(1, \frac{P(x')Q(x|x')}{P(x)Q(x'|x)}\right)$$

Notice this implies the following:

If $A(x'|x) \leq 1$ then $\frac{P(x)Q(x'|x)}{P(x')Q(x|x')} \geq 1$ and thus $A(x|x') = 1$
Why does MH work?

• Now suppose \( A(x'|x) < 1 \) and \( A(x|x') = 1 \). We have

\[
A(x'|x) = \frac{P(x')Q(x'|x')}{P(x)Q(x|x')}
\]

\[
P(x)Q(x'|x)A(x'|x) = P(x')Q(x|x')
\]

\[
P(x)Q(x'|x)A(x'|x) = P(x')Q(x|x')A(x|x')
\]

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

• This is the detailed balance condition:
  • The MH algorithm leads to a stationary distribution \( P(x) \)
  • We defined \( P(x) \) to be the true distribution of \( x \)
  • Thus, MH eventually converges to the true distribution
Gibbs Sampling

• Gibbs Sampling is an MCMC algorithm that samples each random variable of a graphical model, one at a time

• GS is fairly easy to derive for many graphical models

• GS has reasonable computation and memory requirements (because we sample one r.v. at a time)
Gibbs Sampling Algorithm

1. Suppose the graphical model contains variables $x_1, \ldots, x_n$
2. Initialize starting values for $x_1, \ldots, x_n$
3. Do until convergence:
   1. Pick an ordering of the $n$ variables (can be fixed or random)
   2. For each variable $x_i$ in order:
      1. Sample $x$ from $P(x_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, i.e. the conditional distribution of $x_i$ given the current values of all other variables
      2. Update $x_i \leftarrow x$
Gibbs Sampling Example

**Variables Factors**

- **If we set v₁ to True, we are rewarded by 5 points!**
  \[ f₁(a) = \begin{cases} 5, & a = \text{True} \\ 0, & \text{otherwise} \end{cases} \]

- **If we set v₂ and v₃ to the same, we get 10 more points!**
  \[ f₂(a,b) = \begin{cases} 10, & a = b \\ 0, & \text{otherwise} \end{cases} \]

**Probability**

\[ \infty \]
\[ \exp\{\text{total points}\} \]

---

**Gibbs Sampling: A Descriptive Tutorial**

1. Initialize variables with a random assignment. **T** **F**
2. For each random variable:
   2.1 Calculate the points we earn for each assignment:
      - e.g., \( v₁ = \text{T} \) ⇒ 0 points
      - e.g., \( v₁ = \text{F} \) ⇒ 10 points
   2.2 Randomly pick one assignment:
      - e.g., \( P(v₂ = \text{T}) = \frac{\exp(0)}{\exp(0) + \exp(10)} \)
      - \( P(v₂ = \text{F}) = \frac{\exp(10)}{\exp(0) + \exp(10)} \)
3. Generate one sample. Goto 2 if we want more samples.

**Billions!**
Parallel Gibbs Sampling

- Run Gibbs independently on full copies of the same model
- Fewer iterations per copy
- More samples means more accurate marginals

Complete Model Copies

Run sequential Gibbs

Data to materialize factor graph

Variable Tally
Parallel Gibbs Sampling

- Compute a k-coloring of the factor graph
- Sample all variables with same color in parallel
- Load balancing is a key challenge
Summary

• Sampling can be easy to implement but we can get poor quality samples
  • We need a good proposal distribution

• 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 $Q(x'|x)$

• Gibbs sampling sets the proposal $Q(x'|x)$ to the conditional $P(x'|x)$
  • Acceptance rate is always 1 but this means slow exploration

• Burn-in is an art!