PROBABILISTIC MODELS – PART 4b: APPROXIMATE INFERENCE IN BAYESIAN NETWORKS

Gerhard Widmer

Institute of Computational Perception
Johannes Kepler University
Linz, Austria

gerhard.widmer@jku.at
www.cp.jku.at/people/widmer

JOHANNES KEPLER UNIVERSITY LINZ

Institute of Computational Perception

October 14, 2025

Presentation partly based on and inspired by [Koller & Friedman, 2009] and [Russell & Norvig, 2021], including the use of some figures from their books and/or lecture slides.

Many thanks to Daphne Koller, Nir Friedman, Stuart Russell, and Peter Norvig for making these available (`pgm.stanford.edu`; `aima.cs.berkeley.edu`).

Goals of this Lecture

- Discuss the need for approximate reasoning in BNs
- Introduce a specific class of approximate reasoning algorithms: Stochastic Sampling
- Show how to sample events from a network
- Show how to answer probabilistic queries based on such samples
- Introduce the general concept of Importance Sampling
- Introduce the framework of Markov Chain Monte Carlo (MCMC) methods
- Describe Gibbs Sampling as an instance of MCMC
- Briefly characterise a general strategy for constructing Markov chains for arbitrary target distributions: the Metropolis-Hastings Algorithm

Outline

Motivation

Remember: The Bad News

- It can be shown that the problem of inference in graphical models is $\mathcal{NP}$**-hard**, and therefore requires **exponential time**, in the worst case.

The (Partly) Good News:

- For some sparsely connected networks (i.e., networks where variables tend to have few parents), inference can be done quite efficiently with optimised algorithms (e.g., ☞ *Variable Elimination*)

But: For many real-world problems, exact inference is impossible.

One Possible Solution: Approximate Inference

General Idea:

- Look for ways to compute **approximate** answer with tractable computational effort.

Different Approaches:

- **Optimisation-based Methods ('Variational Inference'):** Do *exact* inference in a distribution Q that is 'simpler' than target distribution P; search for best approximation distribution Q in a class $\mathcal{Q}$ of 'easy' distributions.
- **Stochastic Sampling Methods:** Do not perform exact inference; answer queries by generating random sample events according to P and estimating probabilities via counting.

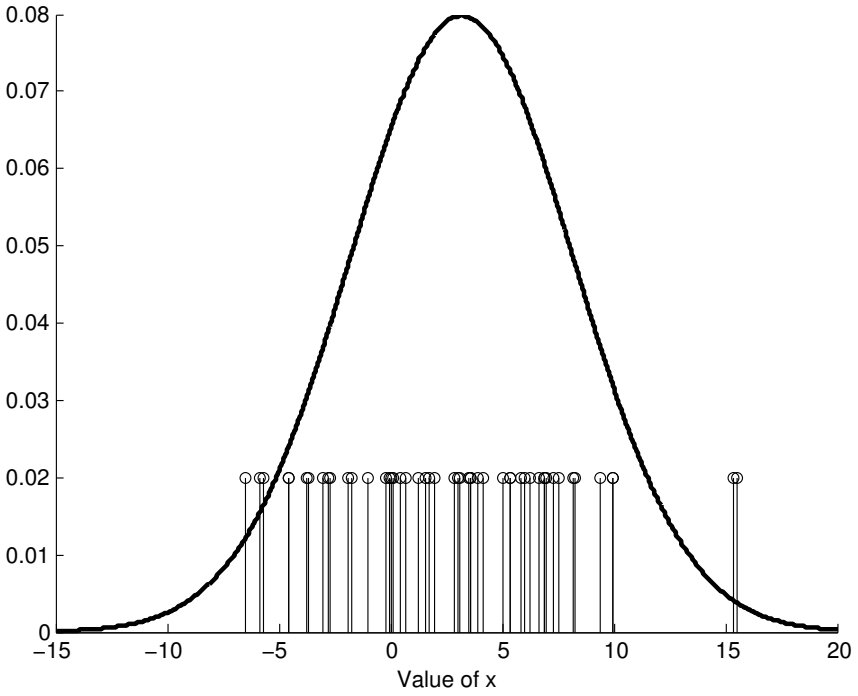
In this Class:

- **Stochastic Sampling** (also known as **'Particle-Based Methods'**)

Approximate Inference via Stochastic Sampling

Basic Idea:

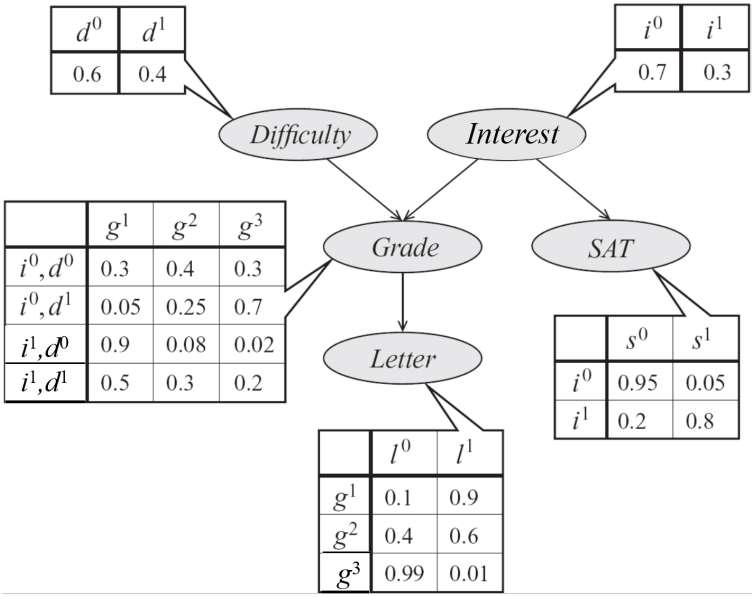
- Instead of exactly computing a probability distribution $P(\mathcal{X})$, approximate it via a limited set of points randomly drawn from P (if that can be done)
- These points are called **(random) samples** or **particles**
- Estimate any desired probability $P(\boldsymbol{x})$ from these samples.



Approximate Inference via Stochastic Sampling

Basic Idea:

- Instead of exactly computing a probability distribution $P(\mathcal{X})$, approximate it via a limited set of points randomly drawn from P (if that can be done)
- These points are called **(random) samples** or **particles**
- Estimate any desired probability $P(x)$ from these samples.



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	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^3	0.99	0.01

$\Rightarrow$ sampled particles (atomic events):

$$
\begin{array}{lllll}
(d^1, & i^0, & g^2, & s^1, & l^0) \\
(d^1, & i^1, & g^1, & s^0, & l^1) \\
(d^0, & i^0, & g^3, & s^0, & l^1) \\
(d^1, & i^0, & g^1, & s^1, & l^1) \\
(.. & .. & .. & .. & ..) \\
(.. & .. & .. & .. & ..)
\end{array}
$$

Sampling Events from a Bayesian Network

Question:

- How to generate a random event according to a distribution $P_{\mathcal{B}}(\mathcal{X})$ that is represented as a Bayesian Network $\mathcal{B}$?

What does it mean to "sample from a distribution P"?

- Want to generate random events in the same way as the **Target Distribution** P would generate events (if it could *generate* events ...)
- The probability with which an event $\boldsymbol{x}$ is generated should correspond to the probability $P(\boldsymbol{x})$ assigned by P
- Events that are highly probable according to P should be generated more often than events with a low probability
- In the limit (for $N \to \infty$), the **Empirical Distribution** $\hat{P}$ (the number of times each event occurs) should converge to the Target Distribution P

☞ A sampling algorithm that satisfies this is called **'consistent'**.

A Step Back: How to Sample a Single Variable

Problem:

- Given a random variable X with distribution $P(X)$, probabilistically select a **value** $x^i \in Val(X)$ according to P

Two Cases with Simple Solutions:

Uniform distribution $P_U(x)$ over a continuous variable $X \in [0, 1]$:

- Use a standard pseudo-random number generator to select a value between 0 and 1.

Discrete (categorical) distribution $P_M(x)$ over variable X with $Val(X) = \{x^1, \ldots, x^k\}$ and k probabilities $\{\theta_1, \ldots, \theta_k\}$, where $\theta_i = P(x^i)$:

- Partition $[0, 1]$ into k subintervals with lengths proportional to these probabilities: $[0, \theta_1), [\theta_1, \theta_1 + \theta_2), \ldots, [\sum_{i=1}^{k-1} \theta_i, 1]$
- Generate a random real number s uniformly from interval $[0, 1]$
- If s is in the i^{th} interval, then the sampled value is x^i

May be tricky or impossible for other distributions (especially continuous ones)

Sampling Events from a Bayesian Network

Question:

- How to generate a random **atomic event** according to a distribution $P_{\mathcal{B}}(\mathcal{X})$ represented as a Bayesian Network $\mathcal{B}$?

The Forward Sampling Algorithm:

- Sample variables independently, one by one, starting at the 'top'

The Forward Sampling Algorithm

Given:

- A Bayesian Network $\mathcal{B}$ over random variables $\mathcal{X}$

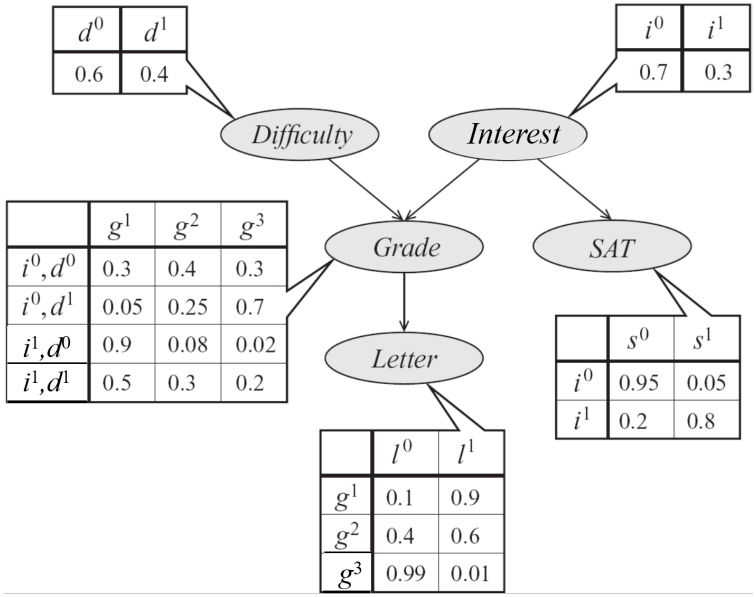
Compute:

- A random event $\boldsymbol{x}$, drawn according to the probability distribution $P_{\mathcal{B}}(\mathcal{X})$

Algorithm:

- Choose a **topological ordering** $\boldsymbol{X} = \{X_1, X_2, ..., X_N\}$ of the $\mathcal{X}$
- For $i = 1$ to N:
  Sample a value x_i from $Val(X_i)$ ***according to the conditional distribution*** $P(X_i \mid \mathrm{pa}(X_i))$, given the parent values $\mathrm{pa}(X_i)$ sampled previously.

Forward Sampling: An Example



Topological ordering: D, I, G, S, L

Sampling steps:

1. Sample D: randomly choose a value from $Val(D) = \{d^0, d^1\}$ acc. to $P(D) = [0.6; 0.4]$
   $\Rightarrow$ suppose we get $\underline{d^1}$

2. Sample I from $\{i^0, i^1\}$ acc. to $P(I) = [0.7; 0.3]$
   $\Rightarrow$ suppose we get $\underline{i^0}$

3. Sample G from $\{g^1, g^2, g^3\}$ acc. to $P(G \mid \underline{d^1, i^0}) = [0.05; 0.25; 0.7]$
   $\Rightarrow$ suppose we get $\underline{g^3}$

4. Sample S from $\{s^0, s^1\}$ acc. to $P(S \mid \underline{i^0}) = [0.95; 0.05]$
   $\Rightarrow$ suppose we get $\underline{s^0}$

5. Sample L from $\{l^0, l^1\}$ acc. to $P(L \mid \underline{g^3}) = [0.99; 0.01]$
   $\Rightarrow$ suppose we get $\underline{l^1}$

Result:

Random sample (event)

$$\boxed{(d^1, i^0, g^3, s^0, l^1)}$$

Proof: Forward Sampling is Consistent

To be shown:

- Forward Sampling is **consistent**
- **In words:** The probability distribution P_S according to which the samples are generated (the ***Sampling Distribution***) is equal to the true distribution $P_{\mathcal{B}}$ represented by the Bayesian network.

Proof (trivial):

- Let $P_S(x_1, \ldots, x_n)$ = probability that a specific event $(x_1, \ldots, x_n)$ is generated by the Forward Sampling algorithm
- Observation 1: Each variable X_i is sampled according to $P(X_i \mid \mathrm{pa}(X_i))$
- Observation 2: Each variable is sampled *independently* of the other variables (except for using the variable's previously sampled parent values)
- Consequence:

$$P_S(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(X_i \mid \mathrm{pa}(X_i)) = P_{\mathcal{B}}(x_1, \ldots, x_n). \qquad \square$$

Computational Complexity of Forward Sampling

Let

$$\begin{aligned} n &= |\mathcal{X}| \\ k &= \max_i |\mathrm{Pa}(X_i)| \\ d &= \max_i |Val(X_i)| \end{aligned}$$

Then the cost for

- indexing into the appropriate CPD $P(X_i \mid \mathrm{pa}(X_i))$ of a variable X_i, given specific values of its parents, is $O(|\mathrm{Pa}(X_i)|) \leq O(k)$
- sampling a value for a discrete variable X_i from a CPD $P(X_i \mid \mathrm{pa}(X_i))$ is $O(\log |Val(X_i)|) \leq O(\log d)$ (via binary search)
- sampling a complete event is $O(nk \log d)$

☞ **Generating one random sample event can be done in time**

$$\boxed{O(nk \log d)}$$

Approximate Inference (1): Estimating the Full Joint Distribution

Problem:

- Estimate the probability $P_{\mathcal{B}}(x_1, \ldots, x_n)$ of an **atomic event** $(x_1, \ldots, x_n)$, where $\{X_1, \ldots, X_n\} = \mathcal{X}$

Solution:

- Sample a set $\mathcal{D}$ of M events randomly from $\mathcal{B}$ via Forward Sampling
- Count the number of times K that our event $(x_1, \ldots, x_n)$ appears in $\mathcal{D}$
- Estimate $P_{\mathcal{B}}(x_1, \ldots, x_n)$ via its relative frequency:

$$\hat{P}_{\mathcal{D}}(x_1, \ldots, x_n) = \frac{K}{M}$$

☞ **Can estimate the entire Full Joint Distribution (theoretically ...)**

Approximate Inference (2): Estimating the Probability of an Arbitrary Event

Trivial generalisation to probability estimation for **general (not necessarily atomic)** events:

Problem:

- Estimate the probability $P_{\mathcal{B}}(x_1, \ldots, x_k)$ of an event $(x_1, \ldots, x_k)$, where $\{X_1, \ldots, X_k\} \subseteq \mathcal{X}$

Solution:

- Sample a set $\mathcal{D}$ of M atomic events randomly from $\mathcal{B}$ via Forward Sampling
- Count the number K of events $\boldsymbol{e}_i$ in $\mathcal{D}$ that **match** our event $(x_1, \ldots, x_k)$ (that is: where $\boldsymbol{e}_i\langle X_1, \ldots, X_k\rangle = (x_1, \ldots, x_k)$)
- Estimate $P_{\mathcal{B}}(x_1, \ldots, x_k)$ via the relative frequency

$$\hat{P}_{\mathcal{D}}(x_1, \ldots, x_k) = \frac{K}{M}$$

Examples

Query 1 (atomic event): $\boxed{P(d^1, i^0, g^3, s^0, l^1) = ?}$

Suppose we sample $M = 1000$ events, of which $K = 27$ events have precisely the values $(d^1, i^0, g^3, s^0, l^1)$

$$\hat{P}(d^1, i^0, g^3, s^0, l^1) = \frac{27}{1000} = 0.027$$

Query 2: $\boxed{P(i^0, l^1) = ?}$

Suppose we sample $M = 1000$ events, of which $K = 199$ events have values $\{i^0, l^1\}$ for $\{I, L\}$ (and arbitrary values for the other variables $\{D, G, S\}$)

$$\hat{P}(i^0, l^1) = \frac{199}{1000} = 0.199$$

Approximate Inference (3): Probabilistic Queries

Remember:

A **Probabilistic Query** involves computing the conditional distribution

$$P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e}) \quad \text{(or } P(\boldsymbol{X} \mid \boldsymbol{e}) \text{ for short)}$$

for some sets of variables $\boldsymbol{X}, \boldsymbol{E} \subseteq \mathcal{X}$ and a specific value assignment $\boldsymbol{E} = \boldsymbol{e}$.

☞ Estimating $P(\boldsymbol{X} \mid \boldsymbol{e})$ means estimating $P(\boldsymbol{x} \mid \boldsymbol{e})$ for all value combinations $\boldsymbol{x}$.

Overview of the methods to be presented in the following:

- Simple Idea 1: Rejection Sampling (RS)
- Simple Idea 2: Likelihood Weighting (LW)
- More general view of LW: Importance Sampling
- Less simple Idea 3: Getting closer to the posterior via MCMC
- Specific realisation: Gibbs Sampling
- General method to create appropriate Markov Chains: The Metropolis-Hastings Algorithm

Estimating $P(\boldsymbol{X}|e)$ [1]: Rejection Sampling

Idea:

- Produce a set of samples from $P_{\mathcal{B}}$
- Keep only samples consistent with evidence $\boldsymbol{E} = \boldsymbol{e}$
- Estimate probabilities by counting relative frequencies (for each $\boldsymbol{x}$)

Justification:

$$P(\boldsymbol{x} \mid \boldsymbol{e}) = \frac{P(\boldsymbol{x}, \boldsymbol{e})}{P(\boldsymbol{e})}$$

and we can estimate both $P(\boldsymbol{x}, \boldsymbol{e})$ and $P(\boldsymbol{e})$ as before:

$$\hat{P}(\boldsymbol{x} \mid \boldsymbol{e}) = \frac{\frac{N(\boldsymbol{x},\boldsymbol{e})}{M}}{\frac{N(\boldsymbol{e})}{M}} = \frac{N(\boldsymbol{x}, \boldsymbol{e})}{N(\boldsymbol{e})}$$

	$N(\boldsymbol{x}, \boldsymbol{e})$	... number of samples with values $(\boldsymbol{x}, \boldsymbol{e})$
where	$N(\boldsymbol{e})$	... number of samples with values $\boldsymbol{e}$
	M	... total number of samples.

Estimating $P(X|e)$ [1]: Rejection Sampling

The Rejection Sampling Algorithm

Given:

- A Bayesian network $\mathcal{B}$ over $\mathcal{X}$
- A query $P(\boldsymbol{X} \mid \boldsymbol{e})$ with $\boldsymbol{X}, \boldsymbol{E} \subseteq \mathcal{X}$

Algorithm:

1. **Sample:** Probabilistically sample a set $\mathcal{D}$ of M events from $\mathcal{B}$
2. **Filter / Reject:**
   Discard ("reject") all samples not compatible with the evidence $\boldsymbol{e}$
   $\Rightarrow$ the remaining samples $\mathcal{D}_{\boldsymbol{e}} \subseteq \mathcal{D}$ represent the distribution $P(\boldsymbol{X} \mid \boldsymbol{e})$
3. **Estimate:** For each value combination $\boldsymbol{X} = \boldsymbol{x}$, count the number K of samples $\in \mathcal{D}_{\boldsymbol{e}}$ that match $\boldsymbol{x}$. Estimate $P(\boldsymbol{x} \mid \boldsymbol{e})$ as

$$\hat{P}(\boldsymbol{x} \mid \boldsymbol{e}) = \frac{K}{|\mathcal{D}_{\boldsymbol{e}}|}$$

Rejection Sampling: An Example

Query: $\boxed{P(i^1 \mid g^2, s^1, l^0) = ?}$

Suppose

- we sample a set $\mathcal{D}$ of $M = 1000$ events
- 916 of these are *not* consistent with evidence $G = g^2 \wedge S = s^1 \wedge L = l^0$
  $\Rightarrow \mathcal{D}_e$ contains 84 events
- $K = 31$ of these have $I = i^1$

Answer:

$$\hat{P}(i^1 \mid g^2, s^1, l^0) = \frac{31}{84} \approx 0.369$$

Rejection Sampling: Pros and Cons

Advantages:

- Extremely simple algorithm
- Is consistent: produces provably correct estimate (in the limit)

Disadvantages / Problems:

- Wasteful: Immediately discards a large part of the generated samples
- More precisely: Expected number of samples that are *not* rejected (in sample set $\mathcal{D}$ of size M) is $M \cdot P(\boldsymbol{e})$
- Example: with $P(\boldsymbol{e}) = 0.001$, would need to generate $100,000$ samples to obtain 100 samples useful for estimation!
- Probability of evidence $\boldsymbol{e}$ decreases exponentially with $|\boldsymbol{E}|$

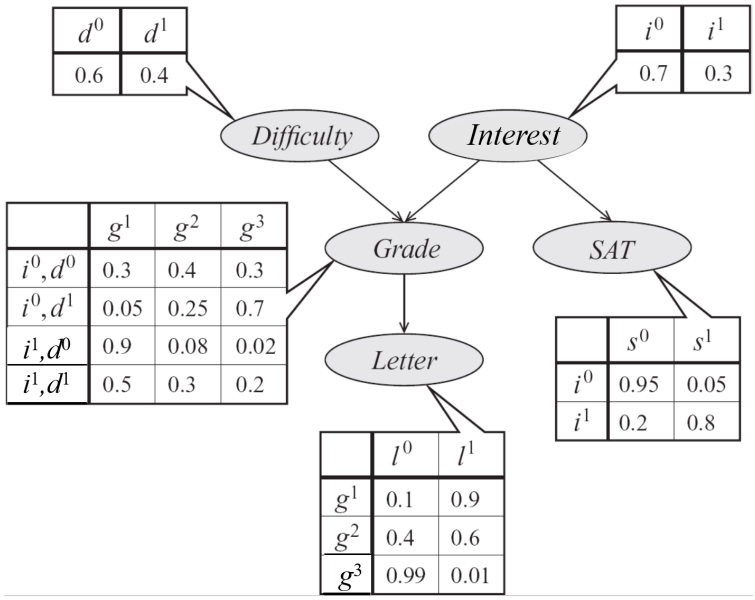
Source of the problem:

- Want to estimate the *posterior distribution* $P(\boldsymbol{X} \mid \boldsymbol{e})$
- ... but draw our samples $\mathcal{D}$ from the *full joint* distribution $P(\mathcal{X})$ (may be very different from the posterior)
- Need to fix this by reducing $\mathcal{D}$ to $\mathcal{D}_{\boldsymbol{e}}$, which represents our target $P(\boldsymbol{X} \mid \boldsymbol{e})$

☞ **In the following:** Try to sample from distributions closer to the posterior ...

Estimating $P(\boldsymbol{X}|e)$ [2]: Likelihood Weighting

Motivation:



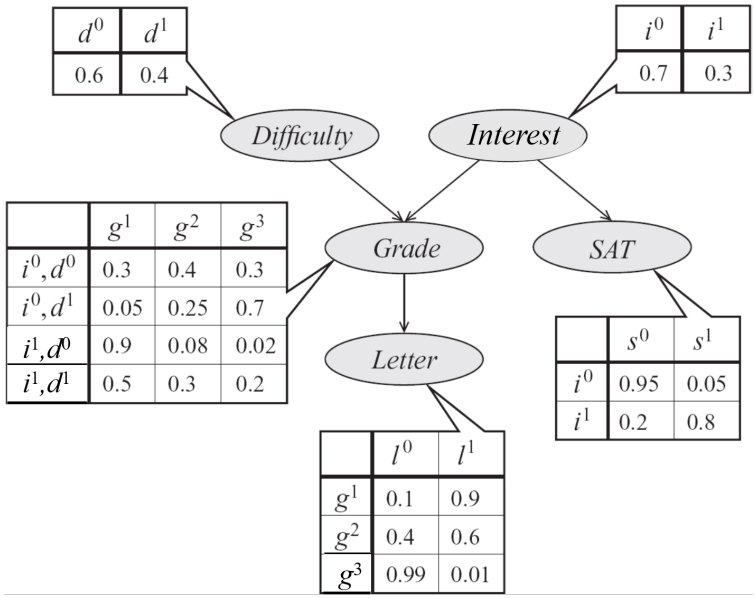
Query: $P(L \mid d^1, s^1) = ?$

- Forward Sampling might start by generating a value d^0 for D
- No matter how the sampling proceeds, the resulting event will eventually be rejected!

☞ **... so why not just:**

- Set evidence variables $\boldsymbol{E}$ to their observed values $\boldsymbol{e}$
  and only sample the remaining variables $\mathcal{X} - \boldsymbol{E}$ to directly obtain $\mathcal{D}_{\boldsymbol{e}}$

Problem with Naive Approach



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	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^3	0.99	0.01

Query: $P(L \mid d^1, s^1) = ?$

Naive Idea:

1. Set $D = d^1, S = s^1$
2. Sample I from its prior distribution
3. Sample G and L accordingly.

- All samples will have $D = d^1, S = s^1$, as desired
- Around 30% of the samples will have $I = i^1$
  (because I is sampled from its prior distribution)
- **Problem:** True probability of $I = i^1$ in situations with $S = s^1$ is much higher:

$$P(i^1 \mid s^1) \gg P(i^1)$$

- Naive algorithm is unaware of this: it ignores evidence s^1 when sampling I
- Sampling distribution will be wrong and not reflect $P(I, G, L \mid d^1, s^1)$

Likelihood Weighting (LW): The Intuition

Consider this:

- Run standard Forward Sampling many times
- $P(s^1 \mid i^1) = .8$
  $\Rightarrow$ Samples starting with $I = i^1$ would generate $S = s^1$ in 80% of the cases
- $P(s^1 \mid i^0) = .05 \quad \Rightarrow$ Samples with $I = i^0$ would generate $S = s^1$ in only 5%
- Observing $S = s^1$ (evidence) in $I = i^1$ is $16\times$ as probable as when $I = i^0$

Idea:

- Simulate this by **sample weights** that reflect the probability of the evidence in a given context
- Example: Sample with $I = i^1$ and fixed evidence $S = s^1$ should be worth 80% of a sample, one with $I = i^0$ and $S = s^1$ only 5%
- Weight contributed by evidence variable S corresponds to $P(s^1 \mid I)$
  (in general: $P(e_i \mid \mathrm{pa}(E_i))$
- Variables are sampled independently $\Rightarrow$ weight for entire sample $=$ product of individual weights contributed by evidence variables.

☞ **Weighted Particles**

Likelihood Weighting (LW): The Algorithm

The Likelihood Weighting Algorithm

Given:

- A Bayesian network $\mathcal{B}$ over $\mathcal{X}$
- A query $P(\boldsymbol{X} \mid \boldsymbol{e})$ with $\boldsymbol{X}, \boldsymbol{E} \subseteq \mathcal{X}$

Algorithm:

1. **Sample:** Generate a set $\mathcal{D}_{\boldsymbol{e}}$ of **weighted samples** from $\mathcal{B}$ (all consistent with $\boldsymbol{e}$), via algorithm SAMPLE-LW (see next slide)
2. **Estimate:** For each value combination $\boldsymbol{X} = \boldsymbol{x}$, select the subset $\mathcal{D}_{\boldsymbol{e}}^{\boldsymbol{x}} \subseteq \mathcal{D}_{\boldsymbol{e}}$ of samples that match $\boldsymbol{x}$. Estimate $P(\boldsymbol{x} \mid \boldsymbol{e})$ as

$$\hat{P}(\boldsymbol{x} \mid \boldsymbol{e}) = \frac{\sum_{s_i \in \mathcal{D}_{\boldsymbol{e}}^{\boldsymbol{x}}} w[s_i]}{\sum_{s_i \in \mathcal{D}_{\boldsymbol{e}}} w[s_i]}$$

where $w[s_i]$ is the weight of sample s_i

Drawing a Weighted Sample

The SAMPLE-LW Algorithm

Given:

- A Bayesian network $\mathcal{B}$ over $\mathcal{X}$
- Fixed evidence $\boldsymbol{E} = \boldsymbol{e}$

Compute:

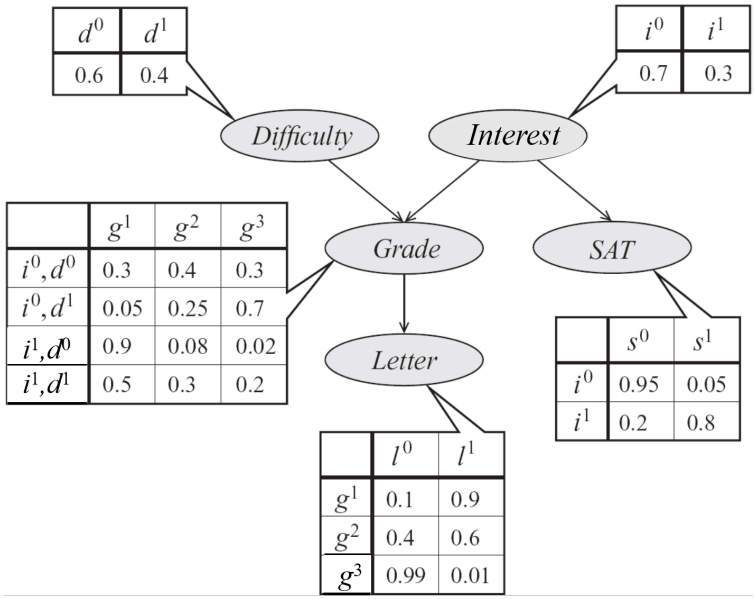
- A weighted sample $\langle (\boldsymbol{y}, \boldsymbol{e}), w \rangle$ (where $\boldsymbol{Y} \cup \boldsymbol{E} = \mathcal{X}$)

Algorithm:

- Choose a **topological ordering** $\boldsymbol{X} = \{X_1, X_2, \ldots, X_N\}$ of the $\mathcal{X}$
- Initialise weight w of current sample to $w = 1.0$
- For $i = 1$ to N:
  if X_i is one of the evidence variables $E_j \in \boldsymbol{E}$:
  set X_i to its given evidence value e_j
  update weight: $w \leftarrow w \times P(e_j \mid \mathrm{pa}(X_i))$
  else
  sample a value for X_i from $Val(X_i)$ according to $P(X_i \mid \mathrm{pa}(X_i))$
  (as in Forward Sampling)

SAMPLE-LW: An Example

Query: $P(I \mid g^3, s^1) = ?$



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	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^3	0.99	0.01

Result:

Weighted random sample:

$$\langle (d^1, i^0, g^3, s^1, l^1), 0.035 \rangle$$

Topological ordering: D, I, G, S, L

1. Initialise weight: $w \leftarrow 1.0$
2. Sample D from $\{d^0, d^1\}$ acc. to $P(D) = [0.6; 0.4]$
   $\Rightarrow$ suppose we get $\underline{d^1}$
3. Sample I from $\{i^0, i^1\}$ acc. to $P(I) = [0.7; 0.3]$
   $\Rightarrow$ suppose we get $\underline{i^0}$
4. G is given (evidence): Set G to $\underline{g^3}$ and update weight: $w \leftarrow w \times P(g^3 | d^1, i^0) = 1.0 \times 0.7 = 0.7$
5. S is given (evidence): Set S to $\underline{s^1}$ and update weight: $w \leftarrow w \times P(s^1 | i^0) = 0.7 \times 0.05 = 0.035$
6. Sample L from $\{l^0, l^1\}$ acc. to $P(L | \underline{g^3}) = [0.99; 0.01]$
   $\Rightarrow$ suppose we get $\underline{l^1}$

Correctness Proof of SAMPLE-LW

- Think of $\mathcal{X} = \boldsymbol{Y} \cup \boldsymbol{E}$ (the $\boldsymbol{Y}$ are sampled, $\boldsymbol{E} = \boldsymbol{e}$ given)
- Probability P_{LW} that SAMPLE-LW generates a sample $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{e})$ is

$$P_{LW}(\boldsymbol{y}, \boldsymbol{e}) = \prod_{Y_i \in \boldsymbol{Y}} P(y_i \mid \mathrm{pa}(Y_i)) \prod_{E_i \in \boldsymbol{E}} 1.0 = \prod_{Y_i \in \boldsymbol{Y}} P(y_i \mid \mathrm{pa}(Y_i))$$

- Weight of sample $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{e})$ is

$$w(\boldsymbol{y}, \boldsymbol{e}) = \prod_{E_i \in \boldsymbol{E}} P(e_i \mid \mathrm{pa}(E_i))$$

- **Weighted probability** of a sample [1] is

$$\begin{aligned} \underline{P_{LW}(\boldsymbol{y}, \boldsymbol{e}) \times w(\boldsymbol{y}, \boldsymbol{e})} &= \prod_{Y_i \in \boldsymbol{Y}} P(y_i \mid \mathrm{pa}(Y_i)) \prod_{E_i \in \boldsymbol{E}} P(e_i \mid \mathrm{pa}(E_i)) \\ &= \prod_{X_i \in \mathcal{X}} P(x_i \mid \mathrm{pa}(X_i)) \\ &= \underline{P(\boldsymbol{y}, \boldsymbol{e})} \propto \underline{P(\boldsymbol{y} \mid \boldsymbol{e})} \qquad \square \end{aligned}$$

[1] A sample influences the final probability estimate for the query through the probability with which it will appear, and through its weight.

Likelihood Weighting: Pros and Cons

Advantages:

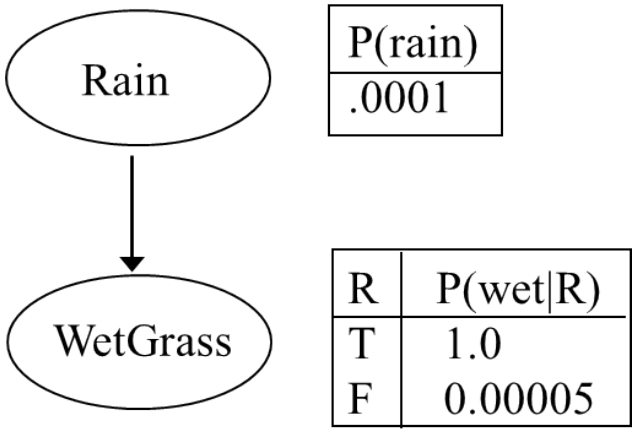
- Consistent: produces provably correct estimate
- Less wasteful than Rejection Sampling:
  Generates only samples consistent with the evidence.

Disadvantages / Problems:

- Evidence affects sampling only for nodes *below* the $\boldsymbol{E}$ variables ...
- ... but **not** for nodes *above* the evidence
  (more generally: for any non-descendants of the $\boldsymbol{E}$)
- In above example: still sample I ignoring the fact that $S = s^1$
- Consequence: many samples will be nearly irrelevant to the given evidence;
  (correctly) reflected in tiny weights
- Only few samples typical for the evidence will be available for estimation
- Particularly bad when most evidence variables are 'downstream' in the net
  (which is often the case in causally directed networks).

An Extreme Example

Consider this model of the climate in Death Valley:



P(rain)
.0001

R	P(wet|R)
T	1.0
F	0.00005

... and the query

$$P(Rain \mid wet) = ?$$

- LW will start by sampling $Rain$
- Might produce thousands of samples $(\neg rain, wet)$
  (and correctly attach low weight 0.00005 to each of them)
  ... before it happens to generate even *one* sample $(rain, wet)$
  (which is much more typical given the evidence: weight $w(rain, wet) = 1.0$)
- In fact, when drawing 5000 random samples, we might not get *even a single* sample $(rain, wet)$!

A More General View: Importance Sampling

General Goal:

- Draw samples from some distribution P

Problem:

- May be impossible (or undesirable) to *sample* from P directly
- But: may be possible to *evaluate* P pointwise (compute $P(\boldsymbol{x})$ for a given $\boldsymbol{x}$)

Idea of Importance Sampling:

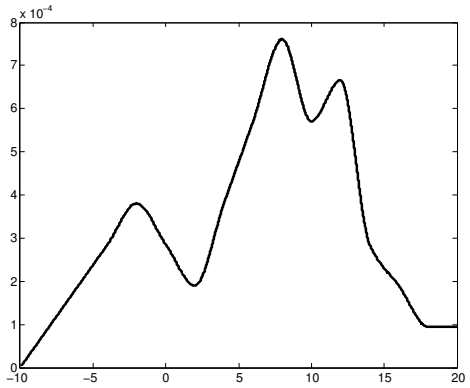
- Generate samples from a different ('easier') distribution Q instead of P
- Correct for difference between Q and P by attaching **weights** to samples: high weights where Q under-estimates P, low weights where Q over-estimates P

Terminology:

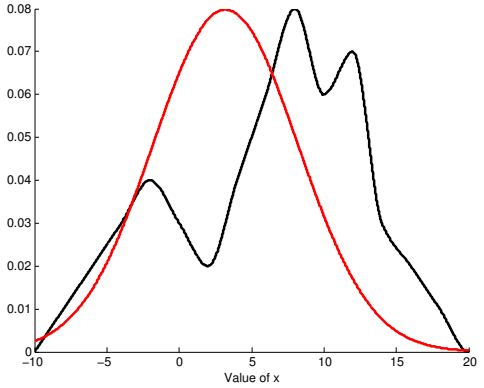
- P is called **Target Distribution**
- Q is called **Proposal Distribution**

Importance Sampling

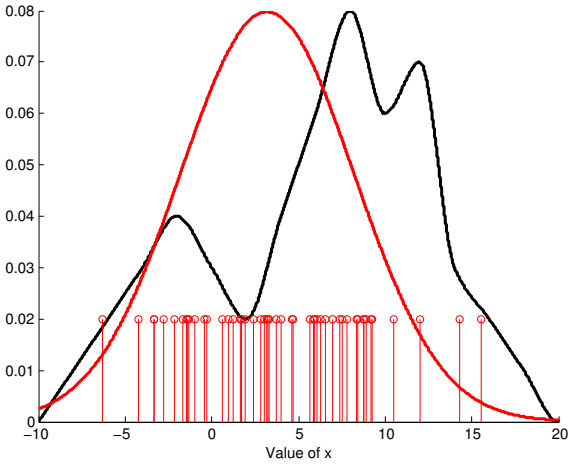
Target distribution P



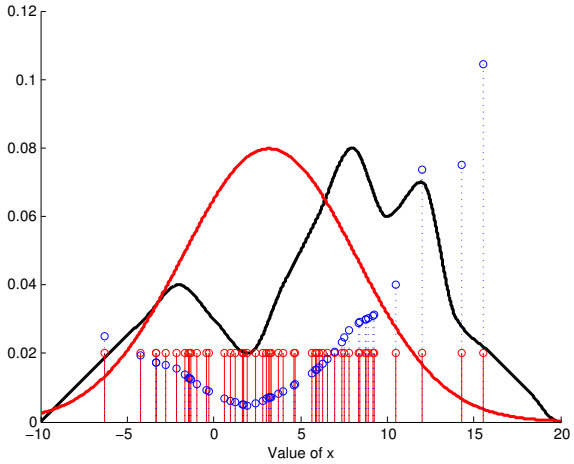
Proposal distribution Q



Points sampled from Q



Weights to correct difference $P : Q$



Importance Sampling

Computing the Weight:

- $w(\boldsymbol{x}_i)$ of sample $\boldsymbol{x}_i$ must correct for the fact that $\boldsymbol{x}_i$ was drawn from the 'wrong' distribution Q
- Correct weight turns out to be

$$w(\boldsymbol{x}_i) = \frac{P(\boldsymbol{x}_i)}{Q(\boldsymbol{x}_i)}$$

Proof:

- omitted (but simple)

Intuition:

- If at some point $\boldsymbol{x}$ the probability/density according to P is n times higher than Q (from which the samples were drawn), then P would generate n times as many samples around $\boldsymbol{x}$ as would Q
- The Q sample at $\boldsymbol{x}$ must be "upweighted" to count for n samples of P.

Likelihood Weighting as an Instance of Importance Sampling

Reconsider the sampling step SAMPLE-LW **in Likelihood Weighting:**

Given a query $P(\boldsymbol{X} \mid \boldsymbol{e})$, SAMPLE-LW

- sets all evidence variables E_i to $E_i = e_i$
- samples the remaining variables $\boldsymbol{Y} = \mathcal{X} - \boldsymbol{E}$ in topological order (and weights the resulting sample by $\prod_i P(e_i \mid \mathrm{pa}(E_i))$).

Question:

- What distribution is SAMPLE-LW actually drawing its samples from?

Definition

Let $\mathcal{B}$ be a network and $\boldsymbol{E} = \boldsymbol{e}$ fixed evidence $(E_1 = e_1, \ldots, E_k = e_k)$

We define the **Mutilated Network** $\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}$ as follows:

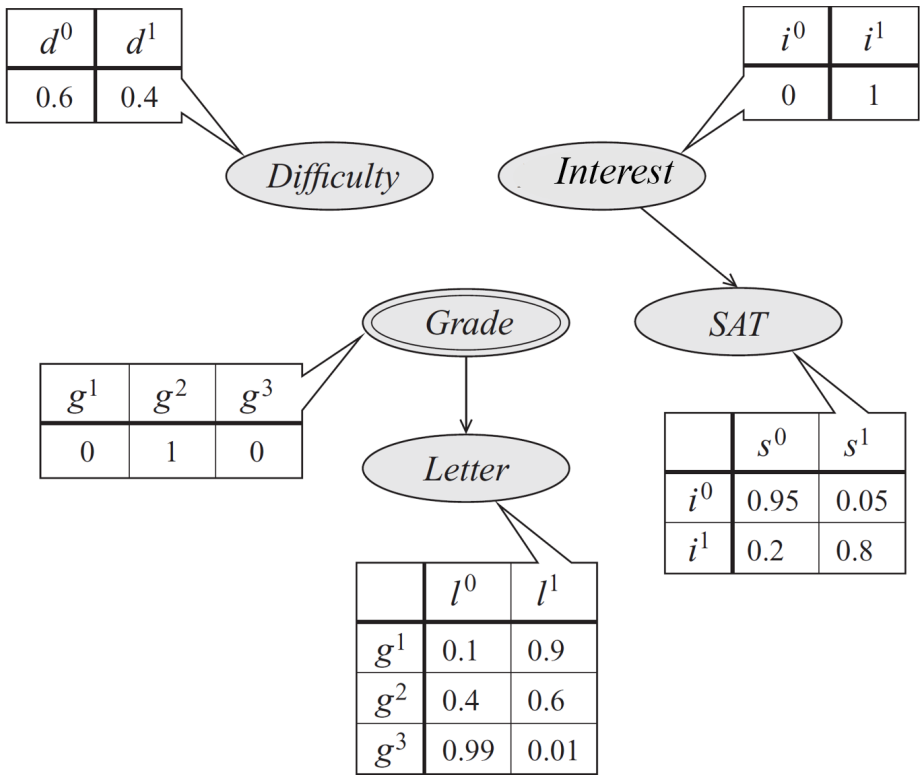
- Variables E_i have *no parents* in $\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}$
- The CPD of each E_i in $\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}$ gives probability 1.0 to $E_i = e_i$ and probability 0 to all other $e_i' \in Val(E_i)$
- The parents and CPDs of all other variables $X \notin \boldsymbol{E}$ are as in $\mathcal{B}$

An Example

Query:

$$P(D \mid i^1, g^2) = ?$$

Corresponding mutilated network $\mathcal{B}_{I=i^1, G=g^2}$



d^0	d^1
0.6	0.4

i^0	i^1
0	1

g^1	g^2	g^3
0	1	0

	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^3	0.99	0.01

Corresponding distribution that we actually sample from:

$$P_{\mathcal{B}_{I=i^1, G=g^2}}(D, S, L) = P(D) \cdot P(S \mid i^1) \cdot P(L \mid g^2) = \prod_{Y_i \in \mathcal{X} - \boldsymbol{E}} P(y_i \mid \mathrm{pa}(Y_i))$$

Likelihood Weighting as an Instance of Importance Sampling

THEOREM

Let $\langle \boldsymbol{x}, w(\boldsymbol{x}) \rangle$ be a weighted sample generated by SAMPLE-LW.
Then the distribution over $\boldsymbol{x}$ is as defined by the network $\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}$ and

$$w(\boldsymbol{x}) = \frac{P_{\mathcal{B}}(\boldsymbol{x})}{P_{\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}}(\boldsymbol{x})}$$

IN WORDS:

- Instead of sampling from the full joint distribution $P_{\mathcal{B}}(\mathcal{X})$[2] SAMPLE-LW samples from $P_{\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}}(\mathcal{X})$ (which is guaranteed to be consistent with $\boldsymbol{E} = \boldsymbol{e}$)
- $P_{\mathcal{B}}$ is the **target distribution** P, $P_{\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}}$ the **proposal distribution** Q
- Correct for use of Q to estimate P via weight

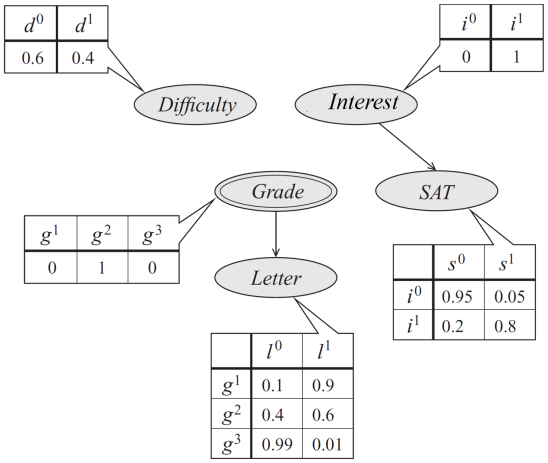
$$\boxed{w(\boldsymbol{x})} = \prod_{E_i \in \boldsymbol{E}} P(e_i \mid \mathrm{pa}(E_i)) = \frac{\prod_{X_i \in \mathcal{X}} P(x_i \mid \mathrm{pa}(X_i))}{\prod_{Y_i \in \mathcal{X} - \boldsymbol{E}} P(y_i \mid \mathrm{pa}(Y_i))} = \frac{P_{\mathcal{B}}(\boldsymbol{x})}{P_{\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}}(\boldsymbol{x})} = \boxed{\frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})}}$$

[2] which represents the true probabilities over world $\mathcal{X}$, but which we do not *want to* sample from, because it gives us lots of useless samples with non-matching evidence values

The Problem with Likelihood Weighting, Revisited

Query:

$$P(D \mid i^1, g^2) = ?$$



d^0	d^1
0.6	0.4

i^0	i^1
0	1

g^1	g^2	g^3
0	1	0

	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^3	0.99	0.01

Observations:

- Proposal distribution $P_{\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}}(\mathcal{X})$ can be **very** different from $P_{\mathcal{B}}(\mathcal{X})$
- Does not reflect influence of fixed evidence on **non-descendants** of the $\boldsymbol{E}$
- Example: D is sampled without taking into account that $G = g^2$
- Gets worse when many evidence variables are **'downstream'** in the network
- Consider the mutilated network for query $P(D \mid l^0)$ ...

Consequences:

- Extremely low weights for many samples
- Only a few samples typical for the evidence available for estimation.

Summary of the Dilemma So Far

The *real* target distribution we would like to sample from (but cannot):

$$\boxed{P_{\mathcal{B}}(\boldsymbol{Y} \mid \boldsymbol{e})}$$

Rejection Sampling:

- samples from full joint distribution $P_{\mathcal{B}}(\mathcal{X})$
  (which is correct, but produces many samples that don't match the evidence)
- has to reduce sample set to $\mathcal{D}_{\boldsymbol{e}}$ to obtain samples $(\boldsymbol{y}, \boldsymbol{e})$ from $P_{\mathcal{B}}(\boldsymbol{Y} \mid \boldsymbol{e})$

Likelihood Weighting / Importance Sampling:

- samples from proposal distribution $P_{\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}}(\mathcal{X})$
  (which produces only samples with given evidence, but can be very wrong)
- has to correct for wrong proposal distribution by weights (which can be tiny)

Question:

- Can't we do better?
- Can we sample directly from our target $P_{\mathcal{B}}(\boldsymbol{Y} \mid \boldsymbol{e})$ (or "almost" ...),
  without explicitly calculating it (which would require solving the query)?

Markov Chain Monte Carlo (MCMC) Sampling

General Goal:

- **Sample directly from** $P_{\mathcal{B}}(\boldsymbol{Y} \mid \boldsymbol{e})$ (= our *real* target)
- Then all samples would have the correct $\boldsymbol{e}$, and would be equally relevant ...

Basic Idea in MCMC:

- Construct a **sequence of samples** $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots$ iteratively, with
- $\boldsymbol{x}^{(0)}$ drawn from some proposal distribution $Q^{(0)}$ and[3]
- the distribution $Q^{(i+1)}$ of the next sample $\boldsymbol{x}^{(i+1)}$ depending on the current sample $\boldsymbol{x}^{(i)}$ [4]
- Do this such that $Q^{(i+1)}$ is **closer to the target distribution** than $Q^{(i)}$

Consequence:

- Implicitly, we construct a sequence of proposal distributions $Q^{(0)}, Q^{(1)}, Q^{(2)}$... that **converge** to our target distribution $P_{\mathcal{B}}(\boldsymbol{Y}|\boldsymbol{e})$
- Later samples are "almost" drawn from $P_{\mathcal{B}}(\boldsymbol{Y}|\boldsymbol{e}) \Rightarrow$ no weighting necessary!

[3] $Q^{(0)}$ might be the mutilated network $\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}$

[4] That makes the sequence of distributions a **Markov chain** – see later.

A Simple MCMC Algorithm: Gibbs Sampling

Given:

- Probabilistic query $P(\boldsymbol{X} \mid \boldsymbol{e})$

Goal:

- Generate a random sample from target distribution $P(\boldsymbol{Y} \mid \boldsymbol{e})$

Basic Idea:

- Start with a sample $\boldsymbol{x}^{(0)} = (\boldsymbol{y}^{(0)}, \boldsymbol{e})$ drawn from mutilated network $\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}$ (i.e., use SAMPLE-LW, but ignore weight)
- Iteratively try to "fix" (improve) the current sample $\boldsymbol{x}^{(t)}$ by ...

... iterating over each non-evidence variable Y_i and **re-sampling** a value for it, **given the current values of all the other variables**

- Allows information to "flow" across network (from evidence to non-evidence variables).

The Gibbs Sampling Algorithm

Gibbs Sampling Algorithm

Given:

- Probabilistic query $P(\boldsymbol{X} \mid \boldsymbol{e})$
- Let $\mathcal{X} = \boldsymbol{Y} \cup \boldsymbol{E}$

Algorithm:

- Sample $\boldsymbol{x}^{(0)} = (\boldsymbol{y}^{(0)}, \boldsymbol{e})$ from $\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}$
- For $t = 1, \ldots, T$ do:
  $\boldsymbol{x}^{(t)} \leftarrow \boldsymbol{x}^{(t-1)}$
  for each $Y_i \in \boldsymbol{Y}$:
  Resample $y_i^{(t)}$ according to $P(Y_i \mid \boldsymbol{x}_{-y_i}^{(t)})$ % replace value of Y_i in $\boldsymbol{x}^{(t)}$
- Return $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$

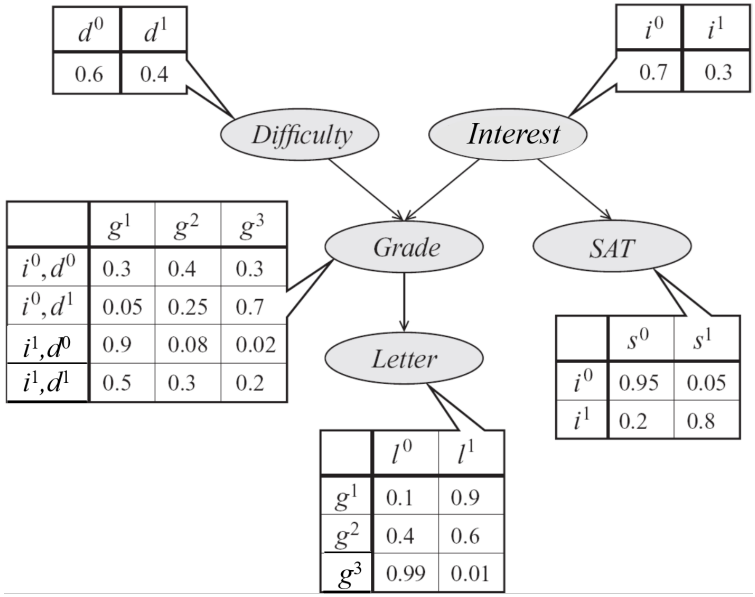
Central Questions:

- What does the **Resampling Distribution** $P(Y_i \mid \boldsymbol{x}_{-y_i}^{(t)})$ look like?
- Can we effectively calculate it (so that we can sample from it)?
- ... and why does this make sense at all??

Gibbs Sampling: An Example

Query: $P(I \mid s^1, l^0) = ?$

Step 0: Generate first sample: Suppose $\boldsymbol{x}^{(0)} = (d^1, i^0, g^2, \underline{s^1}, \underline{l^0})$



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	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^3	0.99	0.01

One Gibbs Sampling Iteration:

Loop over $\boldsymbol{Y} = \{G, I, D\}$:

1. Resample G acc. to $P(G \mid d^1, i^0, s^1, l^0)$
   $\Rightarrow$ suppose we get g^3
2. Resample I acc. to $P(I \mid d^1, g^3, s^1, l^0)$
   $\Rightarrow$ suppose we get i^1
3. Resample D acc. to $P(D \mid i^1, g^3, s^1, l^0)$
   $\Rightarrow$ suppose we get d^1

Result: New sample $\boldsymbol{x}^{(1)} = (d^1, i^1, g^3, \underline{s^1}, \underline{l^0})$

Computing the Re-Sampling Distribution

Central Question: **How to compute and sample from** $P(G \mid d^1, i^0, s^1, l^0)$ **?**

Sampling a single variable, given all others, turns out to be easy:

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

$$= \frac{P(d^1)\ P(i^0)\ P(G|d^1, i^0)\ P(s^1|i^0)\ P(l^0|G)}{\sum_g P(d^1)\ P(i^0)\ P(g|d^1, i^0)\ P(s^1|i^0)\ P(l^0|g)}$$

$$= \frac{P(d^1)\ P(i^0)\ P(s^1|i^0) \quad P(G|d^1, i^0)\ P(l^0|G)}{P(d^1)\ P(i^0)\ P(s^1|i^0) \sum_g P(g|d^1, i^0)\ P(l^0|g)}$$

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

$$= \boxed{\frac{1}{Z} \times P(G \mid d^1, i^0)\ P(l^0 \mid G)}$$

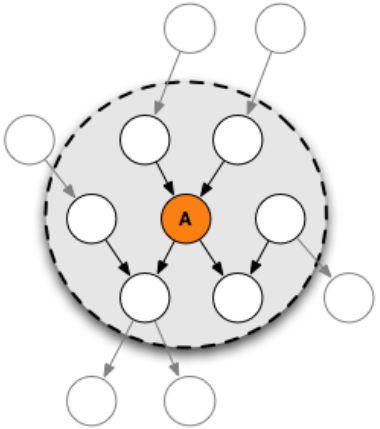
In Words:

- The re-sampling distribution depends only on G's parents and G's child
- Product of all CPDs in the network that involve G

Computing the Re-Sampling Distribution

Definition

The **Markov Blanket** $\mathrm{Mb}(A)$ of a variable A in a Bayesian network is the set of variables consisting of A's parents, its children, and its children's other parents.



Definition

The **Gibbs Resampling Distribution** for a variable Y_i, given the values of all other variables $\mathcal{X} - Y_i$ in a Bayesian network is defined as:

$$P(Y_i \mid \boldsymbol{x}_{-y_i}) = P(Y_i \mid \mathrm{mb}(Y_i))$$

Easy to show:

- $P(Y \mid \mathrm{mb}(Y))$ is proportional to the probability of Y given its parents, times the probability of each child given its respective parents:

$$P(Y \mid \mathrm{mb}(Y)) = \frac{1}{Z} \times P(Y \mid \mathrm{pa}(Y)) \times \prod_{C_i \in Children(Y)} P(c_i \mid \mathrm{pa}(C_i))$$

Estimating $P(X|e)$ [3]: Gibbs Sampling

Probabilistic Inference with Gibbs Sampling

Given:

- A Bayesian network $\mathcal{B}$ over $\mathcal{X}$
- A query $P(\boldsymbol{X} \mid \boldsymbol{e})$ with $\boldsymbol{X}, \boldsymbol{E} \subseteq \mathcal{X}$

Algorithm:

1. **Sample:** Probabilistically sample a set $\mathcal{D}_{\boldsymbol{e}}$ of M events from $\mathcal{B}$ via Gibbs Sampling[a]
2. Take $\mathcal{D}_{\boldsymbol{e}}$ as a direct approximation of the target distribution $P(\boldsymbol{Y} \mid \boldsymbol{e})$
3. **Estimate:** For a each value combination $\boldsymbol{X} = \boldsymbol{x}$, count the number K of samples $\in \mathcal{D}_{\boldsymbol{e}}$ that match $\boldsymbol{x}$. Estimate $P(\boldsymbol{x} \mid \boldsymbol{e})$ as

$$\hat{P}(\boldsymbol{x} \mid \boldsymbol{e}) = \frac{K}{|\mathcal{D}_{\boldsymbol{e}}|} = \frac{K}{M}$$

[a] Note: Due to (generally) low dimensionality of the variables' Markov Blankets, we can pre-compute and store all possible re-sampling distributions in tables.

Why Gibbs Sampling Works

Intuitive Explanation:

- Consider example from before:
  Query $P(I \mid \underline{s^1}, \underline{l^0})$. First sample $\boldsymbol{x}^{(0)} = (d^1, i^0, g^2, \underline{s^1}, \underline{l^0})$
- Create $\boldsymbol{x}^{(1)}$, first step: Re-sample G according to the resampling distribution

$$P(G \mid d^1, i^0, \underline{s^1}, \underline{l^0}) = \boxed{1/Z \times P(G \mid d^1, i^0) P(\underline{l^0} \mid G)}$$

  $\Rightarrow$ Sampling G now takes into account downstream evidence of G's child $\underline{L}$
  $\Rightarrow$ Distribution for new instance $\boldsymbol{x}^{(1)}$ is a bit closer to target distribution

- Next steps in creating $\boldsymbol{x}^{(1)}$:
  Re-sample I (using G, D and $\underline{S}$), then re-sample D (using G and I)

  $\Rightarrow$ Information from evidence nodes propagates through sampling process
  $\Rightarrow$ Sampling distribution becomes more and more similar to our target distribution $P(\boldsymbol{Y} \mid \boldsymbol{e})$ ...

Formal Explanation:

- Gibbs Sampling is an **MCMC** algorithm with **stationary distribution** $P(\boldsymbol{Y} \mid \boldsymbol{e})$

Markov Chain Monte Carlo (MCMC): Basic Concepts

Definition

A **Markov Chain** is a model of a discrete-time probabilistic process that

- at any time t is in exactly one of a set of possible states $\mathcal{S}$:

$$s^{(t)} \in \mathcal{S}, \quad \text{for } t = 0, 1, 2, \ldots$$

- fulfils the **Markov property:** the probability distribution $P(S^{(t+1)})$ over states at time $t+1$ depends only on the state $s^{(t)}$ at time t:

$$P(S^{(t+1)} \mid s^{(t)}, s^{(t-1)}, s^{(t-2)}, \ldots) = P(S^{(t+1)} \mid s^{(t)})$$

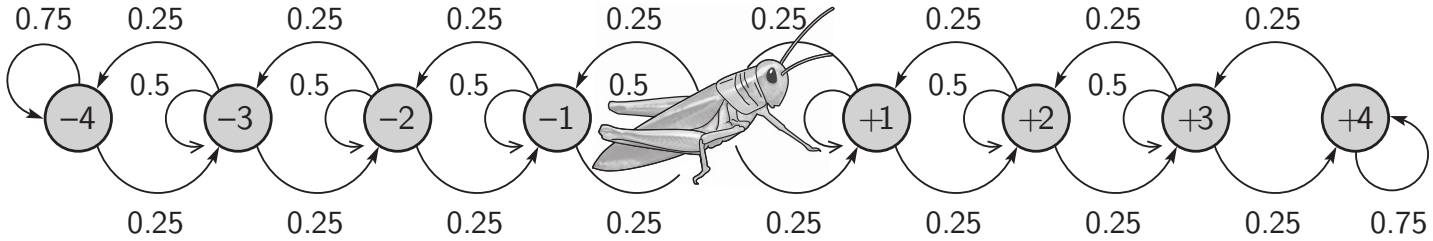
Formalisation

A Markov Chain model is formally defined by

- a **state space** $\mathcal{S} = \{s_1, \ldots, s_N\}$
- a **transition model** $\mathcal{T}$ that specifies, for each pair of states s, s', the probability of going from s to s'

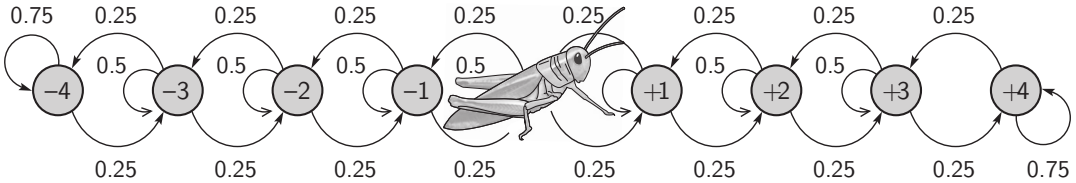
$$\mathcal{T}(s \to s') = P(S^{(t+1)} = s' \mid S^{(t)} = s)$$

An Example: The Grasshopper Markov Chain



- Consider a Markov chain with nine states: $\mathcal{S} =$ the integers $\{-4, \ldots, +4\}$
- A grasshopper starts out in position 0
- At each point in time, it
  - stays where it is with probability 0.5
  - jumps left or right with equal probability 0.25
- The two end positions are blocked: jumping left from -4 leads back to -4, jumping right from $+4$ leads to $+4$
- Transition Model = arrows + probabilities in figure
  (+ many zero probability transitions – not shown)

Long-term Behaviour: The Stationary Distribution



Consider grasshopper's location at time t as a random state variable $S^{(t)}$ with $Val(S^{(t)}) = \{-4, \ldots, +4\}$

- $t = 0$: Grasshopper starts at $0 \quad \Rightarrow P(S^{(0)}) = \{0, 0, 0, 0, 1, 0, 0, 0, 0\}$
- $t = 1$: $P(S^{(1)}) = \{0, 0, 0, .25, .5, .25, 0, 0, 0\}$
- $t = 2$: $P(S^{(2)}) = \{0, 0, .0625, .25, .375, .25, .0625, 0, 0\}$
- $\ldots$
- $t = 50$: distribution is almost uniform: $P(S^{(50)}) \approx \{\frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}\}$
- The distributions $P(S^{(t)})$ for this chain **converge to** the uniform distribution (and **stay there**)

☞ **The uniform distribution is called the Stationary Distribution for the Grasshopper Markov Chain.**

Gibbs Sampling as a Markov Process

Remember Gibbs Sampling:

- Start with an initial sample $\boldsymbol{x}^{(0)}$ from initial distribution $P^{(0)} = P_{\mathcal{B}_{\boldsymbol{E}=\boldsymbol{e}}}$
- Iterate: Generate new sample $\boldsymbol{x}^{(t+1)}$ from $\boldsymbol{x}^{(t)}$ by resampling each Y_i according to $P(Y_i \mid \boldsymbol{x}^{(t)}_{-y_i})$, for $Y_i \in \mathcal{X} - \boldsymbol{E}$

Gibbs Samples form a Markov Chain:

- States are the possible value assignments to the variables in the network (= all possible atomic events = all possible samples we could generate)

$$\mathcal{S} = Val(\mathcal{X})$$

- We start with one of these ($\boldsymbol{x}^{(0)}$)
- Probability of next sample $\boldsymbol{x}^{(t+1)}$ only depends on current sample $\boldsymbol{x}^{(t)}$
- Write a sample as $\boldsymbol{x} = (y_i, \boldsymbol{x}_{-y_i})$ (Y_i = variable to be resampled)
- Transition model = the *re-sampling distribution* (specific for each Y_i):

$$\mathcal{T}_i((y_i, \boldsymbol{x}_{-y_i}) \rightarrow (y_i', \boldsymbol{x}_{-y_i})) = P(y_i' \mid \boldsymbol{x}_{-y_i})$$

Gibbs Sampling as an MCMC Algorithm

In Words:

- In Gibbs Sampling, the sequence of samples is produced by a Markov chain where the states are the possible value assignments (samples, atomic events), and the transition model $\mathcal{T}$ is the Gibbs Resampling Distribution
- It can be proven[5] that the **stationary distribution** of this Markov chain is exactly the **posterior distribution** $P(\boldsymbol{Y} \mid \boldsymbol{e})$
- In other words: As we continue the sampling process for a long time, the samples come from distributions $P^{(t)}$ that converge to our target distribution!

- Gibbs Sampling is an instance of the general class of MCMC algorithms
- If we sample long enough before starting to use the samples, our samples will closely approximate our target distribution $P(\boldsymbol{Y} \mid \boldsymbol{e})$.

[5] Proof omitted for time reasons

Markov Chain Monte Carlo (MCMC) Methods

Given:

- A target distribution P^* we want to estimate (e.g., posterior $P(\boldsymbol{Y} \mid \boldsymbol{e})$)

General Idea of MCMC Algorithms:

- Construct a Markov Chain over the space of possible events (samples) whose sequence of state distributions $P^{(0)}, P^{(1)}, P^{(2)}, \ldots$ **converges to the target distribution** P^*
  (i.e., has P^* as its **stationary distribution**)
- Draw a sequence of samples $\mathcal{D} = \{\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t+1)}, \boldsymbol{x}^{(t+2)} \ldots\}$ from the chain
- Estimate target distribution by counting relative frequencies in $\mathcal{D}$

Central Problem:

- How to construct a **transition model (re-sampling strategy)** $\mathcal{T}$ that
- ... guarantees convergence to P^*
- ... is effectively computable and can be sampled from?

The Metropolis-Hastings Algorithm (briefly)

Metropolis-Hastings is a general algorithm for constructing Markov chains for *any desired Stationary Distribution*

Metropolis-Hastings: Sketch of the Basic Idea

- General method for constructing a Markov chain $\mathcal{T}^P$ with a particular stationary distribution P^*
- Unlike Gibbs Sampling, does *not* assume that we can sample from the transition model distribution $\mathcal{T}^P$ directly
- Rather, uses idea of a *proposal distribution* $\mathcal{T}^Q$ (as in Importance Sampling) as a substitute for $\mathcal{T}^P$
- Replace importance weights with *acceptance probabilities* $\mathcal{A}(\boldsymbol{x} \to \boldsymbol{x}')$: the probability that the next sample proposed by $\mathcal{T}^Q(\boldsymbol{x}^{(t)})$ will be accepted
- Central part of the algorithm: An analytic method for determining $\mathcal{A}(\boldsymbol{x} \to \boldsymbol{x}')$ such that the resulting Markov chain has exactly the desired stationary distribution P^*.

Relatively straightforward to formulate M-H for Bayesian Networks
(but not further discussed in this class)

The Metropolis-Hastings Algorithm (briefly)

A Bit of History

The Metropolis-Hastings algorithm originally comes from the field of statistical physics. It was developed by a group of theoretical physicists at the Los Alamos National Laboratory, New Mexico, then the center for classified military research in the U.S. One of the authors of the original publication is Edward Teller, colloquially known as the "father of the hydrogen bomb". The method was first derived in the 1950's to approximate the *Boltzmann Distribution* and later generalised by W.K. Hastings in 1970. See:

Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics* 21(6): 1087-1092.

Hastings, W.K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika* 57(1): 97-109.

Using MCMC: Some Practical Issues

Two (of many other) questions when applying MCMC in practice:

1. How long to wait until we start collecting samples from the chain? (Want to wait until sample distribution $P^{(t)}$ is close to P^*)
2. How to sample a set of *uncorrelated* samples from the chain?

Question 1: Burn-in Time

Question 1: Number of steps T until we collect a sample from the chain?

- **"Burn-in Time"**

Answer:

- Depends on how fast (we believe) the chain converges to P^*

Definition

Let $\mathcal{T}$ be a Markov chain; and
T_ϵ the minimal T such that, for any starting distribution $P^{(0)}$,

$$\mathbb{D}(P^{(T)}; P^*) \leq \epsilon$$

(where $\mathbb{D}$ is a measure of distance between distributions).
Then T_ϵ is called the **ϵ-Mixing Time** of $\mathcal{T}$.

Mixing Time

- can be extremely long
- cannot be calculated analytically
- is hard to analyse and estimate.

Question 2: How to Collect Samples

Question 2: How to sample a set of *uncorrelated* samples from the chain?

Problem:

- Consecutive samples will be strongly *correlated*
  $\Rightarrow$ cannot be used as *independent samples*
- Estimation based on these samples is **biased**.

Pragmatic Solution:

- After collecting a sample $\boldsymbol{x}^{(t)}$, wait for d steps, then collect next sample $\boldsymbol{x}^{(t+d)}$
- Alleviates problem slightly
- But: good d again depends on the *mixing time* of the chain ...

Consequence:

☞ In practice, simply wait with collecting samples as long as your time budget permits, and choose d as large as you can afford.

Advanced Topics (not treated here)

Reasoning with Collapsed Particles

- Problem: In high-dimensional spaces, need huge number of samples
- “Collapsed Particles”: model part of a variable assignment via particles, combined with closed-form representation of distribution over the rest.

Deterministic Search Methods for Approximate Inference

- Problem: Sampling can waste a lot of effort in low-probability parts of the space
- Deterministic Search Methods: Explicitly search for high-probability states (samples)

Optimisation Algorithms for Approximate Inference

- Entirely different approach: Compute approximate solutions incrementally, using search and message-passing algorithms
- Operate over graphs (clique trees; factor / cluster / region graphs, ...)
- Large class of methods; lots of literature.

What you should remember of this section

- Basic idea of approximate reasoning via sampling
- How to sample values from a discrete distribution
- How to sample events from an Bayesian Network
- How to use samples to estimate a (conditional) distribution: Rejection Sampling, Likelihood Weighting
- The general idea of Importance Sampling
- The general idea of MCMC algorithms
- Gibbs Sampling, and why it is an MCMC method
- That there is something called Metropolis-Hastings Algorithm (and what it is for)
- That there are some non-trivial issues related to actually using this in practice (mixing time, correlated samples)

Literature

Koller, Daphne and Friedman, Nir (2009).
Probabilistic Graphical Models: Principles and Techniques. Cambridge, MA: MIT Press.

Russell, Stuart J. and Norvig, Peter (2021).
Artificial Intelligence: A Modern Approach (Fourth Edition). Pearson Publishers.