**Readings: K&F 12.1, 12.2, 12.3, 12.4**

# Approximate Inference I

Lecture 15 – May 16, 2011
CSE 515, Statistical Methods, Spring 2011

Instructor: Su-In Lee
University of Washington, Seattle

## Inference

- Inference is widely used,
  - Probabilistic queries: Given **E**=**e**, what is P(**X**|**E**=**e**)?
    - For example, given **E**=image what is P(**X**=labels|**E**=image)=?

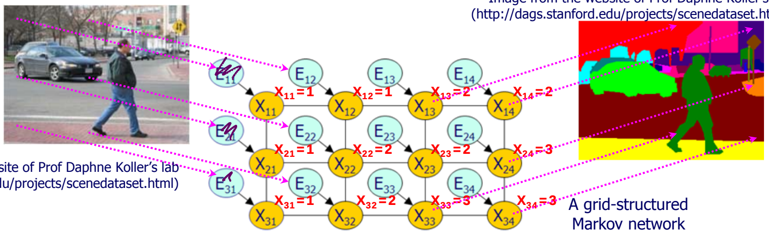


Image from the website of Prof Daphne Koller's lab (http://dags.stanford.edu/projects/scenedataset.html)

Image from the website of Prof Daphne Koller's lab (http://dags.stanford.edu/projects/scenedataset.html)

A grid-structured Markov network

  - Learning parameters with partially observed data
    - EM, gradient ascent, etc

- but exact inference is expensive
  - O(Nn), where N=max$_i$|Val(**C**$_i$)|
  - Exponential blowup can be in N$_i$ which for factor i can be $v^m$ if factor i has m variables with v values each.

# Approximate Inference Overview

**Particle-based approximate inference**

- Full particle methods
  - Sampling methods
  - Deterministic particle generation
- Distributional particles
- Applications & Simulation: data association

**Today…**

**Global approximate inference**

- Inference as optimization
- Generalized Belief Propagation
- Propagation with approximate messages
- Structured variational approximations

# Particle-based Approximate Inference

General framework

- Generate particles (samples) x[1],…,x[M] from P
- Estimate function f by $E_P(f) \approx \frac{1}{M}\sum_{m=1}^{M} f(x[m])$

$M \to \infty$

- For a function f(**X**)=**1**(**Y**=**y**)
- P(**Y**=**y**) can be written as $E_P[f]$

$x.$

- How particles are generated?
  - Use forward sampling ←
  - Use likelihood weighting sampling ←

# Particle-Based Methods Overview

- Full particle methods
  - Sampling methods
    - Forward sampling, Likelihood weighting
    - ⇨ (Un-normalized/normalized) Importance sampling
    - Markov chain Monte Carlo
  - Deterministic particle generation
- Distributional particles

# Unnormalized Importance Sampling

- For any distribution Q(**X**), since:

$$E_{P(\mathbf{X})}[f(\mathbf{X})] = \sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})P(\mathbf{x}) = \sum_{\mathbf{x}\in\mathbf{X}} Q(\mathbf{x}) f(\mathbf{x})\frac{P(\mathbf{x})}{Q(\mathbf{x})} = E_{Q(\mathbf{X})}\left[f(\mathbf{X})\frac{P(\mathbf{X})}{Q(\mathbf{X})}\right]$$

- We can estimate f(x) by generating samples from Q and estimating:

$$E_P(f) \approx \frac{1}{M}\sum_{m=1}^{M} f(x[m])\frac{P(x[m])}{Q(x[m])}$$

Target distribution

Proposal distribution

  - Can show that estimator variance decreases with more samples M    M→∞
  - Can show that Q=P is the lowest variance estimator

M

# Normalized Importance Sampling

- Un-normalized importance sampling assumes known P
  - Usually we know P up to a constant: P = P'/α, where α=Σ$_x$P'(x)
    - Example: Posterior distribution P(**X**|**E**=**e**) = P(**X**,**E**=**e**) / α where α=P(**E**=**e**)
- We can estimate α by: $E_{Q(\mathbf{X})}\left[\frac{P'(\mathbf{X})}{Q(\mathbf{X})}\right] = \sum_{\mathbf{x}\in\mathbf{X}} Q(\mathbf{x})\frac{P(\mathbf{x})}{Q(\mathbf{x})} = \sum_{\mathbf{x}\in\mathbf{X}} P'(\mathbf{x}) = \alpha$
- Thus

$$\begin{aligned}
E_{P(\mathbf{X})}[f(\mathbf{X})] &= \sum_x P(x) f(x) \\
&= \sum_x Q(x) f(x) \underset{P'/\alpha}{P(x)} / Q(x) \\
&= \frac{1}{\alpha}\sum_x Q(x) f(x) P'(x) / Q(x) \\
&= \frac{1}{\alpha} E_{Q(\mathbf{X})}[f(\mathbf{X})P'(x)/Q(x)] \\
&= \frac{E_{Q(\mathbf{X})}[f(\mathbf{X})P'(x)/Q(x)]}{E_{Q(\mathbf{X})}[P'(x)/Q(x)]}
\end{aligned}$$

# Normalized Importance Sampling

$$E_{P(\mathbf{X})}[f(\mathbf{X})] = \frac{E_{Q(\mathbf{X})}[f(\mathbf{X})P'(x)/Q(x)]}{E_{Q(\mathbf{X})}[P'(x)/Q(x)]}$$

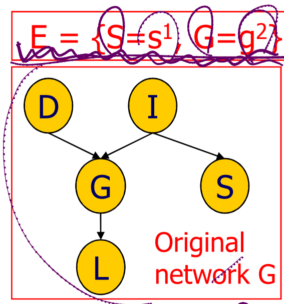
- Given M samples from Q, normalized sampling estimates function f by:

$$E_P(f) \approx \frac{\cancel{\frac{1}{M}}\sum_{m=1}^{M} f(x[m])P'(x[m])/Q(x[m])}{\cancel{\frac{1}{M}}\sum_{m=1}^{M} P'(x[m])/Q(x[m])}$$

# Importance Sampling for BayesNets

- Let's think about how to generate particles to estimate P(**Y**|**e**) $Y \subseteq X \rightarrow P(Y|E=e)$
- Define *mutilated network* $G_{E=e}$ as:
  - Nodes X∈E have no parents in $G_{E=e}$
  - Nodes X∈E have CPD that is 1 for X=E[X] and 0 otherwise
  - The parents and CPDs for all other variables are unchanged

E = {S=s$^1$, G=g$^2$}



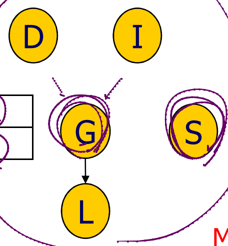
Target distribution P'(**X**,e)

p(Y|e)

| d$^0$ | d$^1$ |
|---|---|
| 0.4 | 0.6 |

| i$^0$ | i$^1$ |
|---|---|
| 0.7 | 0.3 |

| g$^0$ | g$^1$ | g$^2$ |
|---|---|---|
| 0 | 0 | 1 |



| s$^0$ | s$^1$ |
|---|---|
| 0 | 1 |

| *G* | l$^0$ | l$^1$ |
|---|---|---|
| g$^0$ | 0.1 | 0.9 |
| g$^1$ | 0.4 | 0.6 |
| g$^2$ | 0.99 | 0.01 |

Mutilated network $G_{E=e}$

Proposal distribution Q(**X**,e)=$P_{G_{E=e}}$(**X**,e)

# Likelihood Weighting as IS

- Target distribution P'(**X**,**e**)
- Proposal distribution Q(**X**,**e**)=$P_{G_{E=e}}$(**X**,**e**)

- Claim: Likelihood weighting is precisely normalized importance sampling with the above distributions

- Proof sketch:
  - LW estimate: $P(y|e) \approx \dfrac{\sum_{m=1}^{M} w[m]\overset{f}{\mathbf{1}\{x[m](Y)=y\}}}{\sum_{m=1}^{M} w[m]}$
  - IS estimate: $P(y|e) \approx \dfrac{\sum_{m=1}^{M} P'(x[m])/Q(x[m])\overset{f}{\mathbf{1}\{x[m](Y)=y\}}}{\sum_{m=1}^{M} P'(x[m])/Q(x[m])}$
  - But w[m] in LW precisely comes out to be P'(x[m])/Q(x[m])
    - Since

$$\frac{P'(x[m])}{Q(x[m])} = \frac{\prod_{e_i \in \mathbf{E}} P(e_i \mid Pa(e_i)) \prod_{X_i \notin \mathbf{E}} \cancel{P(X_i \mid Pa(X_i))}}{\prod_{X_i \notin \mathbf{E}} \cancel{P(X_i \mid Pa(X_i))}} = \prod_{e_i \in \mathbf{E}} P(e_i \mid Pa(e_i)) = w[m]$$

## Particle-Based Methods Overview

- Full particle methods
  - Sampling methods
    - Forward sampling
    - Importance sampling
    - ⇨ Markov chain Monte Carlo (MCMC)
  - Deterministic particle generation
- Distributional particles

## Markov Chain Monte Carlo

- Limitations of importance sampling
  - An evidence node affects the sampling only for nodes that are its descendants. E
  - The effect on nodes that are non-descendants is accounted for only by the weights w's.
  - What if much of the evidence is at the leaves of the network?
    - We are essentially sampling from the prior distribution $P(\mathbf{X})$, which is often very far from the desired posterior $P(\mathbf{X}|\mathbf{E}=\mathbf{e})$.
    - → Decreases the accuracy of the estimates
- An alternative sampling approach (MCMC)
  - General idea: Define a sampling process that is guaranteed to converge to taking samples from the posterior distribution of interest $P(\mathbf{X}|\mathbf{E}=\mathbf{e})$
  - Generate samples from the sampling process
  - Estimate f(X) from the samples

# Markov Chains

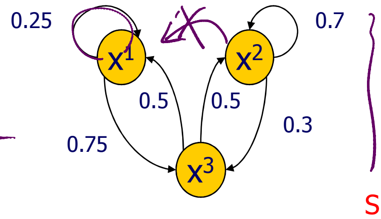
- A Markov chain consists of
  - A state space Val(**X**)
  - Transition probability T(x→x') of going from state x to x'
- Distribution over subsequent states is defined as

Distribution over the "next" state

Distribution over the "current" state

$$P^{(t+1)}(X^{(t+1)} = x') = \sum_{x \in Val(X)} P^{(t)}(X^{(t)} = x)T(x \to x')$$

$$p^{(t+1)}(X^{(t+1)} = x^1)$$
$$= p^{(t)}(X^{(t)} = x^1)\cdot 0.25$$
$$+ p^{(t)}(X^{(t)} = x^3)\cdot 0.5$$



Simple Markov chain

# Stationary Distribution

Distribution over the "next" state

Distribution over the "current" state

$$P^{(t+1)}(X^{(t+1)} = x') = \sum_{x \in Val(X)} P^{(t)}(X^{(t)} = x)T(x \to x')$$

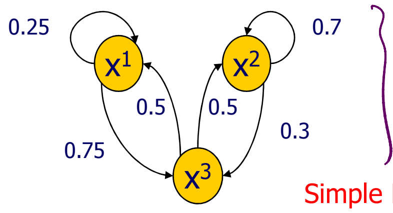
- A distribution π(X) is a stationary distribution for a Markov chain T if it satisfies

$$\pi(X = x') = \sum_{x \in Val(X)} \pi(X = x)T(x \to x')$$

$$\begin{cases} \pi(x^1) = 0.25\pi(x^1) + 0.5\pi(x^3) \\ \pi(x^2) = 0.7\pi(x^2) + 0.5\pi(x^3) \\ \pi(x^3) = 0.75\pi(x^1) + 0.3\pi(x^2) \end{cases}$$

⇓

$$\begin{pmatrix} \pi(x^1) = 0.2 \\ \pi(x^2) = 0.5 \\ \pi(x^3) = 0.3 \end{pmatrix}$$



Simple Markov chain

## Markov Chains & Stationary Dist.

- A Markov chain is regular if there is k such that for every x, x'∈Val(X), the probability of getting from x to x' in exactly k steps is greater than zero

$x \rightarrow x'$

- Theorem: A finite state Markov chain T has a unique stationary distribution if and only if it is regular

- Goal: Define a Markov chain whose stationary distribution is $P(\mathbf{X}|\mathbf{e})$

## Gibbs Sampling ←

- States $E=e$
  - Legal (=consistent with evidence) assignments to variables
- Transition probability
  - $T = T_1 \cdot \ldots \cdot T_k$ ← $X_i$
  - For each variable $X_i$, let $\mathbf{X}_{-i}$ be $\mathbf{X}-\{X_i\}$. Say that $\mathbf{X}_{-i} = \mathbf{x}_{-i}$.
    $T_i = T((\mathbf{x}_{-i}, x_i) \rightarrow (\mathbf{x}_{-i}, x'_i)) = P(x'_i|\mathbf{x}_{-i})$

    $X_i$ ↓ $X_{-i} = x_{-i}$
- Claim: $P(\mathbf{X}|\mathbf{e})$ is a stationary distribution to the chain

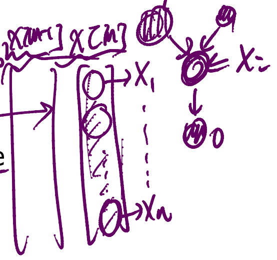
# Gibbs Sampling

**A couple slides ago…**

- A Markov chain is regular if there is k such that for every x, x′∈Val(X), the probability of getting from x to x′ in exactly k steps is greater than zero
- → ***has a stationary distribution***

- Gibbs-sampling Markov chain is regular if:
  - Bayesian networks: all CPDs are strictly positive
  - Markov networks: all clique potentials are strictly positive

- Gibbs sampling procedure (generating one sample x[m]) x[m]
  - Set x[m]=x[m-1]
  - For each variable $X_i \in$ **X-E**
    - Set $\mathbf{x}_{-i}$ = x[m](**X**-$X_i$)
    - Sample from $P(X_i \mid \mathbf{x}_{-i})$ P. graph
    - Save sample x[m]($X_i$) = sampled value
  - Return x[m]



  - Note: $P(X_i \mid \mathbf{x}_{-i})$ is easily computed from Markov blanket

# Gibbs Sampling

- How do we evaluate $P(X_i \mid \mathbf{x}_{-i})$ ?
  - Again, let $\mathbf{x}_{-i} = \{x_1,\dots,x_{i-1},x_{i+1},\dots,x_n\}$

- Let's say that P factorizes as

$$P(\mathbf{X}) = \frac{1}{Z}\prod_j \phi_j(\mathbf{D}_j) = \frac{1}{Z}\prod_{j:X_i\in\mathbf{D}_j}\phi_j(\mathbf{D}_j)\prod_{j:X_i\notin\mathbf{D}_j}\phi_j(\mathbf{D}_j)$$

- Let $\mathbf{x}_{j,-i}$ denote the assignment in $\mathbf{x}_{-i}$ to $\mathbf{D}_j$-{$X_i$},

$$P(x_i'|\mathbf{x}_{-i}) = \frac{P(x_i',\mathbf{x}_{-i})}{\sum_{x_i''}P(x_i'',\mathbf{x}_{-i})} \quad p(X_{-i})$$

$$= \frac{\prod_{j:X_i\in\mathbf{D}_j}\phi_j(x_i',\mathbf{x}_{j,-i})\prod_{j:X_i\notin\mathbf{D}_j}\phi_j(\mathbf{x}_{j,-i})}{\sum_{x_i''}\prod_{j:X_i\in\mathbf{D}_j}\phi_j(x_i'',\mathbf{x}_{j,-i})\prod_{j:X_i\notin\mathbf{D}_j}\phi_j(\mathbf{x}_{j,-i})} = \frac{\prod_{j:X_i\in\mathbf{D}_j}\phi_j(x_i',\mathbf{x}_{j,-i})}{\sum_{x_i'}\prod_{j:X_i\in\mathbf{D}_j}\phi_j(x_i'',\mathbf{x}_{j,-i})}$$

  - The last term uses only the factors involving $X_i$.

## Gibbs Sampling in Practice

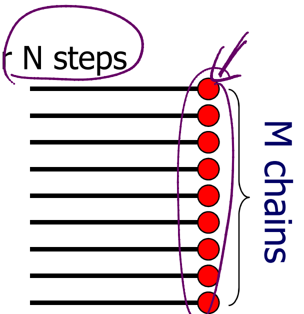
- We need to wait until the burn-in time has ended – number of steps until we take samples from the chain
  - We want to wait until the sampling distribution is close to the stationary distribution P(X|e)
  - Hard to provide bounds in general for this 'mixing time'
  - Once the burn-in time ended, all samples are from the stationary distribution

x[m] ← x[m+1]

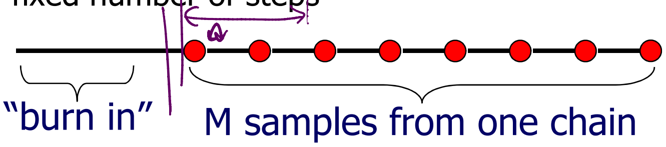
- Note: after the burn-in time, samples are correlated. Consecutive samples from the same trajectory are correlated.
- Since no theoretical guarantees exist, application of Markov chains is somewhat of an art
  - We can evaluate burn-in time by comparing the estimates $\hat{E}_{k}(f) \approx \frac{1}{M}\sum_{m=1}^{M} f(x^{k}[m],e)$ from multiple chains 1,…,K

## Sampling Strategy

- How do we collect the samples?
- Strategy I:
  - Run the chain M times, each run for N steps
    - Each run starts from a different state
  - Return the last state in each run



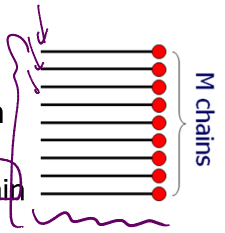
- Strategy II:
  - Run one chain for a long time
  - After some "burn in" period, sample points every some fixed number of steps



## Comparing Strategies

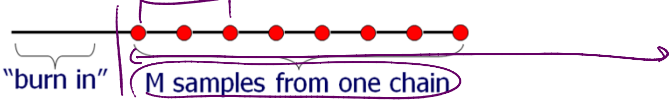
Strategy I:

- Better chance of "covering" the space of points, especially if the chain is slow to reach the stationary distribution
- Have to perform "burn in" steps for each chain



Strategy II:

- Perform "burn in" only once
- Samples might be correlated (although only weakly)



Hybrid strategy:

- Run several chains, and sample few samples from each
- Combines benefits of both strategies

## Particle-Based Methods Overview

- Full particle methods
  - Sampling methods
    - Forward sampling
    - Importance sampling
    - Markov chain Monte Carlo
  - Deterministic particle generation
- Distributional particles

## Deterministic Search Methods

- Idea: if the $\overset{P}{\text{distribution}}$ is dominated by a small set of instances, it suffices to consider only them for approximating the function f
  - For instances that we generate x[1],...x[M], estimate is: $\sum_m P(x[m]) \approx 1$

$$E_P(f) \approx \sum_{m=1}^{M} f(x[m])P(x[m])$$

  - Note: we can obtain lower and upper bounds by examining the part of the probability mass covered by ΣP(x[m])

$\sum_m P(x[m]) < 1$

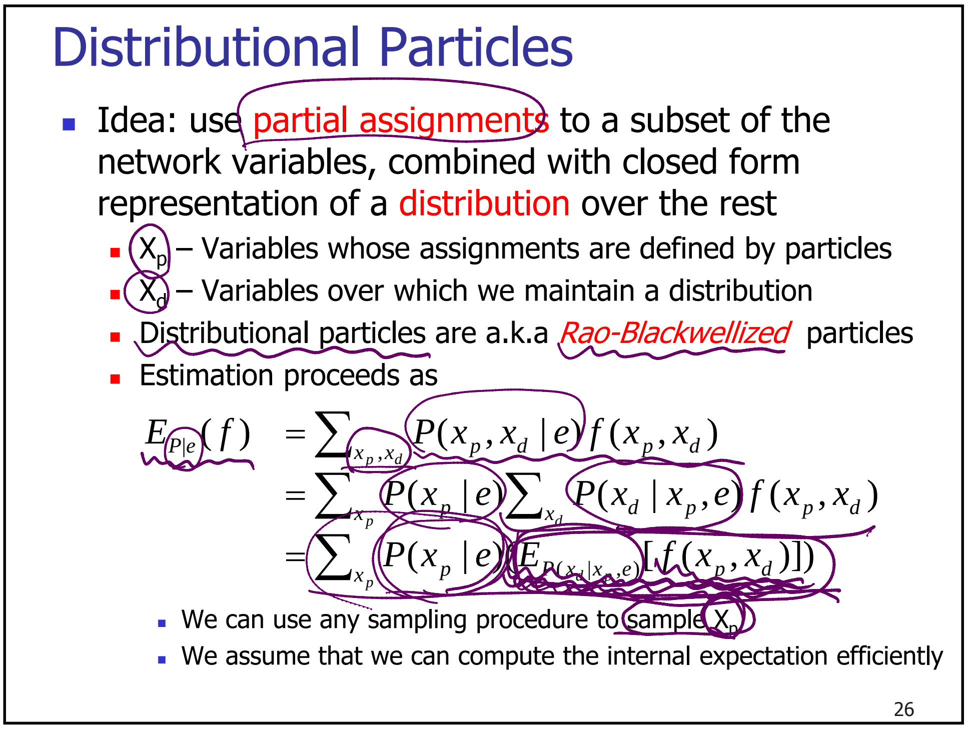
$$\sum_{m=1}^{M} f(x[m])P(x[m]) \le E_P(f) \le \left(1-\sum_{m=1}^{M}(1-f(x[m]))P(x[m])\right)$$

Sum of probabilities that f is 0.

- Key: how can we enumerate highly probably instantiations?
  - Note: the single most probably instantiation is MPE which itself is an NP-hard problem

## Particle-Based Methods Overview

- Full particle methods
  - Sampling methods
    - Forward sampling
    - Importance sampling
    - Markov chain Monte Carlo
  - Deterministic particle generation
- Distributional particles
  - Until now, we discussed the cases where we are given particles that have full assignment of values on all variables.
  - Can we use a partial assignment to estimate f?



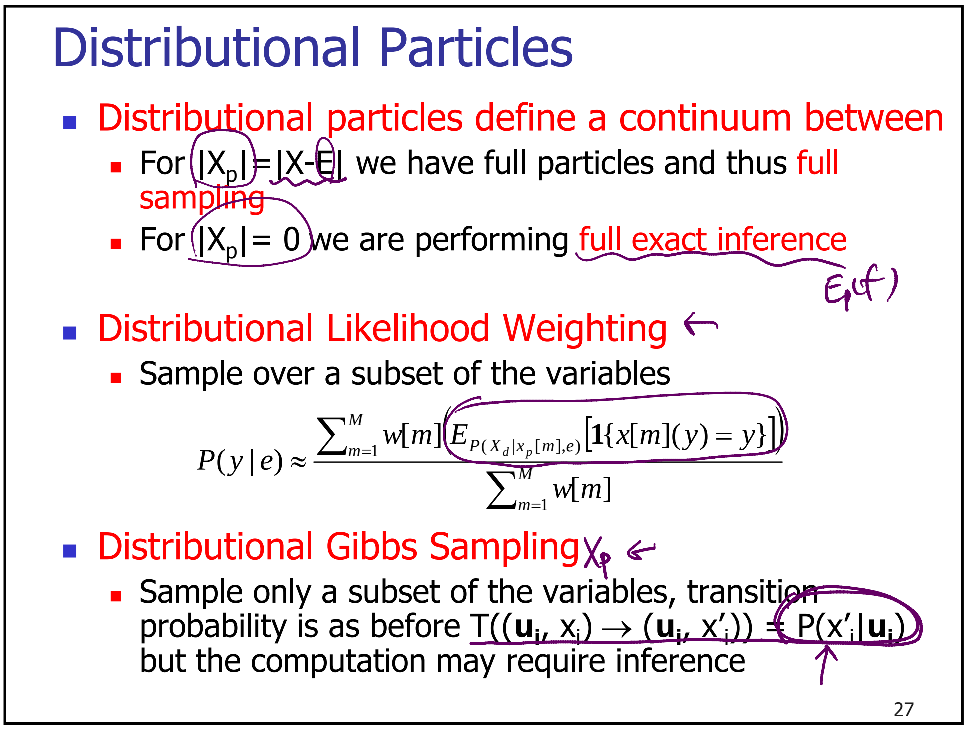
## Distributional Particles

- Idea: use partial assignments to a subset of the network variables, combined with closed form representation of a distribution over the rest
  - $X_p$ – Variables whose assignments are defined by particles
  - $X_d$ – Variables over which we maintain a distribution
  - Distributional particles are a.k.a *Rao-Blackwellized* particles
  - Estimation proceeds as

$$\begin{aligned} E_{P|e}(f) &= \sum_{x_p,x_d} P(x_p,x_d \mid e) f(x_p,x_d) \\ &= \sum_{x_p} P(x_p \mid e) \sum_{x_d} P(x_d \mid x_p,e) f(x_p,x_d) \\ &= \sum_{x_p} P(x_p \mid e) E_{P(x_d \mid x_p,e)}[f(x_p,x_d)] \end{aligned}$$

- We can use any sampling procedure to sample $X_p$
- We assume that we can compute the internal expectation efficiently

26



## Distributional Particles

- Distributional particles define a continuum between
  - For $|X_p|=|X-E|$ we have full particles and thus full sampling
  - For $|X_p|= 0$ we are performing full exact inference

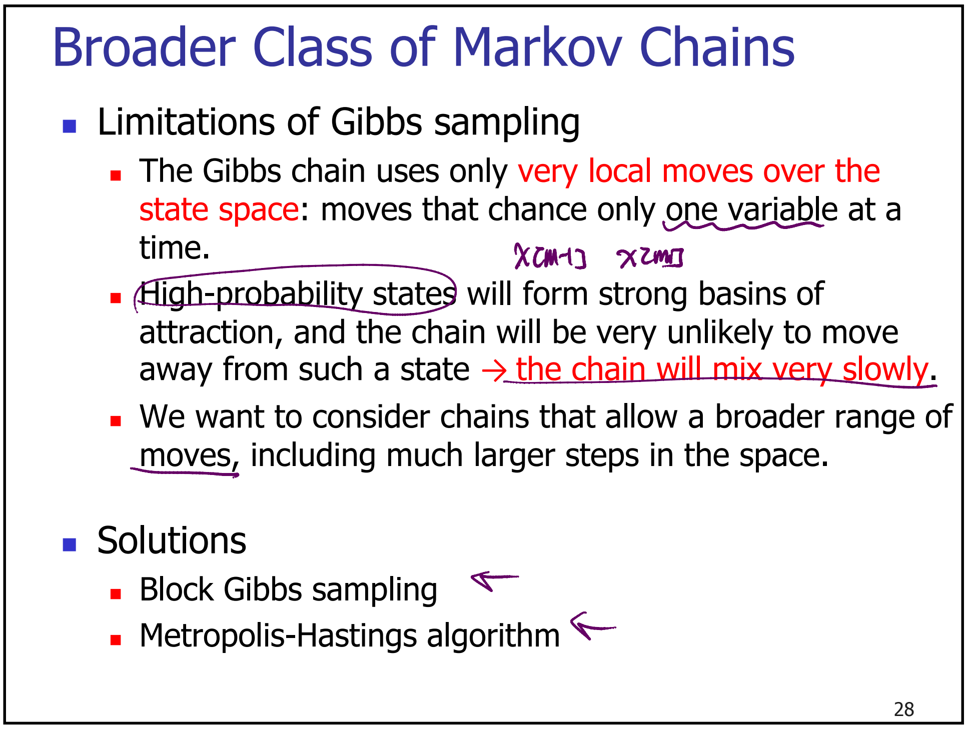
$E_p(f)$

- Distributional Likelihood Weighting ←
  - Sample over a subset of the variables

$$P(y \mid e) \approx \frac{\sum_{m=1}^{M} w[m] \left( E_{P(X_d \mid x_p[m],e)}[\mathbf{1}\{x[m](y)=y\}] \right)}{\sum_{m=1}^{M} w[m]}$$

- Distributional Gibbs Sampling $X_p$ ←
  - Sample only a subset of the variables, transition probability is as before $T((\mathbf{u}_i, x_i) \rightarrow (\mathbf{u}_i, x'_i)) = P(x'_i \mid \mathbf{u}_i)$ but the computation may require inference

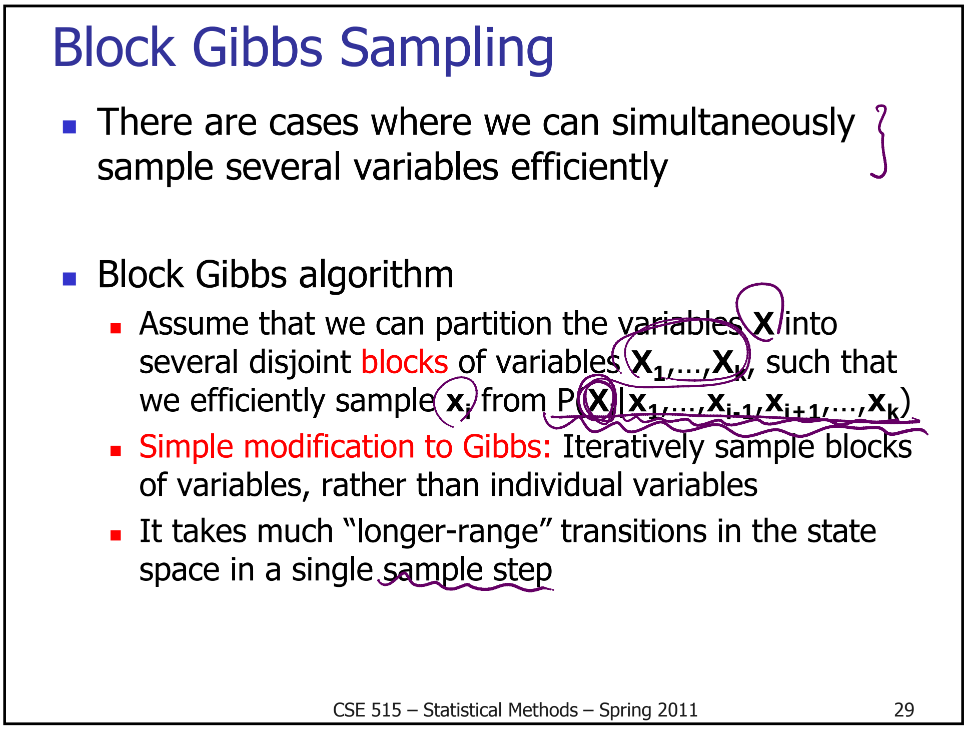
27



## Broader Class of Markov Chains

- Limitations of Gibbs sampling
  - The Gibbs chain uses only very local moves over the state space: moves that chance only one variable at a time.

    $x[m-1]$ $x[m]$
  - High-probability states will form strong basins of attraction, and the chain will be very unlikely to move away from such a state → the chain will mix very slowly.
  - We want to consider chains that allow a broader range of moves, including much larger steps in the space.
- Solutions
  - Block Gibbs sampling
  - Metropolis-Hastings algorithm



## Block Gibbs Sampling

- There are cases where we can simultaneously sample several variables efficiently
- Block Gibbs algorithm
  - Assume that we can partition the variables $\mathbf{X}$ into several disjoint blocks of variables $\mathbf{X}_1,\ldots,\mathbf{X}_k$, such that we efficiently sample $\mathbf{x}_i$ from $P(\mathbf{X}_i|\mathbf{x}_1,\ldots,\mathbf{x}_{i-1},\mathbf{x}_{i+1},\ldots,\mathbf{x}_k)$
  - Simple modification to Gibbs: Iteratively sample blocks of variables, rather than individual variables
  - It takes much "longer-range" transitions in the state space in a single sample step

# Metropolis-Hastings algorithm (1/3)

- Unlike the Gibbs chain, M-H algorithm does not assume that we can generate next-state samples from a particular target distribution
  - It uses the idea of a proposal distribution Q that we have already seen in the case of IS.
- Basic ideas
  - We sample from a different distribution Q and then correct for the resulting error.
  - However, unlike importance sampling, we do not want to keep track of importance weights.
    - They are going to decay exponentially with the number of transitions, leading to a whole slew of problems
  - Instead, we randomly choose whether to accept the proposed transition, with a probability that corrects for the discrepancy between Q and the target distribution P

# Metropolis-Hastings algorithm (2/3)

- Let our proposal distribution $T^Q$ define a transition model over our state space
  - For each state x, $T^Q$ defines a distribution over possible successor states in Val(**X**), from which we select randomly a candidate next state x'
- We can either accept the proposal and transition to the new state x', or reject it and stay at x.
  - For each states x, x' we have an acceptance probability A(x→x'). Then, the actual transition model of the Markov chain is

$$\begin{aligned} &T(x \to x') = T^Q(x \to x')A(x \to x') \qquad x \neq x' \\ &T(x \to x) = T^Q(x \to x) + \sum_{x \neq x'} T^Q(x \to x')\big(1 - A(x \to x')\big) \end{aligned}$$

  - We can show that with the following acceptance probabilities (and the regularity assumption), a chain T has the unique stationary distribution (proof in K&F page 516)

$$A(x \to x') = \min\left[1, \frac{\pi(x')T^Q(x' \to x)}{\pi(x)T^Q(x \to x')}\right]$$

## Metropolis-Hastings algorithm (3/3)

$$A(x \to x') = \min\left[1, \frac{\pi(x')T^Q(x' \to x)}{\pi(x)T^Q(x \to x')}\right]$$

- The M-H algorithm has a particularly natural implementation in the context of graphical models.
  - Each local transition model $T_i$ is defined via an associated proposal distribution $T_i^{Q_i}$, and the acceptance probability for this chain has the form:

$$A(\mathbf{x}_{-i}, x_i \to \mathbf{x}_{-i}, x_i') = \min\left[1, \frac{P(\mathbf{x}_{-i}, x_i')T_i^{Q_i}(\mathbf{x}_{-i}, x_i' \to \mathbf{x}_{-i}, x_i)}{P(\mathbf{x}_{-i}, x_i)T_i^{Q_i}(\mathbf{x}_{-i}, x_i \to \mathbf{x}_{-i}, x_i')}\right]$$

- The proposal distributions are usually fairly simple, so it is easy to compute their ratios.
  - In the case of graphical models, the first ratio can also be computed easily:

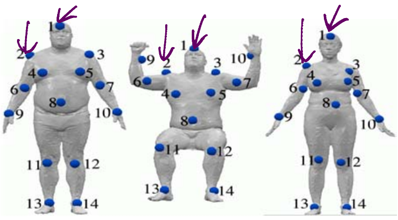
$$\frac{P(\mathbf{x}_{-i}, x_i')}{P(\mathbf{x}_{-i}, x_i)} = \frac{P(x_i' \mid \mathbf{x}_{-i})}{P(x_i \mid \mathbf{x}_{-i})}$$

Similarly to Gibbs sampling, $x_{-i}$ can be reduced to the Markov blanket of $X_i$

**So far, we discussed various sampling strategies. Let me talk about one example of real applications and "show" you how different sampling methods work in a toy model of the application.**

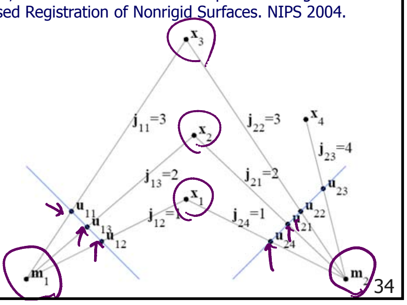
# Application & Simulation: Data Association

- Data association problem (aka correspondence problem)
  - Relating sensor measurements to parameters in the model that is being learned.
- For example,
  - We want to find a correspondence between a set of (automatically selected) landmarks on different 3D scans of human bodies.



D. Anguelov, et al. The Correlated Correspondence Algorithm for Unsupervised Registration of Nonrigid Surfaces. NIPS 2004.

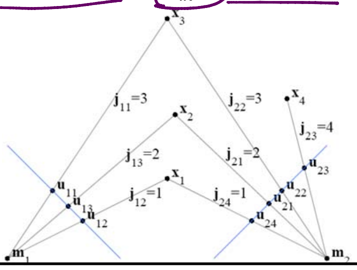
  - Given a set of image measurements **U**={$u_{ik}$ for image i and measurement k}, we want to know which of the 3D objects **X**={$x_j$, 1≤j≤n} each measurement corresponds to.



F. Dellaert, SM. Seitz, CE. Thorpe and S. Thrun. *EM, MCMC, and Chain Flipping for Structure from Motion with Unknown Correspondence.* Machine Learning 2003.

# Application & Simulation: Data Association

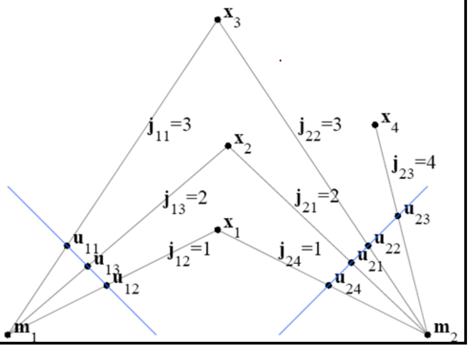
- Data association problem (aka correspondence problem)
  - Relating sensor measurements to parameters in the model that is being learned.
- Example problem
  - Given a set of image measurements **U**={$u_{ik}$ for image i and measurement k}, we want to know which of the 3D objects **X**={$x_j$, 1≤j≤n} each measurement corresponds to.
  - To model the correspondence between measurements $u_{ik}$ and 3D object $x_j$, we introduce a correspondence vector **J**.
  - For each measurement $u_{ik}$, the vector **J** contains an indicator variable $j_{ik}$ indicating that $u_{ik}$ is assigned to the $j_{ik}$-th object x.



F. Dellaert, SM. Seitz, CE. Thorpe and S. Thrun. *EM, MCMC, and Chain Flipping for Structure from Motion with Unknown Correspondence.* Machine Learning 2003.

# Applications: Data Association

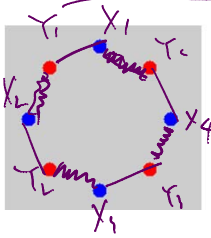
- Maximize $L(\Theta;\mathbf{U}) = \sum_J L(\Theta;\mathbf{U},\mathbf{J})$
  - **J**'s are hidden variables → EM algorithm
  - The E-step requires inference $P(\mathbf{J} \mid \mathbf{U}, \Theta^{(t)})$
  - This problem has been tackled using range of models and a variety of inference methods including MCMC
  - Efficiency is important and we will "see" how various MCMC-based methods show varying performance



F. Dellaert, SM. Seitz, CE. Thorpe and S. Thrun. *EM, MCMC, and Chain Flipping for Structure from Motion with Unknown Correspondence.* Machine Learning 2003.

# Toy Model for Data Association

- Blue dots: variables, $X_i$ (i=1,2,3,4)
- Red dots: observations $Y_j$ (j=1,2,3,4)
- Correspondence variable $C_i$={1,2,3,4} assigns to $X_i$ one of the observations



distance

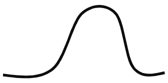
$$P(\mathbf{X}) \propto \exp\left(-\sum_{i=1}^{4} \left\| \frac{x_i - y_{C_i}}{\sigma} \right\|^2\right) \quad \text{if every } x_i \text{ has a different value}$$

$$P(\mathbf{X}) = 0 \quad \text{otherwise}$$

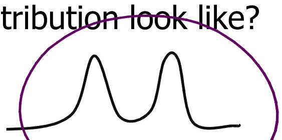
$X_1 - Y_1$

$Y_L - Y_L$

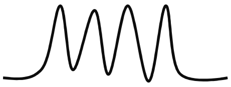
- What does the distribution look like?



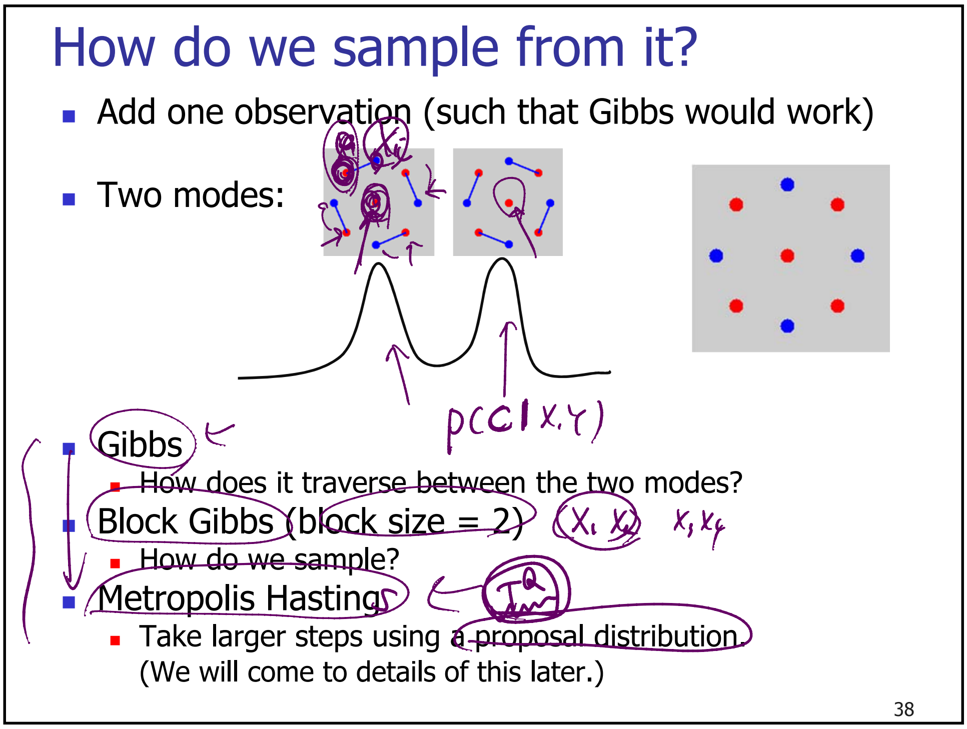
(A)



(B)



(C)

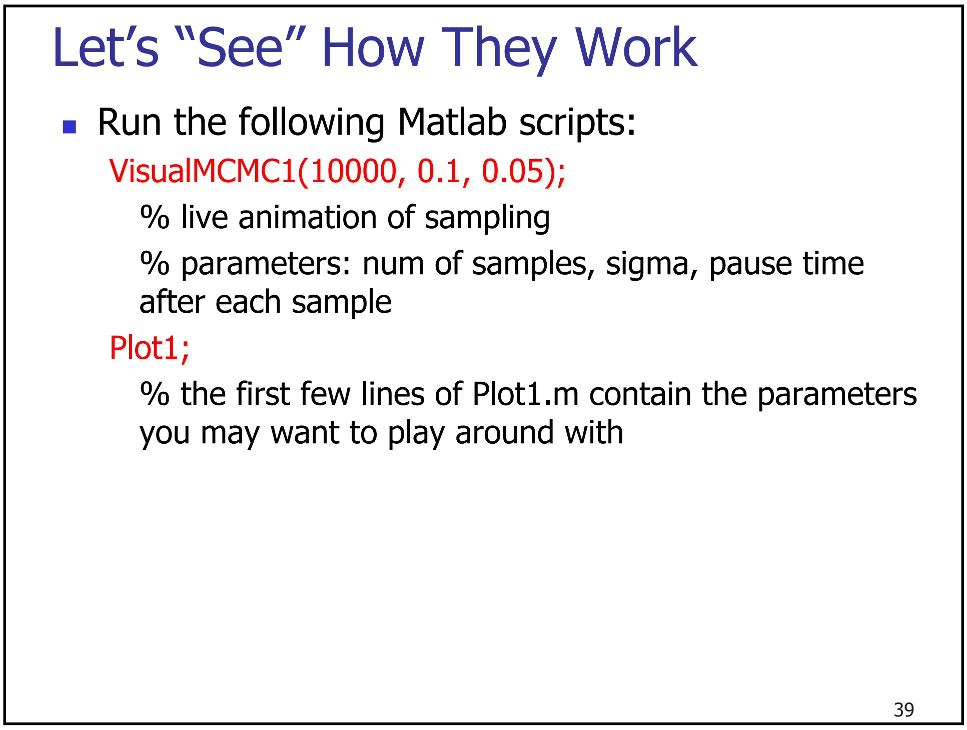


# How do we sample from it?

- Add one observation (such that Gibbs would work)
- Two modes:

- Gibbs
  - How does it traverse between the two modes?
- Block Gibbs (block size = 2) (X1 X2) X1 X4
  - How do we sample?
- Metropolis Hastings
  - Take larger steps using a proposal distribution (We will come to details of this later.)

38



# Let's "See" How They Work

- Run the following Matlab scripts:

  VisualMCMC1(10000, 0.1, 0.05);

  % live animation of sampling

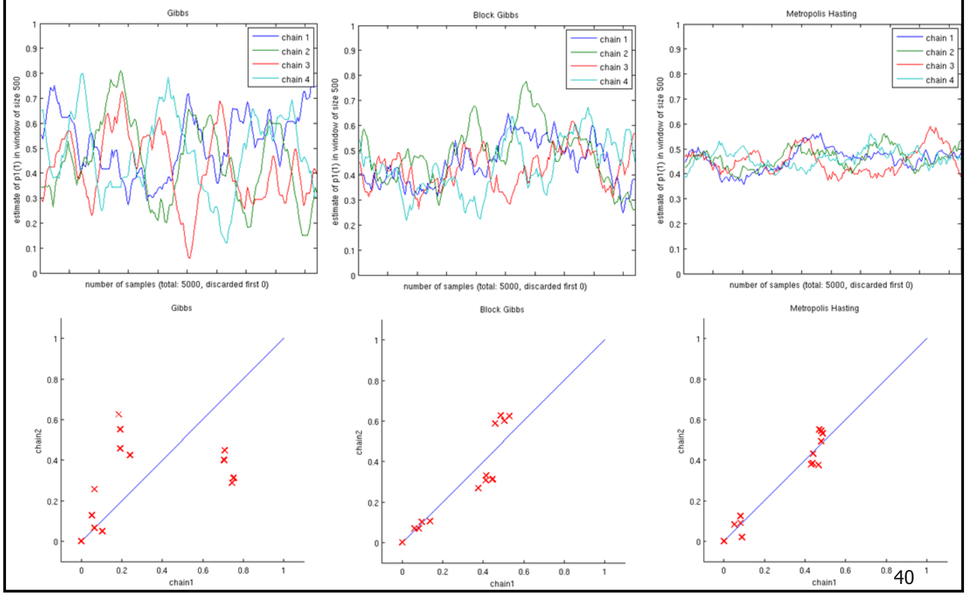
  % parameters: num of samples, sigma, pause time after each sample

  Plot1;

  % the first few lines of Plot1.m contain the parameters you may want to play around with

39

# Plots generated by "Plot1"



# Acknowledgement

- The MCMC sampler shown in class was written by Huayan Wang (Stanford, CS).
- These lecture slides were generated based on the slides from Prof Eran Segal and Huanyan Wang.