18799: Advanced Machine Learning

Ole J. Mengshoel
ole.mengshoel@sv.cmu.edu

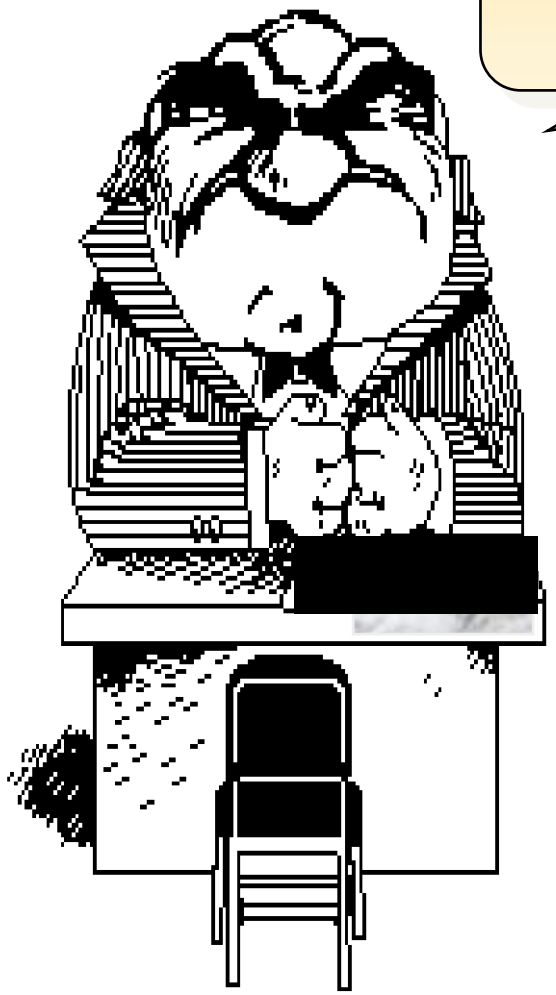
Department of ECE
Carnegie Mellon University, Silicon Valley

February 10th, Spring 2015

Today's Agenda

- Homeworks
  - HW1 is being graded
  - HW2 will be handed out later this month
- Guest lecture
  - No Guest Lecture by Brano Kveton on Tue Feb 17th , regular class or other Guest Lecture instead
- Paper presentations by students
  - Starting on Thu Feb 12th (this Thursday)
  - 5 first papers, their presenters and reviewers should be well-known
- Machine learning techniques
  - Particle-based (aka "sampling"-based) approximations
  - Markov chain Monte Carlo (MCMC)

Questions?



Particle-Based Approximate Inference

Overview (1)

- Motivation for Particle-Based Approximate Inference (aka Sampling) and MCMC
- Sampling methods
  - Forward Sampling
  - Likelihood Weighting and Importance Sampling
  - Collapsed Particles
- Markov chain Monte Carlo (MCMC)
  - Sampling processes based on Markov chains (MCMC)
  - Notable MCMC Algorithms
  - MCMC for Optimization and EM
  - MCMC – Recent Developments
- Applications
- Note: Some of these methods (Forward Sampling and Likelihood Weighting) only apply to Bayesian networks in their simple form

Overview (2)

- Koller & Friedman:
  - Chapter 12
  - Chapter 19, including 19.2 and 19.3
- “An Introduction to MCMC for Machine Learning” by Andrieu, De Freitas, Doucet, Jordan (2003)
  - Thanks to Ritchie Lee

Background

- Particle-based methods are approximate inference methods for P(X | $\boldsymbol{e}$)
- Approximate joint distribution by particles (set of instantiations or configurations) of all or some network nodes
  - Particles need to be designed to give good approximations
- Two axes of particle-based inference:
  - How are particles generated? Main answers: deterministically or probabilistically
  - What is a particle? Main answers: full particles (all nodes assigned a state) or collapsed particles (a strict subset of nodes assigned a state)
- Different algorithms: Forward Sampling, Likelihood Weighting, Gibbs sampling, …

Motivation – Why Approximation?

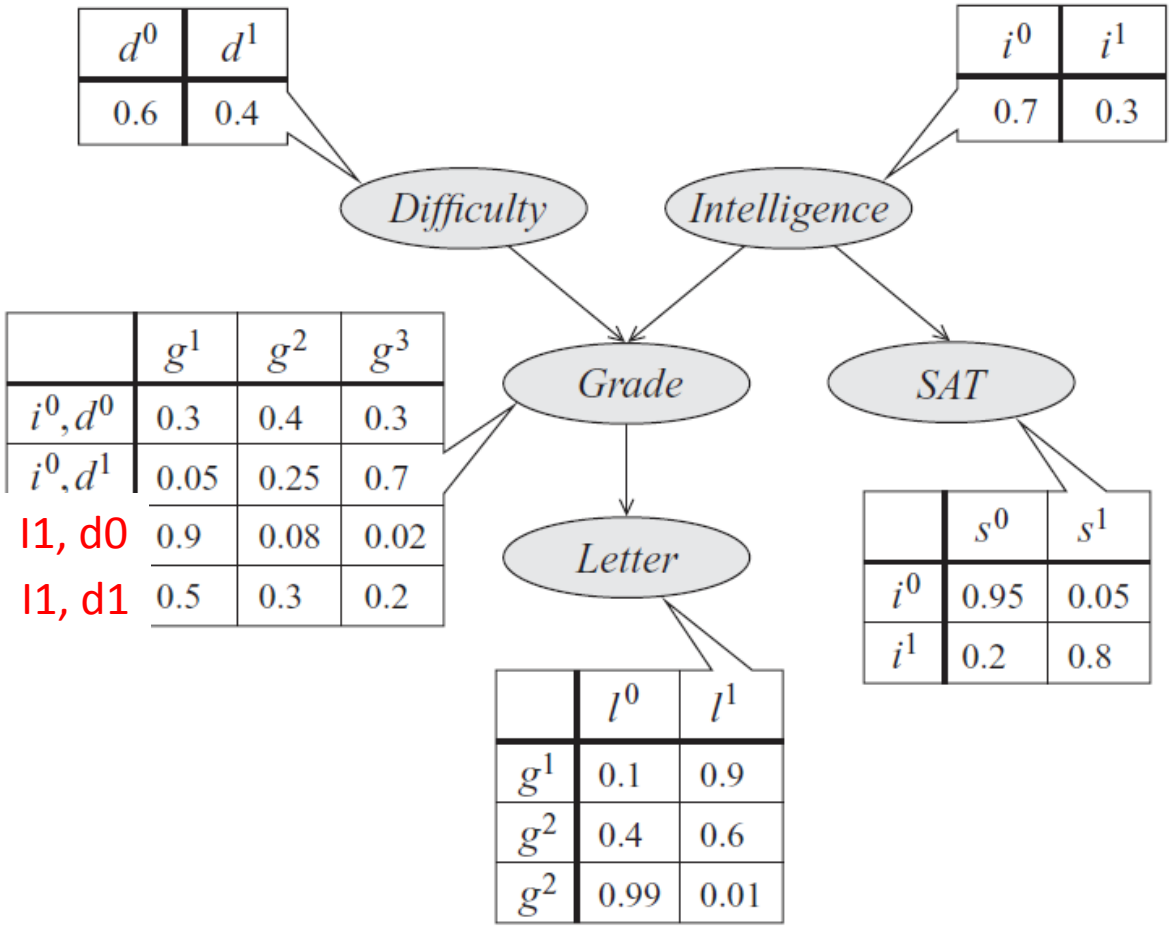
- Many computational problems associated with PGMs are NP-complete (or worse) or have no closed form
  - Both inference and learning problems
  - There are slow computations (measured in wall-clock time) in problem instances that are representative of real-world problems
- Particle-based approximate inference is a potential work-around
  - Gives up exact answers for, sometimes, better computational properties
    - Faster computation, lower memory usage, …
    - But: Particle-based approximate inference methods are “brute-force” or “weak” method - only use when “all else fails”

Framework

- Given a distribution $P(\boldsymbol{X})$ over random variables $\boldsymbol{X}$
- Want to estimate probability P of an event $\boldsymbol{Y} = \boldsymbol{y}$ for some $\boldsymbol{Y} \subseteq \boldsymbol{X}$ and $\boldsymbol{y} \in \text{Val}(\boldsymbol{Y})$
- Generally, estimate expectation of function $f(\boldsymbol{X})$ relative to P
  - Let $\xi\langle\boldsymbol{Y}\rangle$ be assignment in ξ to variables $\boldsymbol{Y}$
  - To estimate $P(\boldsymbol{Y})$, use $f(\xi) = \boldsymbol{1}\{\xi\langle\boldsymbol{Y}\rangle = \boldsymbol{y}\}$
- To approximate expectation, we generate M particles, estimate for each generated particle, and aggregate the results

Sampling Algorithms

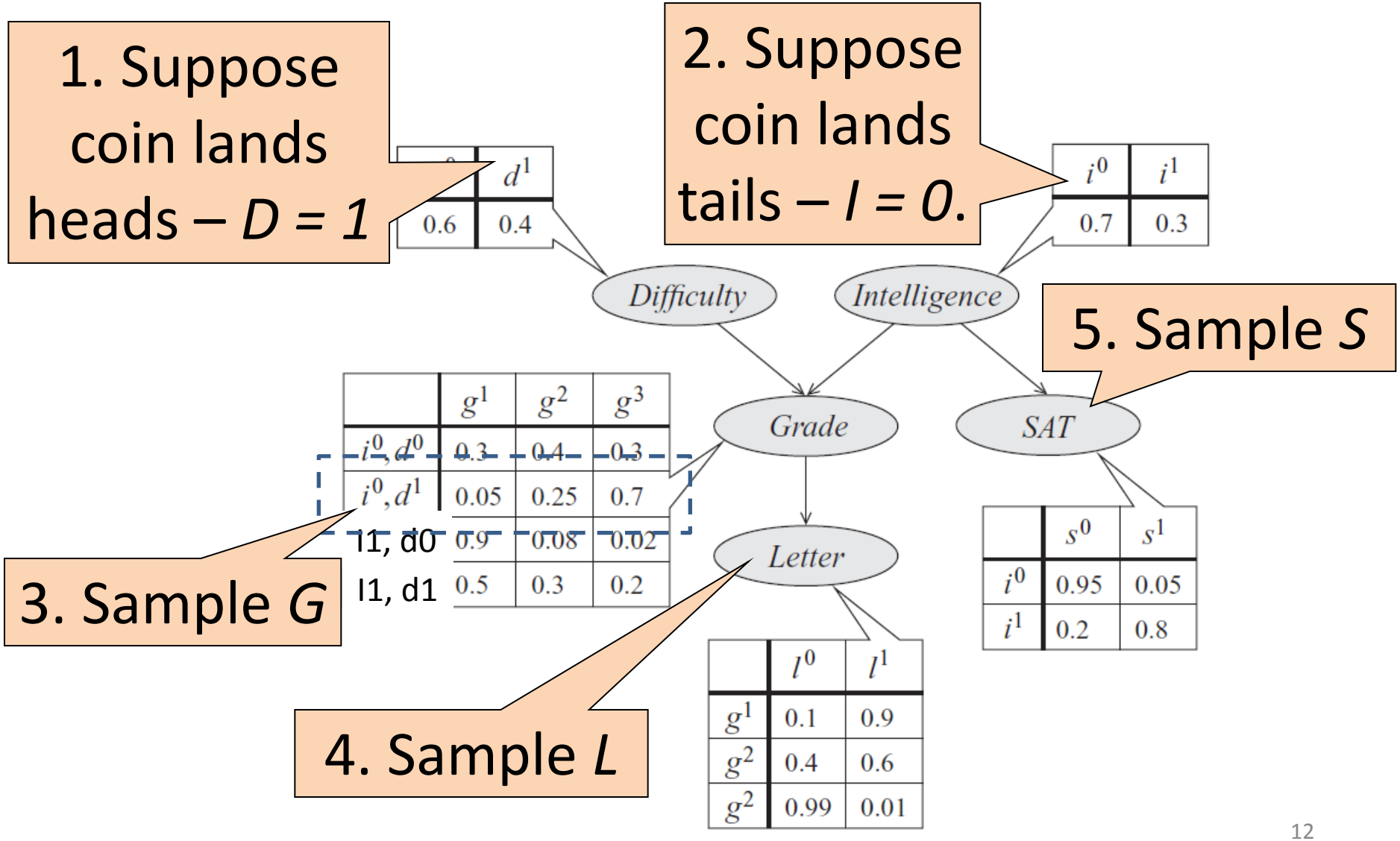
Forward Sampling Example (1)



Forward Sampling Example (2)

1. Suppose coin lands heads – *D = 1*

2. Suppose coin lands tails – *I = 0*.



	g^1	g^2	g^3
i^0, d^0	0.3	0.4	0.3
i^0, d^1	0.05	0.25	0.7
i^1, d^0	0.9	0.08	0.02
i^1, d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^2	0.99	0.01

Forward Sampling (FS) Algorithm

- **Input**:
  - $\boldsymbol{B}$: Bayesian network over nodes $\boldsymbol{X}$
- **let** $X_1, \ldots, X_n$ be topological ordering of $\boldsymbol{X}$
- **for** $i = 1, \ldots, n$
  - $\boldsymbol{u}_i \leftarrow \boldsymbol{x}\langle \mathrm{Pa}(X_i) \rangle$
  - Sample x_i from $\mathrm{Pa}(X_i \mid \boldsymbol{u}_i)$
- **Return** $(x_1, \ldots, x_n)$

Convergence Analysis

Given sample $D = \{\xi[1], \ldots, \xi[M]\}$, we can estimate expectation of any function f:

$$\boldsymbol{E}'_D(f) = \frac{1}{M}\sum_{m=1}^{M} f(\xi[m])$$

Since we want to compute $P(\mathrm{y})$, we get:

$$P'_D(\boldsymbol{y}) = \frac{1}{M}\sum_{m=1}^{M} 1(\boldsymbol{y}[m] = \boldsymbol{y})$$

where $\boldsymbol{y}[\mathrm{m}]$ denotes $\xi[\mathrm{m}]\langle \boldsymbol{Y} \rangle$: assignment to $\boldsymbol{Y}$ in particle $\xi[\mathrm{m}]$

Error Analysis – Hoeffding

The Hoeffding bound:

$$P_{\mathcal{D}}(\hat{P}_{\mathcal{D}}(\boldsymbol{y}) \notin [P_{\mathcal{D}}(\boldsymbol{y}) - \epsilon, P_{\mathcal{D}}(\boldsymbol{y}) + \epsilon] \leq 2e^{-2M\epsilon^2}$$

estimates how many samples M are needed to provide an error bounded by ϵ, with probability $\geq 1 - \delta$. Putting

$$2e^{-2M\epsilon^2} \leq \delta$$

and solving for number of samples M gives:

$$M \geq \frac{\ln(2/\delta)}{2\epsilon^2}.$$

In words: To provide absolute error ϵ with probability at least $1 - \delta$, we need number of samples M to grow (i) logarithmically in $1/\delta$ and (ii) quadratically in $1/\epsilon$.

Conditional Probability Queries

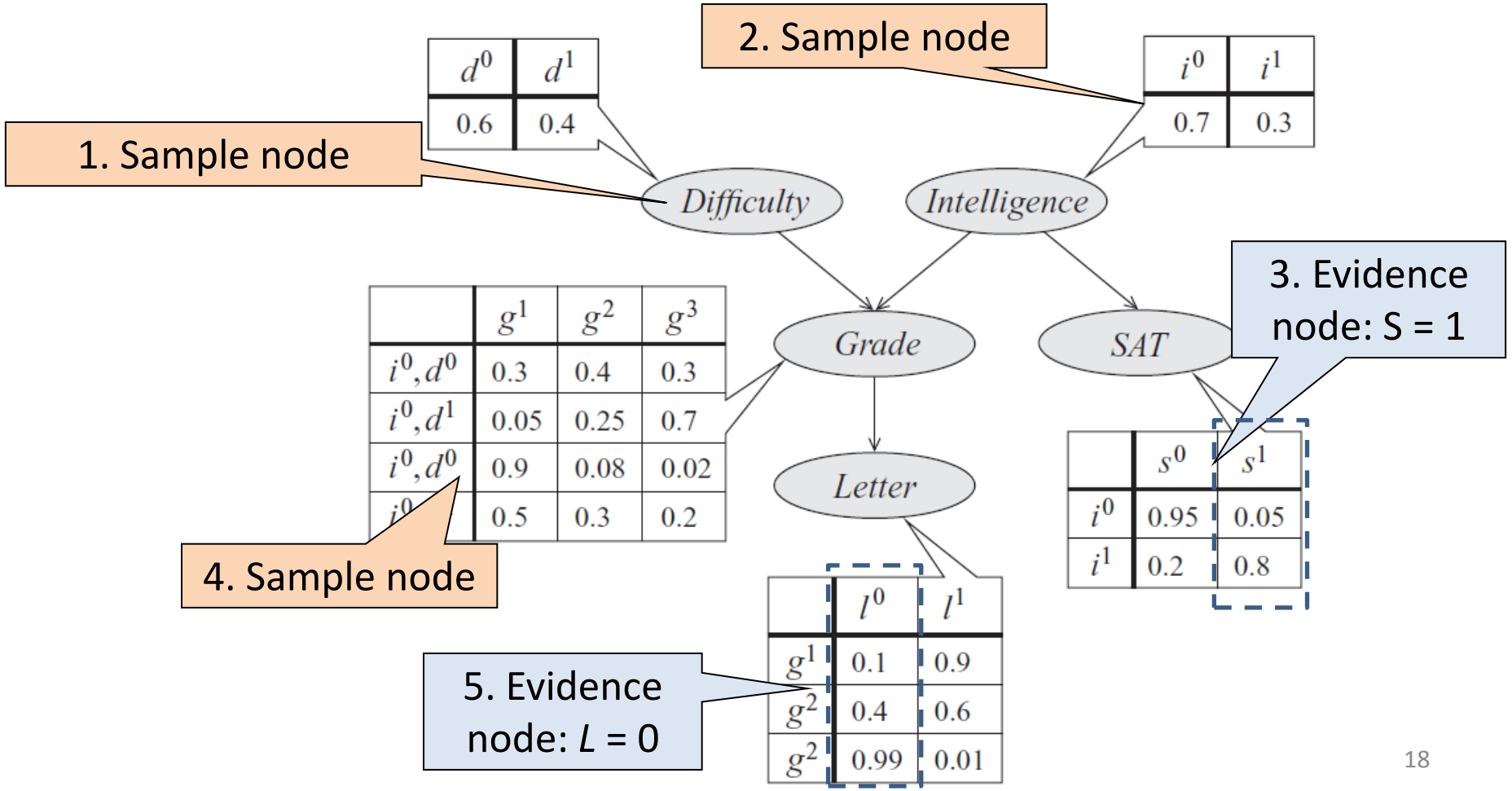
- So far: $P(\boldsymbol{Y} = \boldsymbol{y}) = P(\boldsymbol{y})$
- Now: $P(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{E} = \boldsymbol{e}) = P(\boldsymbol{y} \mid \boldsymbol{e})$
- Computing $P(\boldsymbol{y} \mid \boldsymbol{e})$ by Rejection (aka Logic) Sampling:
  - Generate samples $\boldsymbol{x}$ from $P(\boldsymbol{X})$
  - Reject sample inconsistent with $\boldsymbol{e}$
- FS analysis applies to Rejection Sampling also
  - *But:* The number of accepted particles may be very small (if evidence $\boldsymbol{e}$ is unlikely)
  - *Example:* If $P(\boldsymbol{e}) = 0.001$, even if M = 10,000, expected no of accepted particles is 10 (and low $P(\boldsymbol{e})$ is common)
  - *Generally:* To obtain $\geq$ M* accepted particles, need to generate $M = M^* / P(\boldsymbol{e})$ particles on average

Sampling Algorithms: Beyond Forward Sampling

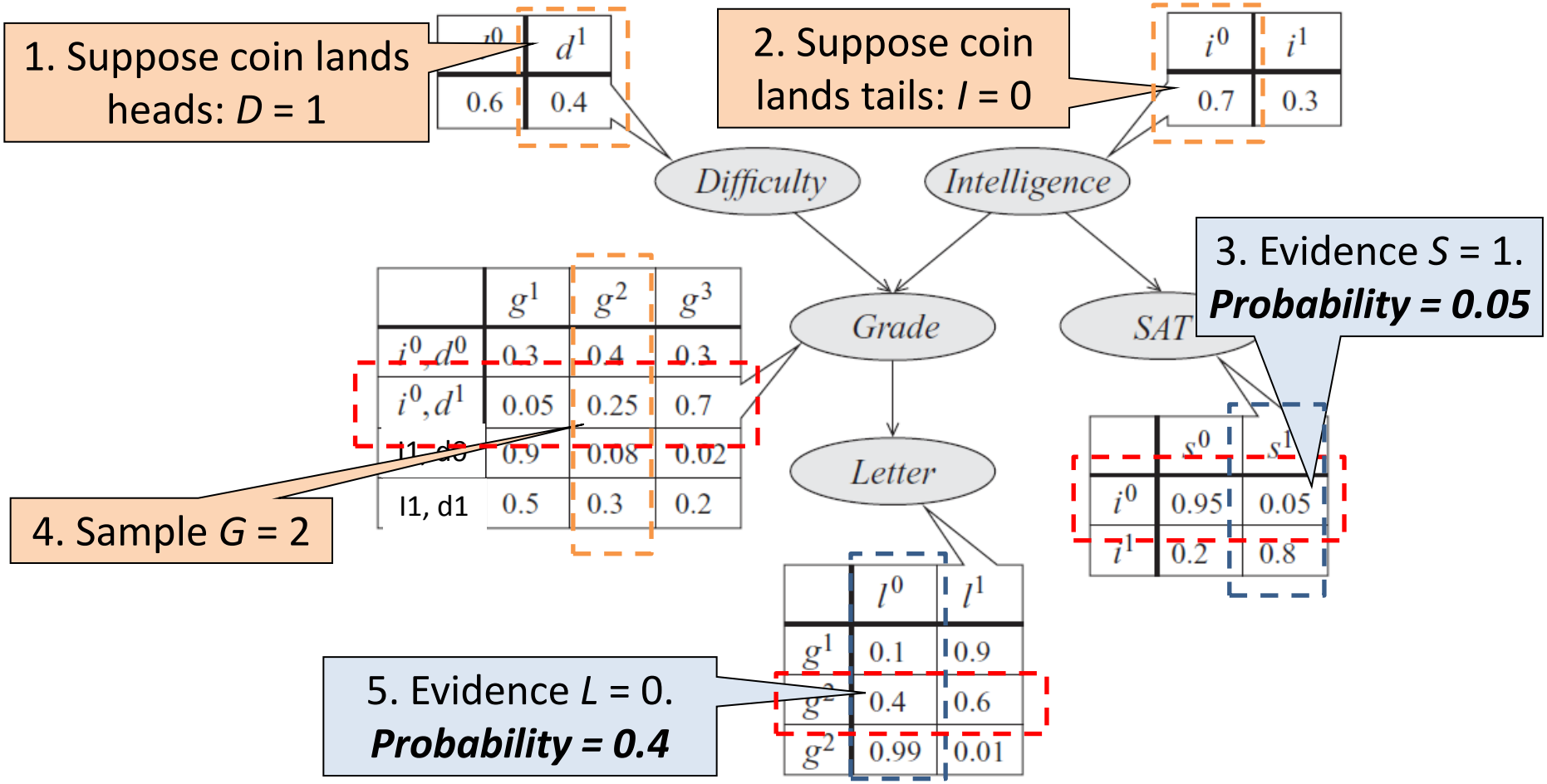
- Rejection Sampling
  - We looked at this already
  - Unfortunately, low $P(\boldsymbol{e})$ is common, exceptions are:
    - Forecasting – no observations in the future (yet)
    - Create ML data from Bayesian network
- Likelihood Weighting Sampling
  - Force samples to take $\boldsymbol{e}$ as their value
- Importance Sampling
  - Sample from proposal (or sampling) distribution

Likelihood Weighting Example (1)

We distinguish between non-evidence nodes and evidence nodes and handle them differently during the sampling process.



Likelihood Weighting Example (2)



Weight computed by Likelihood Weighting sampling process to compensate for setting of evidence: $w = 0.05 \times 0.4 = 0.02$.

Likelihood Weighting Sampling

- Inputs:
  - $\boldsymbol{B}$: Bayesian network over nodes $\boldsymbol{X}$
  - $\boldsymbol{Z} = \boldsymbol{z}$: event
- **let** $X_1, \ldots, X_n$ be topological ordering of $\boldsymbol{X}$
- $w \leftarrow 1$
- **for** $i = 1, \ldots, n$
  - $\boldsymbol{u}_i \leftarrow \boldsymbol{x}\langle \mathrm{Pa}(X_i) \rangle$
  - **if** $X_i \notin \boldsymbol{Z}$ **then** Sample x_i from $\mathrm{Pa}(X_i | \boldsymbol{u}_i)$
  - **else**
    - $x_i \leftarrow \boldsymbol{z}\langle X_i \rangle$
    - $w \leftarrow w \times P(x_i \mid \boldsymbol{u}_i)$
- **Return** $(x_1, \ldots, x_n)$, w

Convergence Analysis

Likelihood Weighting generates $\langle \xi[i], w[i] \rangle$, a *weighted particle*. We generate M such particles:

$$\mathcal{D} = \{\langle \xi[1], w[1] \rangle, ..., \langle \xi[M], w[M] \rangle\}.$$

We get the following estimate

$$\hat{P}_{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{e}) = \frac{\sum_{m=1}^{M} w[m] \mathbf{1}\{\boldsymbol{y}[m] = \boldsymbol{y}\}}{\sum_{m=1}^{M} w[m]}.$$

Clearly, this is a generalization of the estimator for the unweighted particles in Forward Sampling.

Importance Sampling

- Given a distribution $P(\boldsymbol{X})$ over random variables $\boldsymbol{X}$
- Want to estimate probability P of an event $\boldsymbol{Y} = \boldsymbol{y}$ for some $\boldsymbol{Y} \subseteq \boldsymbol{X}$ and $\boldsymbol{y} \in \text{Val}(\boldsymbol{Y})$
- Generally, estimate expectation of function $f(\boldsymbol{X})$ relative to P (target distribution)
- To approximate expectation, we generate M particles, estimate for each generated particle, and aggregate the results
- We might prefer to generates samples from a different distribution Q, the proposal distribution
  - It might be impossible or very hard to sample from P

Markov Chain Monte Carlo

Why Markov Chain Monte Carlo?

- So far:
  - Likelihood Weighting: Evidence only impacts the descendants of evidence nodes, in the weights
  - If much evidence at BN leaves, we are sampling from the prior, potentially far away from posterior
- Now:
  - A sampling approach, MCMC, which generates a sequence of samples
  - Even if the first sample is from the prior, later samples get closer and closer to the posterior
  - Generally, MCMC methods apply both to directed and undirected PGMs

Markov Chain Monte Carlo

- Monte Carlo in general
- Markov chain Monte Carlo algorithms
  - Gibbs Sampling
  - Block Gibbs Sampling
  - Metropolis-Hastings Algorithm

Monte Carlo Principle

Draw iid set of samples $\{x^{(i)}\}$ $i=1,..,N$ from target density $p(x)$:

- Approximate target density:

$$p_N(x) = \frac{1}{N}\sum_{i=1}^{N}\delta_{x^{(i)}}(x),$$

- Approximate an integral:

$$I_N(f) = \frac{1}{N}\sum_{i=1}^{N} f\left(x^{(i)}\right) \xrightarrow[N\to\infty]{a.s.} I(f) = \int_{\mathcal{X}} f(x)p(x)\,dx.$$

- Estimate x that maximizes p:

$$\hat{x} = \underset{x^{(i)};i=1,\ldots,N}{\arg\max}\; p\left(x^{(i)}\right)$$

Need for Sampling

Bayesian Inference and Learning

Normalization: $$p(x \mid y) = \frac{p(y \mid x)p(x)}{\int_{\mathcal{X}} p(y \mid x')p(x')\,dx'}.$$

Marginalization: $$p(x \mid y) = \int_{\mathcal{Z}} p(x, z \mid y)\,dz.$$

Expectation: $$\mathbb{E}_{p(x|y)}(f(x)) = \int_{\mathcal{X}} f(x)p(x \mid y)\,dx$$

Optimization, Estimation, Model selection

When exact solutions do not exist or is too computationally expensive.

Sampling (1)

Sampling from an arbitrary continuous distribution *p(x)*:

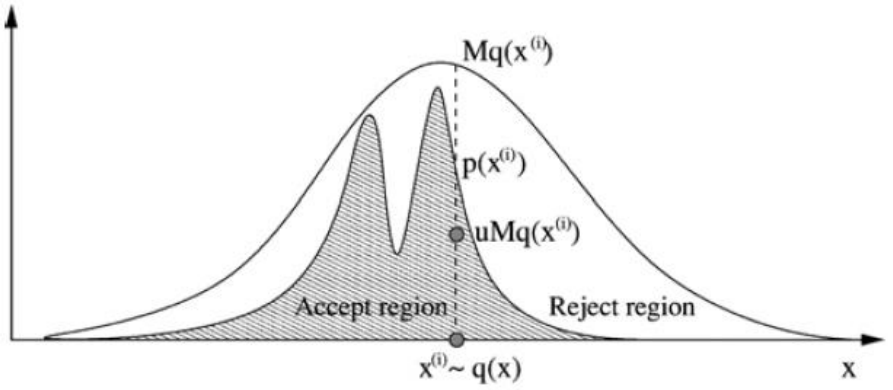
- Inverse CDF method
  - Draw *u* ~ *Uniform(0,1)*
  - Take sample as $x = F^{-1}(u)$
  - **But closed form of $F^{-1}(u)$ may not be accessible!**
- Rejection sampling
  - Sample from another easy-to-sample proposal distribution *q(x)* instead
  - *q(x)* must satisfy
    $p(x) \le Mq(x),\ M < \infty$
  - Requires 2 draws
  - **In most cases, very inefficient!**

Example: *p(x) = x, x ∈ [0,sqrt(2)]*

Set $i = 1$

Repeat until $i = N$

1. Sample $x^{(i)} \sim q(x)$ and $u \sim \mathcal{U}_{(0,1)}$.
2. If $u < \frac{p(x^{(i)})}{Mq(x^{(i)})}$ then accept $x^{(i)}$ and increment the counter i by 1. Otherwise, reject.



Sampling (2)

If you're only evaluating an expectation:

- Importance Sampling:
  - Take expectation w.r.t. proposal distribution *q(x)* instead.
  - Compensate using weights.

$$\mu = \int_{\mathcal{D}} f(\boldsymbol{x}) p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int_{\mathcal{D}} \frac{f(\boldsymbol{x}) p(\boldsymbol{x})}{q(\boldsymbol{x})} q(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \mathbb{E}_q \left(\frac{f(\boldsymbol{X}) p(\boldsymbol{X})}{q(\boldsymbol{X})} \right),$$

  - Very useful in simulating low probability events.
  - Hard to choose good *q(x)*
  - Does not work well in high dimensions

None of these methods worked for you, what now?

Number 1 of Top 10

"Great algorithms are the poetry of computation"

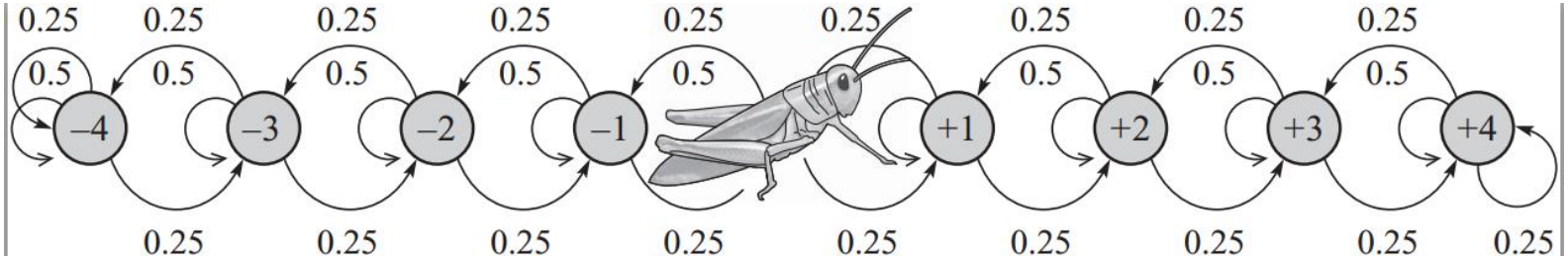
says Francis Sullivan of the Institute for Defense Analyses. He and other folks from the University of Tennessee and Oak Ridge National Laboratory have put together a list of 10 algorithms having "the greatest influence on the development and practice of science and engineering in the 20th century".

The list appears in the January/February issue of Computing in Science & Engineering.

1. **1946: The Metropolis Algorithm for Monte Carlo.** Through the use of random processes, this algorithm offers an efficient way to stumble toward answers to problems that are too complicated to solve exactly.
2. **1947: Simplex Method for Linear Programming.** An elegant solution to a common problem in planning and decision-making.
3. **1950: Krylov Subspace Iteration Method.** A technique for rapidly solving the linear equations that abound in scientific computation.
4. **1951: The Decompositional Approach to Matrix Computations.** A suite of techniques for numerical linear algebra.
5. **1957: The Fortran Optimizing Compiler.** Turns high-level code into efficient computer-readable code.
6. **1959: QR Algorithm for Computing Eigenvalues.** Another crucial matrix operation made swift and practical.
7. **1962: Quicksort Algorithms for Sorting.** For the efficient handling of large databases.
8. **1965: Fast Fourier Transform.** Perhaps the most ubiquitous algorithm in use today, it breaks down waveforms (like sound) into periodic components.
9. **1977: Integer Relation Detection.** A fast method for spotting simple equations satisfied by collections of seemingly unrelated numbers.
10. **1987: Fast Multipole Method.** A breakthrough in dealing with the complexity of n-body calculations, applied in problems ranging from celestial mechanics to protein folding.

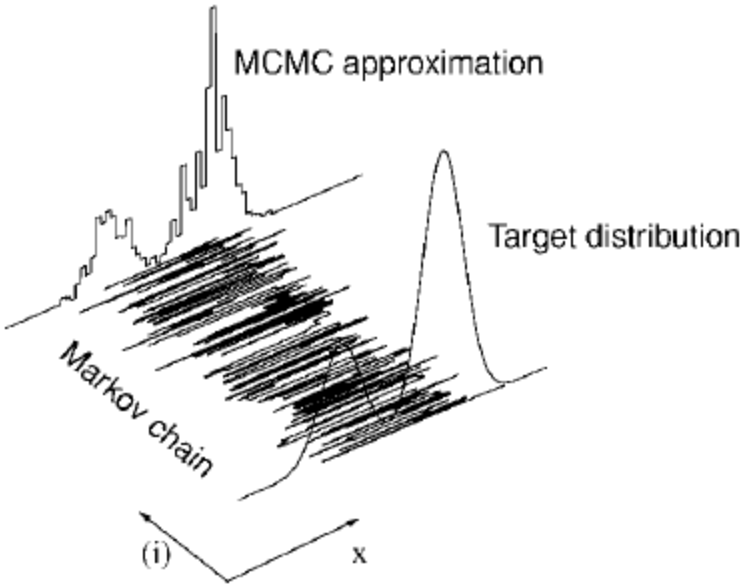
From *Random Samples*, Science page 799, February 4, 2000.

Markov Chain Monte Carlo



Main Ideas:

- Use Monte Carlo methods
- Construct a Markov chain whose stationary distribution is the desired target distribution *p(x)*
- But how?



Overview: Metropolis-Hastings (M-H)

Stationary Distributions:

- Sufficient conditions
  - Irreducibility (no disjoint pieces)
  - Aperiodicity (doesn't get trapped in cycles)

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

     $x^{(i+1)} = x^*$

     else

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

- Detailed balance equation

$$p(x^{(i)})T(x^{(i-1)} \mid x^{(i)}) = p(x^{(i-1)})T(x^{(i)} \mid x^{(i-1)}).$$

See PGM textbook Section 12.3.4.2

Global Optimization

Maximize *p(x):*

Using Naïve MCMC

- Walk around randomly in Markov Chain
- Compute

$$\hat{x} = \underset{x^{(i)};i=1,\ldots,N}{\arg\max}\ p\left(x^{(i)}\right).$$

- **Many wasted samples**

Using Simulated Annealing

- Global optimization, stochastic
- Adds a cooling schedule to M-H.
- Ratio in probabilities gets increasingly emphasized
- Concentration on global maxima in asymptotic behavior:

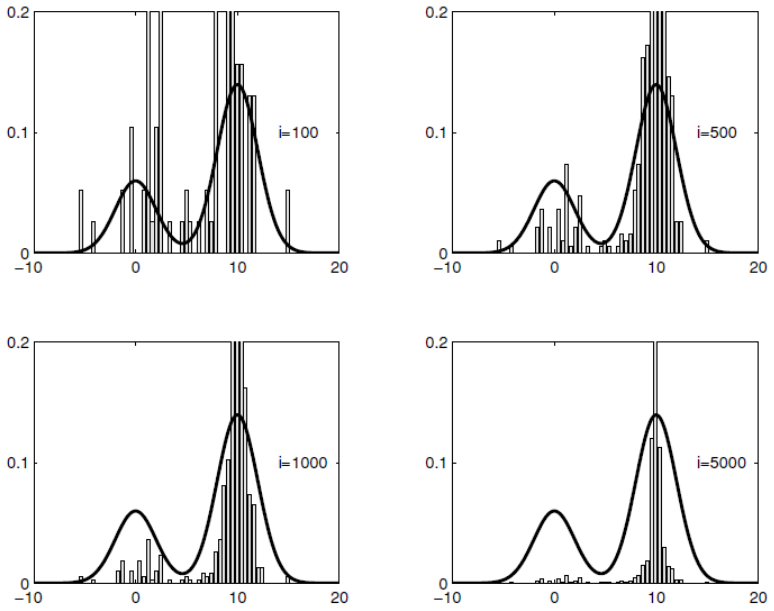
$$\lim_{i\to\infty} T_i = 0.$$

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.



Monte Carlo EM

Expectation Maximization

- Parameter estimation with some unobserved variables
- E-step: Compute the expected value of the log-likelihood function w.r.t the distribution of the hidden variables

$$Q(\theta) = \int_{\mathcal{X}_h} \log(p(x_h, x_v \mid \theta)) p\left(x_h \mid x_v, \theta^{(\text{old})}\right) dx_h,$$

- M-step: Treat the distribution of the hidden variables as observed and perform maximum likelihood estimation of the parameters.

$$\theta^{(\text{new})} = \arg\max_\theta Q(\theta).$$

- In general, this integral may be **intractable** and so we can approximate it using MCMC.

Bayesian network MLE version of this algorithm: See K&F textbook Section 19.2 "Parameter Estimation," in particular Section 19.2.2.3.

See also section K&F textbook Section 19.3 – "Bayesian Learning with Incomplete Data," in particular Section 19.3.2.

Overview: Special Cases of M-H

Metropolis Algorithm

- Assumes symmetric proposal distribution:

$$q(x^\star \mid x^{(i)}) = q(x^{(i)} \mid x^\star)$$

- Simplification of acceptance criteria:

$$\mathcal{A}(x^{(i)}, x^\star) = \min\left\{1, \frac{p(x^\star)}{p(x^{(i)})}\right\}.$$

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

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

     else

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

Gibbs Sampling

- Uses mixtures and cycles property
- Especially suitable for Bayesian networks
- Sample variables in turn given all other variables
- Use posterior as the proposal distribution

Algorithm 12.4 Generating a Gibbs chain trajectory

Procedure Gibbs-Sample (
  $\boldsymbol{X}$ // Set of variables to be sampled
  Φ // Set of factors defining P_Φ
  $P^{(0)}(\boldsymbol{X})$, // Initial state distribution
  T // Number of time steps
)

1 Sample $\boldsymbol{x}^{(0)}$ from $P^{(0)}(\boldsymbol{X})$
2 **for** $t = 1, \ldots, T$
3 $\boldsymbol{x}^{(t)} \leftarrow \boldsymbol{x}^{(t-1)}$
4 **for** each $X_i \in \boldsymbol{X}$
5 Sample $x_i^{(t)}$ from $P_\Phi(X_i \mid \boldsymbol{x}_{-i})$
6 // Change X_i in $\boldsymbol{x}^{(t)}$
7 **return** $\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(T)}$

See PGM textbook Section 12.3.1

Gibbs Sampling: Example (1)

Let evidence be $\boldsymbol{e} = \{S = \mathbf{1}, L = \mathbf{0}\} = \{s^1, l^0\}$. We sample over D, I, and G. Set of reduced factors is: $P(I)$, $P(D)$, $P(G \mid I, D)$, $P(s^1 \mid I)$, and $P(l^0 \mid G)$. Suppose FS gives, in 0th iteration, sample:

$$d^{(0)} = d^1, i^{(0)} = i^0, g^{(0)} = g^2.$$

In 1st iteration, for order G, I, D, sample $g^{(1)}$ from

$$P_{\Phi}(G \mid i^0, d^1) = \frac{P(G \mid i^0, d^1)P(l^0 \mid G)}{\sum_g P(g \mid i^0, d^1)P(l^0 \mid g)},$$

giving g^3. We then sample $i^{(1)}$ from $P_{\Phi}(I \mid d^1, g^3)$, giving i^1 and then sample $d^{(1)}$ from $P_{\Phi}(D \mid g^3, i^1)$, giving d^1. The 1st iteration sample is thus

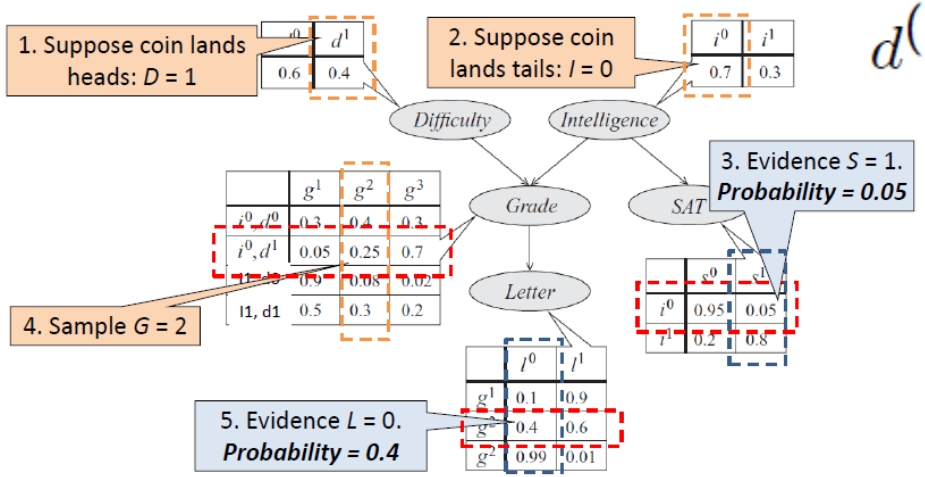
$$d^{(1)} = d^1, i^{(1)} = i^1, g^{(1)} = g^2.$$

The GS sampling process now repeats.

Gibbs Sampling: Example (2)

Evidence: $\boldsymbol{e} = \{S = 1, L = 0\} = \{s^1, l^0\}$

First (initialization) step :



$$d^{(0)} = d^1, i^{(0)} = i^0, g^{(0)} = g^2$$

Second and later step: Perform Gibbs sampling over *G*, *I*, and *D* (iteratively), starting from the above:

$$P_{\Phi}(G \mid i^0, d^1) = \frac{P(G \mid i^0, d^1) P(l^0 \mid G)}{\sum_g P(g \mid i^0, d^1) P(l^0 \mid g)}$$

Gibbs Sampling (GS)

- **Inputs**:
  - $\boldsymbol{X}$: variables (BN nodes) to be sampled
  - Φ: factors defining P_Φ
  - $P^{(0)}(\boldsymbol{X})$: initial state distribution
  - T: number of time steps
- Sample $\boldsymbol{x}^{(0)}$ from $P^{(0)}(\boldsymbol{X})$
- **for** $t = 1, \ldots, T$
  - $\boldsymbol{x}^{(t)} \leftarrow \boldsymbol{x}^{(t-1)}$
  - **for** each $X_i \in \boldsymbol{X}$
    - Sample $\boldsymbol{x}_i^{(t)}$ from $P_\Phi(X_i \mid \boldsymbol{X}_{-i})$
- **Return** $\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(T)}$

Factors are a generalization of CPTs in BNs.

Here, $\boldsymbol{X}_{-i}$ is all the BN nodes except X_i.

Stationary Markov Chains

- **Def**. A Markov chain is regular is there exists a k, such that for every pair of states *x* and *x'*, the probability of getting from *x* to *x'* in exactly k steps is non-zero.
- **Thm**. If a finite Markov chain is regular, then it has a unique stationary distribution.
- Gibbs sampling is a way to set up such a Markov chain

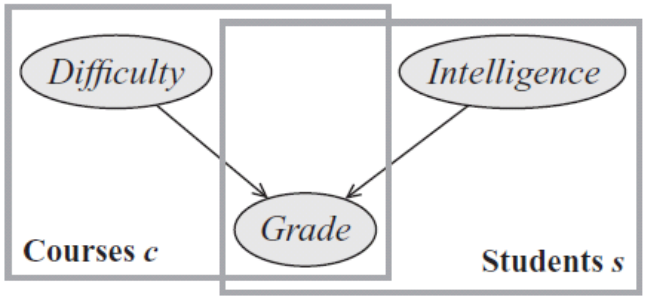
Gibbs Sampling: Summary

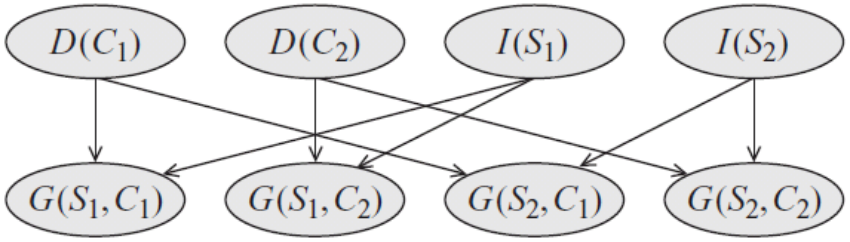
- Intuitive argument:
  - Unlike FS, GS takes downstream evidence into account
  - Thus, sampling distribution gets closer to posterior
  - As we repeat GS, distribution from which we sample gets closer and closer to $P_{\Phi}(\boldsymbol{X}) = P(\boldsymbol{X} \mid \boldsymbol{e})$
- Details:
  - Formalize the above intuition within Markov chain Monte Carlo (MCMC)
    - general approach with the goal of sampling from $P(\boldsymbol{X} \mid \boldsymbol{e})$
    - iterative process, gets closer and closer to $P(\boldsymbol{X} \mid \boldsymbol{e})$, using a Markov chain
  - Key question: How long to iterate before we start creating samples that are "close" to $P(\boldsymbol{X} \mid \boldsymbol{e})$
    - various ways to address this issue

Block Gibbs Sampling

- Standard Gibbs sampling: We sample a single variable Y, conditioned on the rest $\boldsymbol{X} - \{Y\}$
- Assume partitioning of $\boldsymbol{X}$ into blocks: $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$
- We now sample $\boldsymbol{x}_i$ from:
  - $P(\boldsymbol{X}_i \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_k)$
- Block Gibbs sampling: Iteratively sample blocks of variables rather than single variables
  - Generate new particle once all blocks resampled
  - Takes "longer-range" jumps in state space
  - Standard Gibbs is a special case of Block Gibbs (one variable per block)

Gibbs Sampling: Student Example (1)

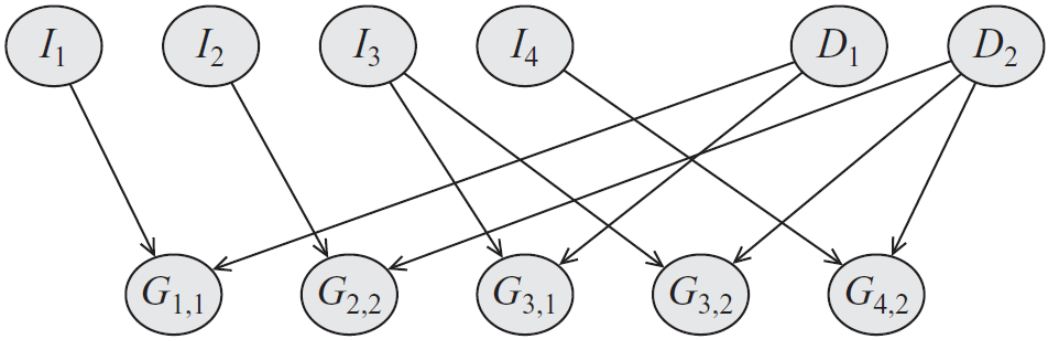




For any pair (student, course), the attribute Grade(s, c) depends on Intelligence(s) and Difficulty(c). CPDs are assumed outside of plates.

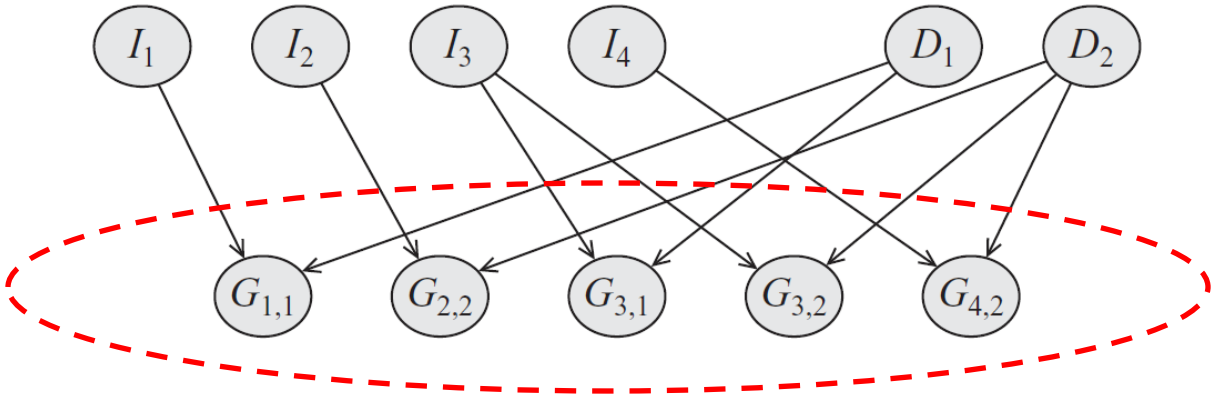
Ground Bayesian network, resulting from instantiating the plate with two students S_1 and S_2, each taking both courses C_1 and C_2.

Gibbs Sampling: Student Example (2)



- Students (Intelligence): $\boldsymbol{I} = \{I_1, \ldots, I_n\}$
- Courses (Difficulty): $\boldsymbol{D} = \{D_1, \ldots, D_m\}$
- Grades: $\boldsymbol{G} = \{G \mid G$ has parents in $\boldsymbol{D}$ and $\boldsymbol{I}$
- Above: $n = 4$ and $m = 2$

Gibbs Sampling: Student Example (3)



- We observe grades: ***G*** = ***g***
- Note:
  - ***I***-variables conditionally independent given ***D*** = ***d***
  - ***D***-variables conditionally independent given ***I*** = ***I***
- Thus we can:
  - Sample ***I***-variables as a block (independently of each other)
  - Sample ***D***-variables as a block (independently of each other)

Gibbs Sampling: XOR Example

- All nodes are binary
- *X* and *Y* are uniform, *Z* is XOR of *X* and *Y*
  - $Z = 1$ iff $X \neq Y$
- Posteriors for evidence $Z = 1$:
  - True posterior: $P(X=1, Y=0, Z = 1) = ½$, $P(X=0, Y=1, Z = 1) = ½$,
  - Gibbs Sampling:
    - Start in state $X=1, Y=0, Z = 1 \Rightarrow$ stuck in this state
    - Start in state $X=0, Y=1, Z = 1 \Rightarrow$ stuck in this state
  - Limitations of Gibbs Sampling

A Broader Class of Markov Chains

- Gibbs Sampling
  - Markov chain: (i) Regularity and (ii) target stationary distribution
  - Gibbs chain: Markov chain with properties (i) and (ii) guaranteed
- Limitations of Gibbs Sampling
  - Complicated to sample from $P(X_i \mid \boldsymbol{x}_{-i})$: continuous case
  - Only uses local moves
    - Changes one variable at a time
    - Creates basins of attraction in models with strongly correlated variables
- Now: Broader class of Markov chains
  - Metropolis-Hastings Sampling

Metropolis-Hastings Algorithm

- Allows us to construct a reversible Markov chain with a particular stationary distribution
  - Reversible Markov chain: obeys detailed balance equation
- It uses the idea of a proposal distribution
  - Sample from different distribution than target
  - Correct for error
- We randomly choose whether or not to accept the proposed transition
  - Unlike importance sampling, weights are not used
- Gibb Sampling is a special case of the Metropolis-Hastings Algorithm

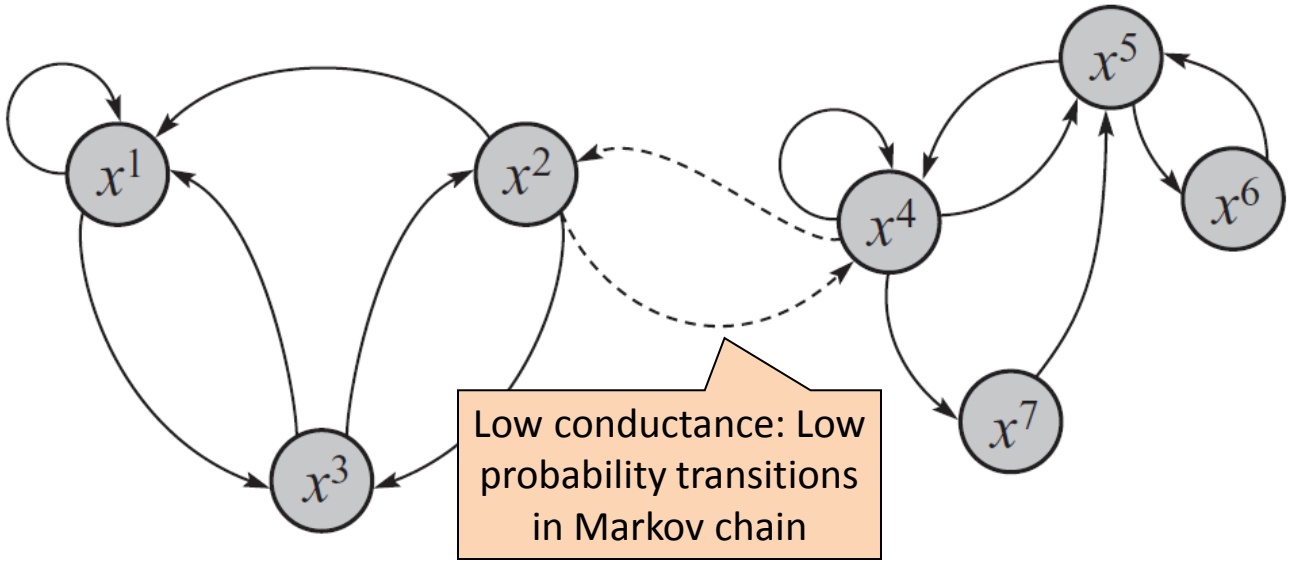
Markov Chain Usage

- So far: Defined Markov chain with desired stationary distribution
- Suppose we have such a Markov chain
  - With a unique stationary distribution
  - This is the one we want to sample from
- Q&A:
  - Question: How to use the chain to answer queries?
  - (Naïve) Answer: Do this:
    - Run MCMC algorithm until converge
    - Collect a sample
    - Repeat for each particle to be collected
    - Result: Data set D with independent particles, collected from close to stationary distribution
  - Is something wrong in the above (Naïve) Answer?

Using a Markov Chain

- MCMC algorithms set up Markov chains that can be used to estimate expectations or posteriors
- Mixing time: How long to wait before we can start sampling?
- Collecting samples: How to collect samples and perform estimation using the Markov chain?

Mixing Time and Low Conductance:



- Upper and lower bounds on mixing, in terms of conductance, exist
- PGMs with deterministic (or 0-1) or highly skewed parameters often have low conductance
- Heuristics are used to evaluate whether sampling has mixed

Collecting Samples

- Bad news: Burn-in time can be large for a large Markov chain
- Good news: If $\boldsymbol{x}(t)$ is sampled from π, then so is $\boldsymbol{x}(t+1)$
- Thus we have this method:
  - First: Burn-in phase: $\boldsymbol{x}(0), \ldots, : \boldsymbol{x}(T)$
  - Second: Collect samples $\boldsymbol{x}[m] = \boldsymbol{x}(T+m)$ for $m = 1, \ldots, M$. This gives samples $D = \{\boldsymbol{x}[1], \ldots, : \boldsymbol{x}[M]\}$
  - Note: If $\boldsymbol{x}(T+1)$ is sampled from π, then so is $\boldsymbol{x}(T+m)$ for $m > 1$.

Particle-Based Approximate Inference: Variants

Two Particle-Based Variants

- Collapsed particles
  - So far: Each particle instantiates each variable in the PGM
    - Unfortunately, there's an exponential number of these
  - Here: Each particle is a partial assignment of variables in the PGM
- Deterministic search
  - So far: Particles in PGM are generated by random sampling
    - Problematic for highly skewed distributions
  - Here: Deterministic search for distinct, high-probability assignments

Collapsed Particles

- Idea: A particle is a partial assignment to a subset of variables
- Partition nodes $\boldsymbol{X}$ into two subsets:
  - Variables defining the particle: $\boldsymbol{X}_p$
  - Variables for which we will use a closed-form distribution: $\boldsymbol{X}_d$
- Collapsed particle ($\boldsymbol{x}_{p,}$, P ($\boldsymbol{X}_d \mid \boldsymbol{x}_p, \boldsymbol{e}$)), where $\boldsymbol{x}_p \in \text{Val}(\boldsymbol{X}_p)$
  - "Collapsed" (aka "Rao-Blackwellized") because some variables are summarized by means of a distribution
  - Use samples $\boldsymbol{x}_p[m]$ to approximate expectations related to P ($\boldsymbol{X}_p \mid \boldsymbol{e}$), which are then used for estimating P ($\boldsymbol{X}_p, \boldsymbol{X}_d \mid \boldsymbol{e}$)

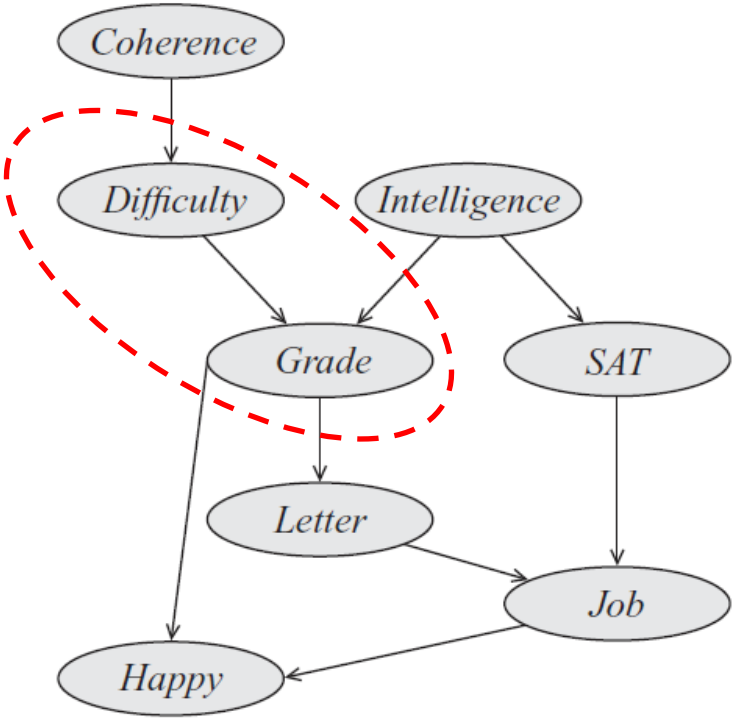
Collapsed Importance Sampling

- Generate samples from proposal distribution Q rather than from target distribution P
- Compensate for discrepancy between Q (proposal) and P (target) by putting a weight $w[m]$ on the m-th particle
  - Collapsed case: Particles only for $\boldsymbol{X}_{\text{p}}$
- Generate data set:
  - $\boldsymbol{D} = \{(\boldsymbol{X}_{\text{p}}[m], w[m], P(\boldsymbol{X}_d \mid \boldsymbol{x}_p[m], \boldsymbol{e})) \mid m \in M\}$
- Estimate expectation using $\boldsymbol{D}$ (see example below and book)

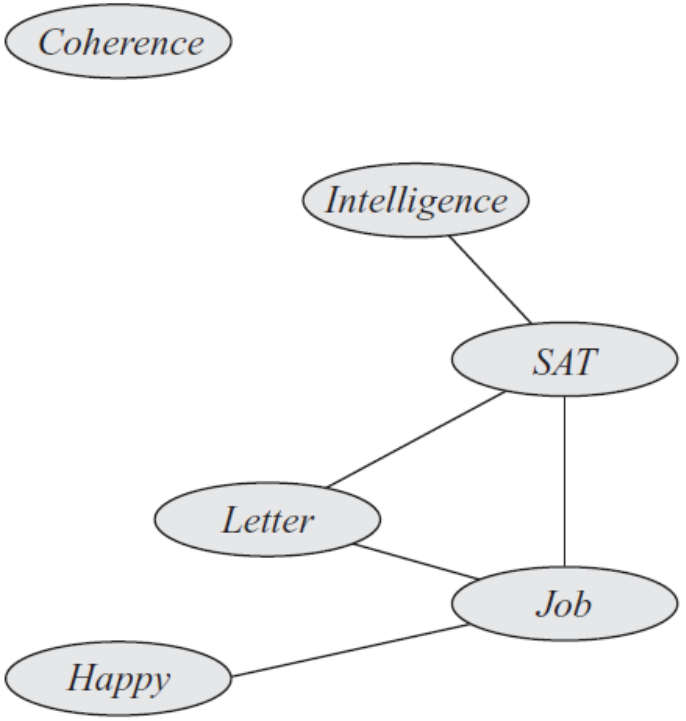
Collapsed Particles: Example (1)

Evidence: d^1, h^0

Partitioning of $\boldsymbol{X}$: $\boldsymbol{X}_p = \{D, G\}$ and $\boldsymbol{X}_d = \{C, I, S, L, J, H\}$



(a) ORIGINAL



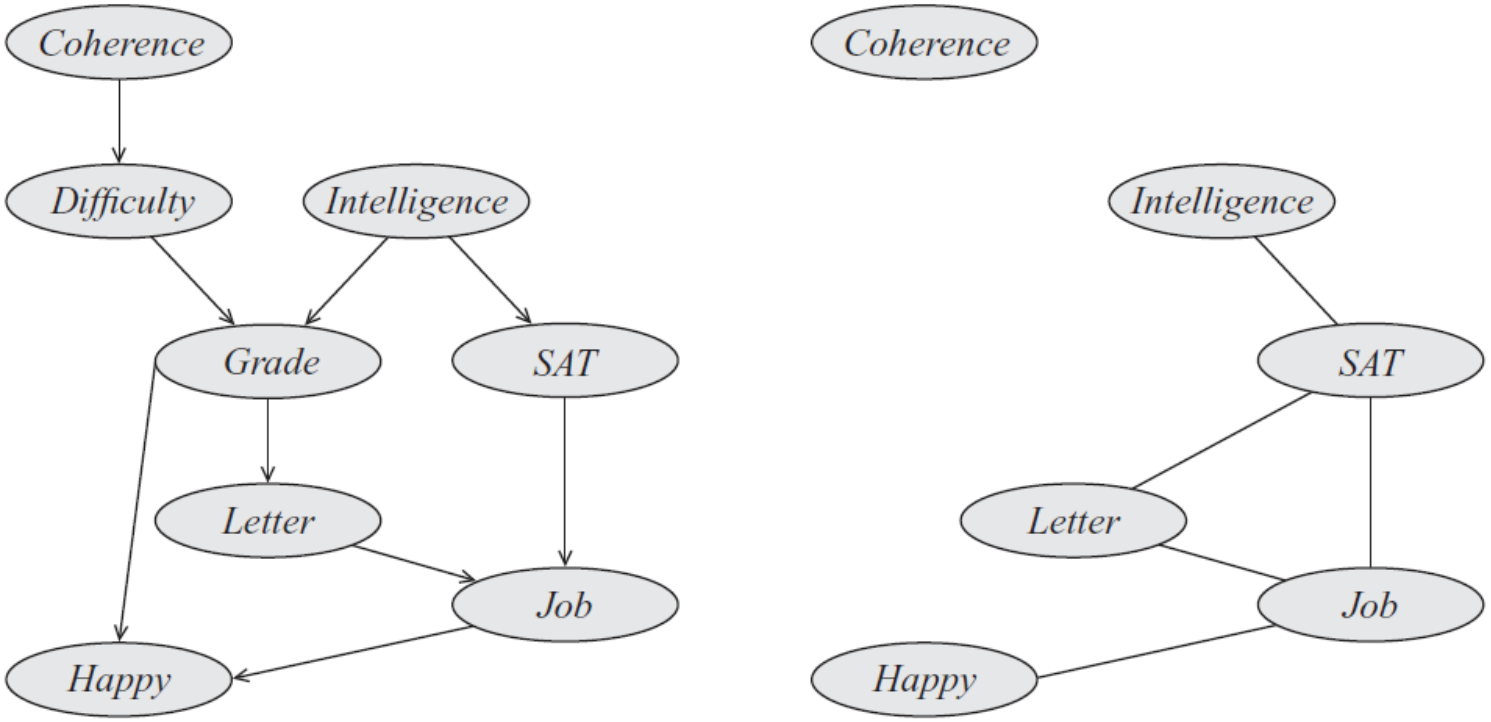
(b) REDUCED

Collapsed Particles: Example (2)

Evidence: d^1, h^0 and $\boldsymbol{X}_p = \{D, G\}$

Computation P($J \mid g, d^1, h^{0)}$): faster in reduced PGM

Compute average, using importance weights



(a) ORIGINAL (b) REDUCED

Applications

Applications – Overview (1)

1. *Computer vision*. Tracking (Isard & Blake, 1996; Ormoneit, Lemieux, & Fleet, 2001), stereo matching (Dellaert et al., this issue), colour constancy (Forsyth, 1999), restoration of old movies (Morris, Fitzgerald, & Kokaram, 1996) and segmentation (Clark & Quinn, 1999; Kam, 2000; Tu & Zhu, 2001).
2. *Web statistics*. Estimating coverage of search engines, proportions belonging to specific domains and the average size of web pages (Bar-Yossef et al., 2000).
3. *Speech and audio processing*. Signal enhancement (Godsill & Rayner, 1998; Vermaak et al., 1999).
4. *Probabilistic graphical models*. For example (Gilks, Thomas, & Spiegelhalter, 1994; Wilkinson & Yeung, 2002) and several papers in this issue.
5. *Regression and classification*. Neural networks and kernel machines (Andrieu, de Freitas, & Doucet, 2001a; Holmes & Mallick, 1998; Neal, 1996; Müller & Rios Insua, 1998), Gaussian processes (Barber & Williams, 1997), CART (Denison, Mallick, & Smith, 1998) and MARS (Holmes & Denison, this issue).
6. *Computer graphics*. Light transport (Veach & Guibas, 1997) and sampling plausible solutions to multi-body constraint problems (Chenney & Forsyth, 2000).

Applications – Overview (2)

7. *Data association*. Vehicle matching in highway systems (Pasula et al., 1999) and multitarget tracking (Bergman, 1999).
8. *Decision theory*. Partially observable Markov decision Processes (POMDPs) (Thrun, 2000; Salmond & Gordon, 2001), abstract Markov policies (Bui, Venkatesh, & West, 1999) and influence diagrams (Bielza, Müller, & Rios Insua, 1999).
9. *First order probabilistic logic*. (Pasula & Russell, 2001).
10. *Genetics and molecular biology*. DNA microarray data (West et al., 2001), cancer gene mapping (Newton & Lee, 2000), protein alignment (Neuwald et al., 1997) and linkage analysis (Jensen, Kong, & Kjærulff, 1995).
11. *Robotics*. Robot localisation and map building (Fox et al., 2001).
12. *Classical mixture models*. Mixtures of independent factor analysers (Utsugi, 2001) and mixtures of factor analysers (Fokoué & Titterington, this issue).

Sampling and MCMC are so fundamental that applications are everywhere.

Correspondence and Data Association: A Class of Applications

- This pattern is often found:
  - Map between one set of objects $U = \{u_1, \ldots, u_n\}$ and another set $V = \{v_1, \ldots, v_m\}$
- For example the problem of *data association*:
  - Map airplanes (objects V) to sensor measurements, blips on a radar screen (objects U): non-cooperative identification (NCID)
  - Map robot's obstacles (objects V) to sensor measurements from its laser range finder (objects U)
- Probabilistic graphical models can often be learned for these problems, and used for inference
  - Particle-based approximate inference often plays a key role
- Other applications of particle-based techniques
  - Latent Dirichlet allocation (LDA)
  - Particle filtering – a sequential technique

Correspondence and Data Association: Example Applications

- Matching citations: match citations in text to their referring entities (aka identity resolution or record matching)
- Image registration: match image features in one view to image features in another view
- Matching words: match words in one sentence in one language to their corresponding works in the “same” sentence in another language (aka world alignment)
- Matching genes: match genes in DNA sequence of one organism to similar genes in DNA sequence of another organism

Correspondence and Data Association: Design Decisions

- Probabilistic correspondence: Introduce variables to control mapping between U and V
  - For example, a sensor reading can only come from one aircraft in radar tracking
- Two-sided mutex: Control mapping "both ways"
- Unobserved attributes: Each $u \in U$ has observed attributes, and each $v \in V$ has hidden attributes
  - An aircraft may be observed through radar, audio, video, …
- Directly correlated correspondences: Relationships between different mapping variables
  - Patch with eye is close to patch with nose in one image, they should be close in the next image as well

The Correlated Correspondence Algorithm for Unsupervised Registration of Nonrigid Surfaces

Dragomir Anguelov, Praveen Srinivasan, Hoi-Cheung Pang, Daphne Koller, Sebastian Thrun, James Davis *
Stanford University, Stanford, CA 94305
e-mail:{drago,praveens,hcpang,koller,thrun,jedavis}@cs.stanford.edu

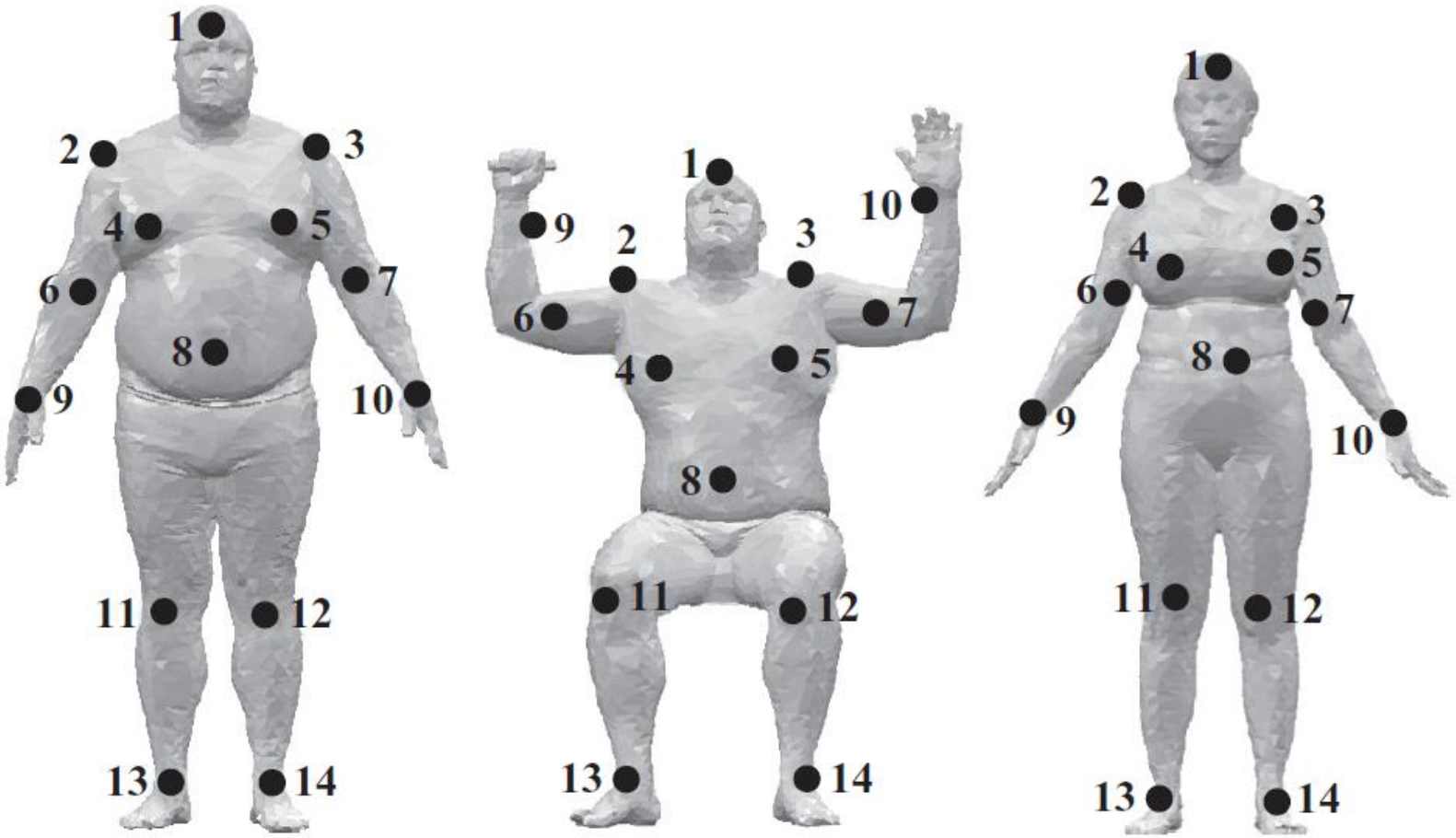
Abstract

We present an unsupervised algorithm for registering 3D surface scans of an object undergoing significant deformations. Our algorithm does not use markers, nor does it assume prior knowledge about object shape, the dynamics of its deformation, or scan alignment. The algorithm registers two meshes by optimizing a joint probabilistic model over all point-to-point correspondences between them. This model enforces preservation of local mesh geometry, as well as more global constraints that capture the preservation of geodesic distance between corresponding point pairs. The algorithm applies even when one of the meshes is an incomplete range scan; thus, it can be used to automatically fill in the remaining surfaces for this partial scan, even if those surfaces were previously only seen in a different configuration. We evaluate the algorithm on several real-world datasets, where we demonstrate good results in the presence of significant movement of articulated parts and non-rigid surface deformation. Finally, we show that the output of the algorithm can be used for compelling computer graphics tasks such as interpolation between two scans of a non-rigid object and automatic recovery of articulated object models.

3D Correspondence

- Construction of 3D models is important in many graphics-oriented applications
  - Range scans of physical objects
- Major difficulty: Determine correspondence of points on one surface to points on another
  - Combinatorially large search problem
- This paper defines a joint probability model over all correspondences
  - Solves it using an approximate inference technique
  - Demonstrates technique on three datasets: 3D scan of a puppet, a human arm, human bodies in different configurations

Correspondence Results for 3D Body Scans



MCMC – Recent Developments (1)

Adaptive MCMC

- Automatically adapt proposal distribution q based on collected samples.
- May affect stationary distribution

Sequential Monte Carlo/Particle Filters

- Time-extended problems, e.g., state estimation in robot navigation
- Use samples to represent distributions
- Use current state distribution to inform proposal distribution for the next step

Sequential importance sampling step

- For $i = 1, ..., N$, sample from the transition priors

$$\widetilde{x}_t^{(i)} \sim q_t\left(\widetilde{x}_t | x_{0:t-1}^{(i)}, y_{1:t}\right)$$

and set

$$\widetilde{x}_{0:t}^{(i)} \triangleq \left(\widetilde{x}_t^{(i)}, x_{0:t-1}^{(i)}\right)$$

- For $i = 1, ..., N$, evaluate and normalize the importance weights

$$w_t^{(i)} \propto \frac{p\left(y_t | \widetilde{x}_t^{(i)}\right) p\left(\widetilde{x}_t^{(i)} | x_{0:t-1}^{(i)}, y_{1:t-1}\right)}{q_t\left(\widetilde{x}_t^{(i)} | x_{0:t-1}^{(i)}, y_{1:t}\right)}.$$

Selection step

- Multiply/Discard particles $\left\{\widetilde{x}_{0:t}^{(i)}\right\}_{i=1}^{N}$ with high/low importance weights $w_t^{(i)}$ to obtain N particles $\left\{x_{0:t}^{(i)}\right\}_{i=1}^{N}$.

MCMC – Recent Developments (2)

Auxiliary Variable Sampling

- Oftentimes easier to sample from *p(x,u)* than *p(x)*
- Include an auxiliary variable u and sample from augmented distribution *p(x,u)* instead
- Then discard u's from samples.
- Algorithms: Hybrid Monte Carlo (HMC), Slice Sampling

Reversible Jump MCMC

- E.g., in Bayesian model selection, the dimensionality of the parameter vector is often different (for example, neural networks with different number of hidden neurons).
- Allows posterior distribution on spaces of varying dimensions
- Algorithm "jumps" between parameters subspaces
- See Green, 1995

Summary

- Particle-based (or sampling) methods are approximate inference methods for P(X | $\boldsymbol{e}$)
- Approximate joint distribution by particles (set of instantiations or configurations) of all or some network nodes
  - Particles need to be designed to give good approximations
- MCMC – combination of Markov Chain and Monte Carlo
- Different algorithms: Forward Sampling, Likelihood Weighting, Metropolis-Hastings, Gibbs sampling, …

Questions?