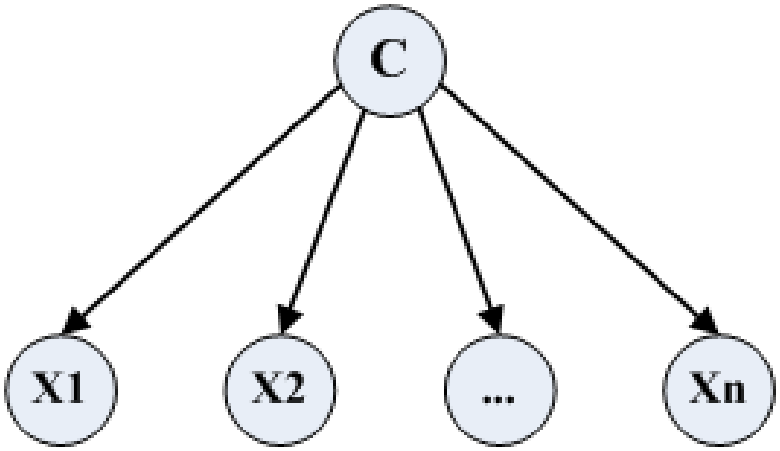
Jin Gu
Department of Automation, Tsinghua University
Email: jgu@tsinghua.edu.cn
Phone: (010) 62794294-866

Chapter 7 Particle-Based Approximate Inference

2021 Fall

Jin Gu (古槿)

Inference



$$P(C|X=x,\theta)$$

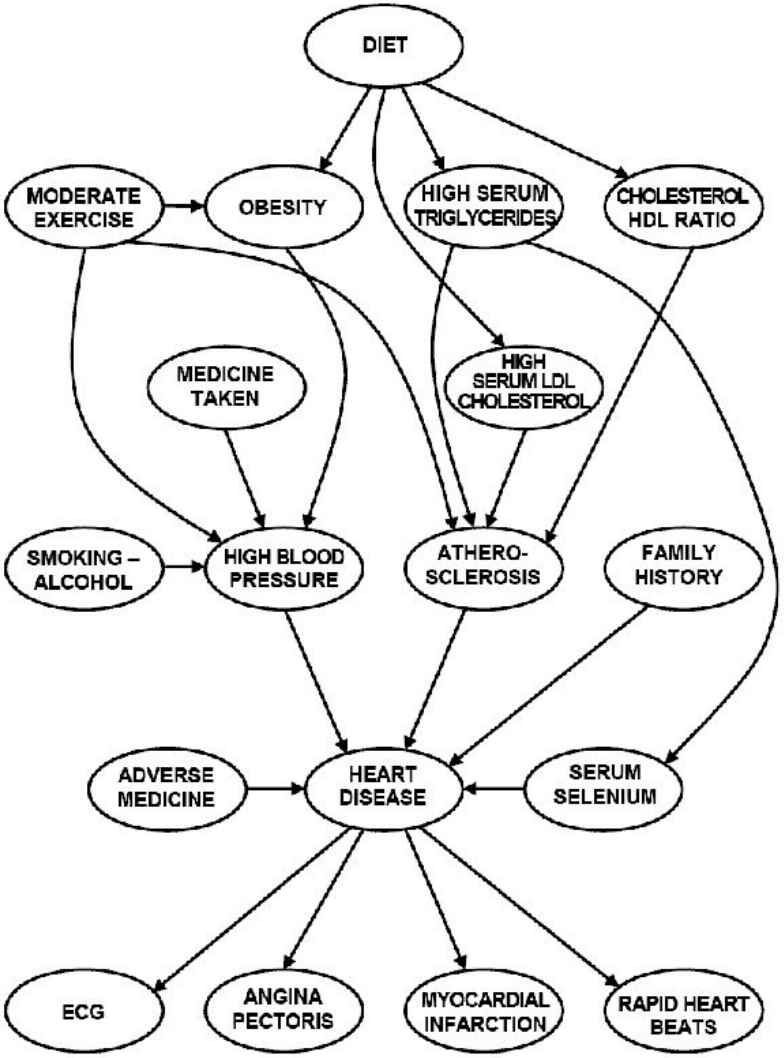
If the model is known, doctors can ***infer*** the possible disease C based on the observed symptoms $X=x$!!

Inferences in PGMs

- For a given probabilistic graphical model
  - $\{P, G/H\}$

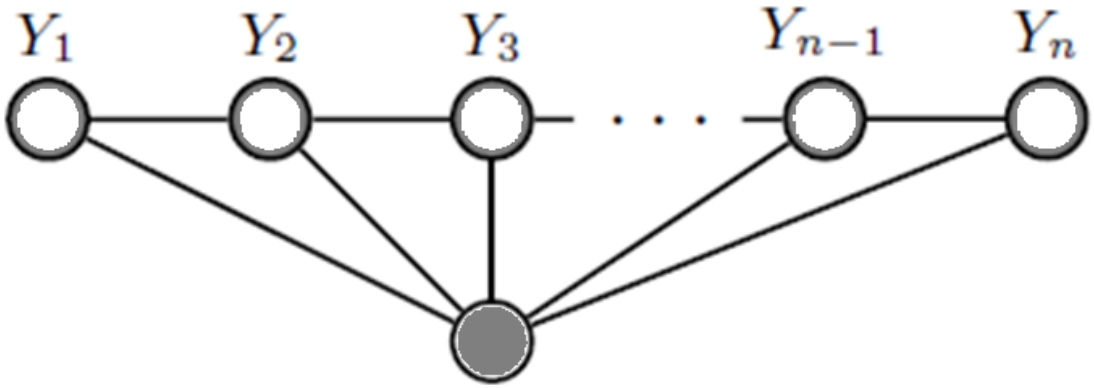
- *Inferences*: given a partial observations $\boldsymbol{E} = \boldsymbol{e}$, infer the target variables $\boldsymbol{Y}$
  - $\mathrm{P}(\boldsymbol{Y}|\boldsymbol{E} = \boldsymbol{e})$
  - Or $\arg\max_{\boldsymbol{y}} \mathrm{P}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{E} = \boldsymbol{e})$ (MAP inference)

Example: Disease Diagnosis



- Doctors or intelligent diagnostic systems need to ***infer*** the possibility of a target disease based on family history, daily lifestyle (causes) and blood test report, symptoms (results), etc.

Example: Gene Name Recognition



Gene Name?

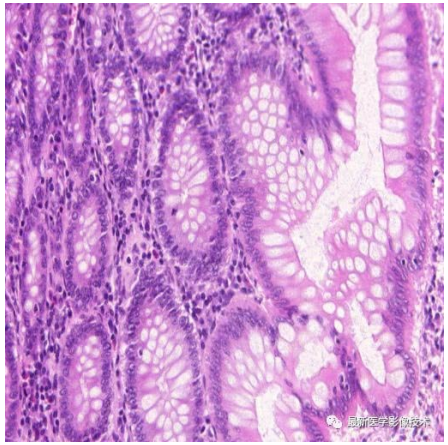
Observed "Texts"

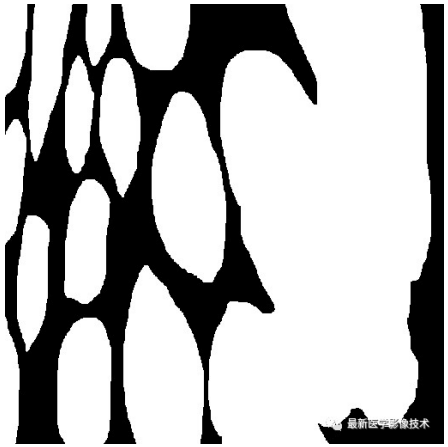
$$X = X_1, \ldots, X_{n-1}, X_n$$

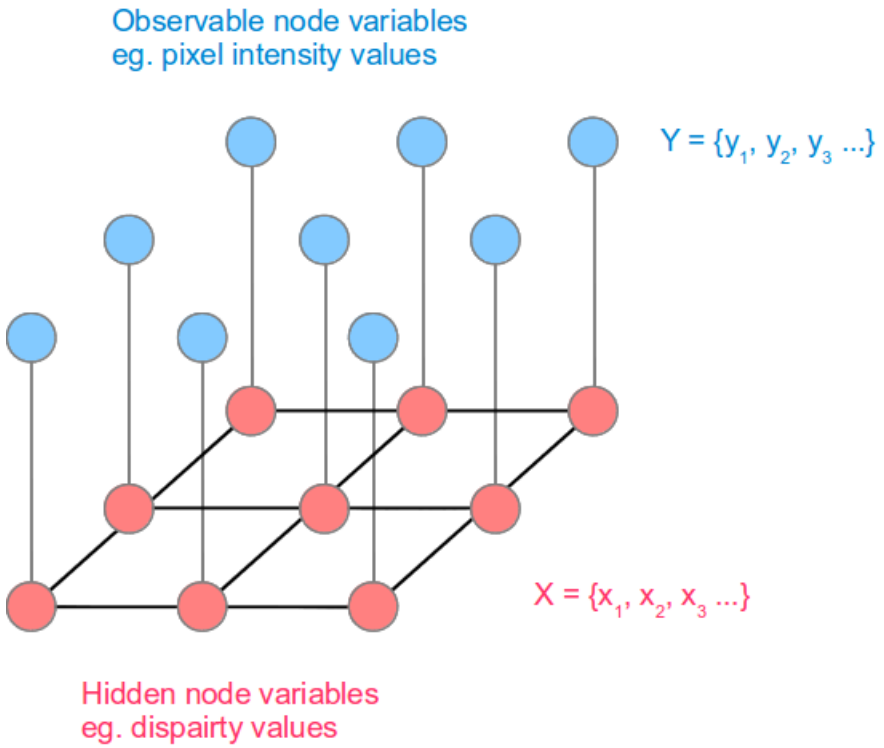
Diversion of aspartate in ASS1-deficient tumours fosters de novo pyrimidine synthesis.
Cancer cells hijack and remodel existing metabolic pathways for their benefit. Argininosuccinate synthase (ASS1) is a urea cycle enzyme that is essential in the conversion of nitrogen from ammonia and aspartate to urea. A decrease in nitrogen flux through ASS1 in the liver causes the urea cycle disorder citrullinaemia. In contrast to the well-studied consequences of loss of ASS1 activity on ureagenesis, the purpose of its somatic silencing in multiple cancers is largely unknown. Here we show that decreased activity of ASS1 in cancers supports proliferation by facilitating pyrimidine synthesis via CAD (carbamoyl-phosphate synthase 2, aspartate transcarbamylase, and dihydroorotase complex) activation. Our studies were initiated by

- ***Infer*** whether a word or a phrase is a gene name

Example: Image Segmentation

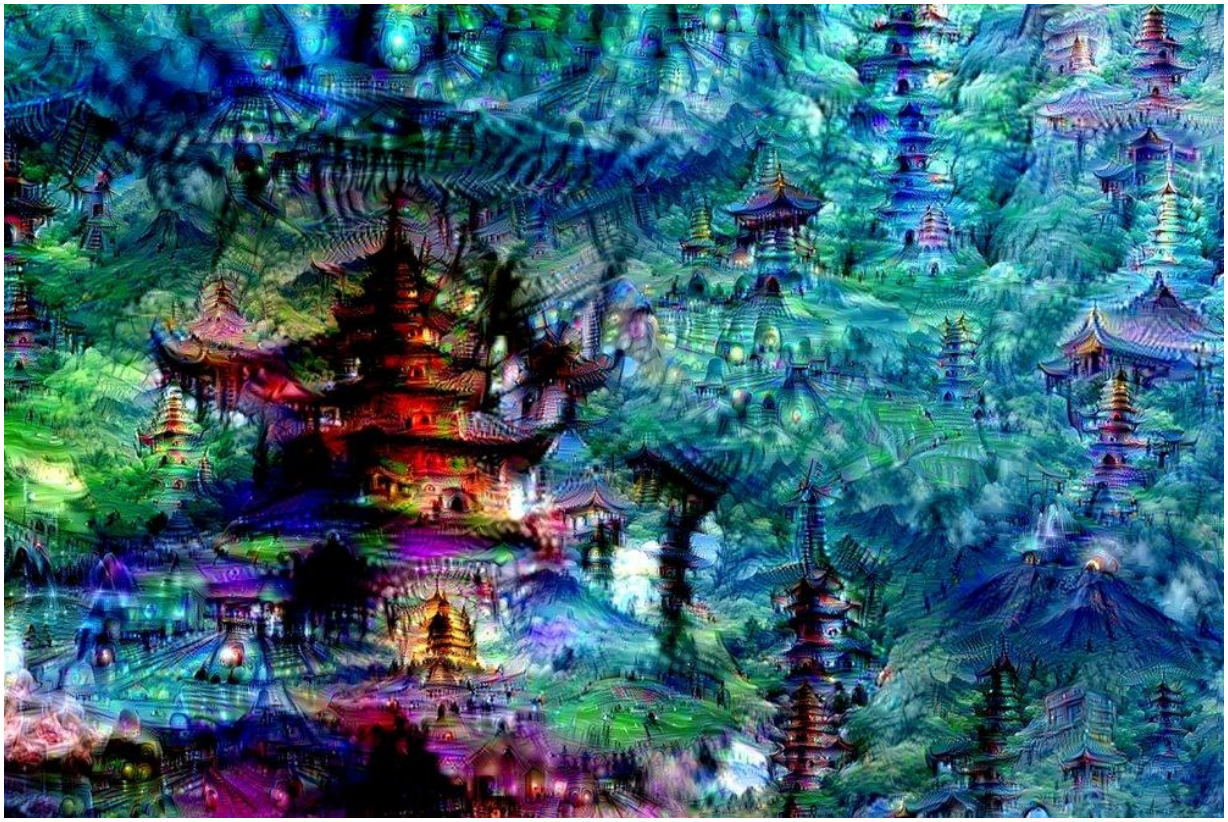






Segment image into different regions: ***infer*** the posteriors of the hidden labels based on observed pixels.

Example: Image / Voice Synthesis



- Generate an image with target style (pre-learned deep texture networks) according to your input image: ***infer*** the posterior colors for each pixel according to your inputs.

Chapter 7 Particle-based (or *Monte Carlo*) approximate inferences

Textbook1

Chapter 12.1	Forward sampling
Chapter 12.2	Likelihood weighting and importance sampling
Chapter 12.3.1-12.3.3	Gibbs sampling
Chapter 12.3.4.2	Metropolis-Hastings algorithm
***Chapter 12.5**	Learning how to use MC for efficient inferences

Textbook2

Chapter 23	Monte Carlo inferences

Other references

[Book] Liu J. **Monte Carlo Strategies in Scientific Computing**. *Springer Series in Statistics* 2001.

Radford MN (2011). **MCMC Using Hamiltonian Dynamics**. In Steve Brooks; Andrew Gelman; Galin L. Jones; Xiao-Li Meng. **Handbook of Markov Chain Monte Carlo**. Chapman and Hall/CRC.

Lindsten F, Helske J, Vihola M. **Graphical model inference: Sequential Monte Carlo meets deterministic approximations**. 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)

Inferences in PGMs

- For a given probabilistic graphical model
  - $\{P, G/H\}$

- *Inferences*: given a partial observations $\boldsymbol{E} = \boldsymbol{e}$, infer the target variables $\boldsymbol{Y}$
  - $\mathrm{P}(\boldsymbol{Y}|\boldsymbol{E} = \boldsymbol{e})$
  - Or $\arg\max_{\boldsymbol{y}} \mathrm{P}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{E} = \boldsymbol{e})$ (MAP inference)

Challenge: Inference Complexity

- **NP-Hard** if do inference directly from the joint probability P (variable elimination)

The curse of dimensionality

- Hopeless?
  - No, we will see *many network structures* that have provably efficient algorithms and we will see cases when *approximate inference works efficiently with high accuracy*

Particle-Based Approximate Inferences

General principle (χ denotes all of the network variables. $Y \subseteq \chi$)

- Generate samples $\xi[1], ..., \xi[M]$ from the distribution $P(\chi)$
- Estimate function by $E_P(f(\chi)) \approx \frac{1}{M}\sum_{m=1}^{M} f(\xi[m])$

- Why does this strategy work?
  - Recall the definition of probability

$$P(x) \approx \frac{n_x}{n_t}.$$

- *Key* for particle-based approximate inferences?
  - How to get samples from posteriors P(Y|E=e)

Outlines

- Forward Sampling
- Likelihood Weighting and Importance Sampling
  - Likelihood Weighting
  - Unnormalized Importance Sampling
  - Normalized Importance Sampling
  - Importance Sampling for Bayesian Networks
- Markov Chain Monte Carlo (MCMC) Methods
  - Markov Chain and Stationary Distribution
  - Gibbs Sampling
  - Metropolis-Hastings Algorithm
  - Hamiltonian Monte Carlo

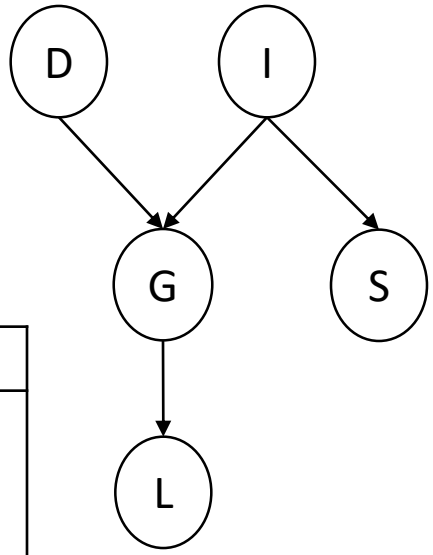
Forward Sampling

- Generate random samples from $\mathrm{P}(\chi)$
  - Use the Bayesian network to generate samples
- Estimate probability by
  - $\mathrm{P}(\chi = \boldsymbol{e}) \approx \frac{1}{M}\sum_{m=1}^{M} \mathrm{I}(\xi[m] = \boldsymbol{e})$
    - Here $\mathrm{I}(\cdot)$ is the indicator function

The probabilities can be ***approximated*** based on the histograms of the generated samples

Forward Sampling

d^0	d^1
0.4	0.6



i^0	i^1
0.7	0.3

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

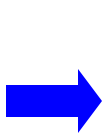
D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	0.2

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

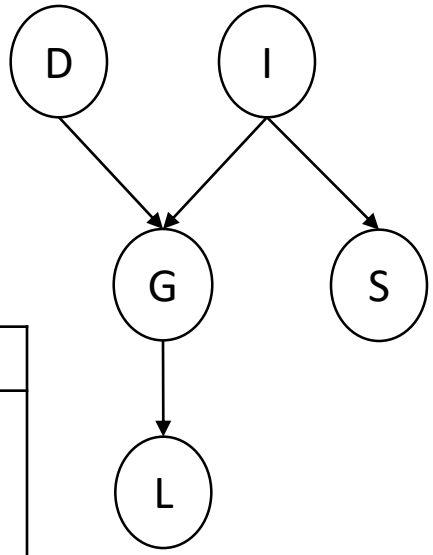
Samples

D	***I***	***G***	***S***	***L***

Forward Sampling



d^0	d^1
0.4	**0.6**



i^0	i^1
0.7	0.3

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	0.2

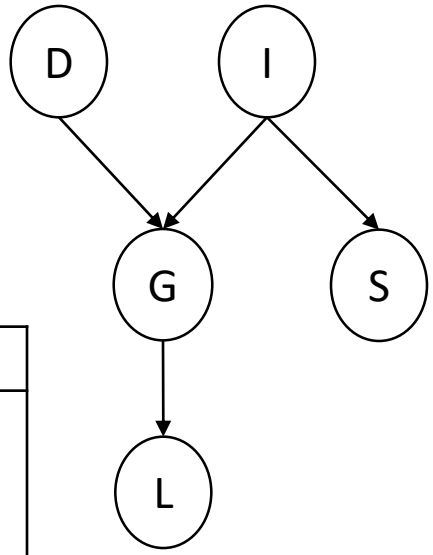
G	l^0	l^1
g^0	0.1	0.9
g^1	0.4	0.6
g^2	0.99	0.01

Samples

D	***I***	***G***	***S***	***L***
d^1				

Forward Sampling

d^0	d^1
0.4	0.6



i^0	i^1
0.7	**0.3**

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	0.2

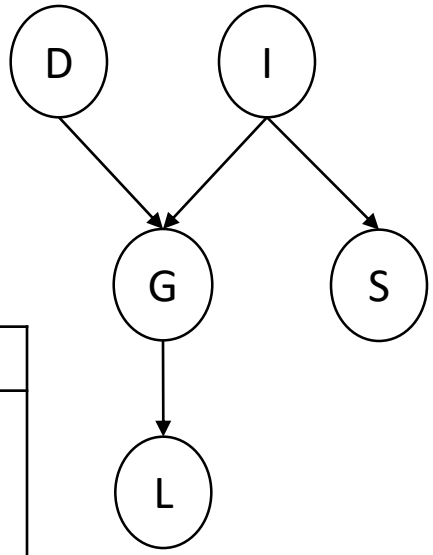
G	l^0	l^1
g^0	0.1	0.9
g^1	0.4	0.6
g^2	0.99	0.01

Samples

D	***I***	***G***	***S***	***L***
d^1	**i^1**			

Forward Sampling

d^0	d^1
0.4	0.6



i^0	i^1
0.7	0.3

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	**0.2**

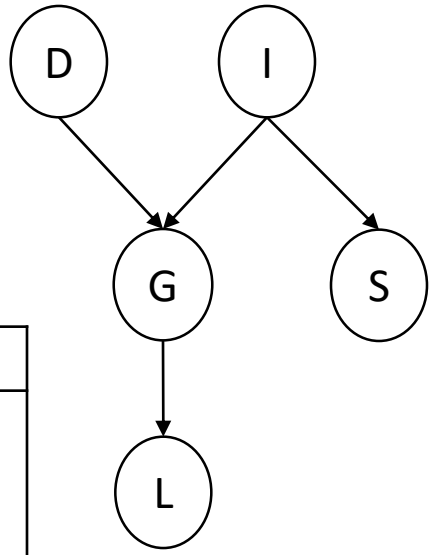
G	l^0	l^1
g^0	0.1	0.9
g^1	0.4	0.6
g^2	0.99	0.01

Samples

D	***I***	***G***	***S***	***L***
d^1	i^1	**g^2**		

Forward Sampling

d^0	d^1
0.4	0.6



i^0	i^1
0.7	0.3

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	0.2

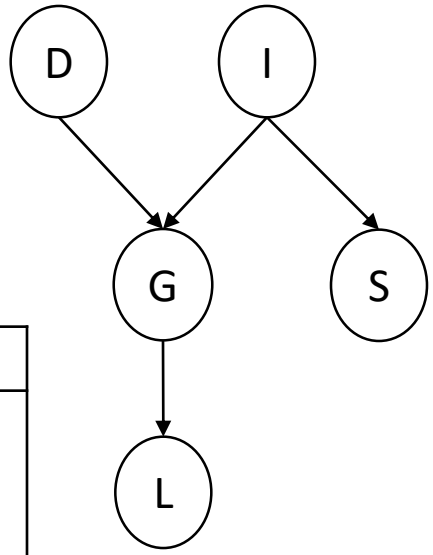
G	l^0	l^1
g^0	0.1	0.9
g^1	0.4	0.6
g^2	0.99	0.01

Samples

D	***I***	***G***	***S***	***L***
d^1	i^1	g^2	**s^1**	

Forward Sampling

d^0	d^1
0.4	0.6



i^0	i^1
0.7	0.3

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	0.2

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

Samples

D	***I***	***G***	***S***	***L***
d^1	i^1	g^2	s^1	l^0

Forward Sampling

Let $X_1, \cdots, X_n$ be a topological order of χ
for $i = 1, \cdots, n$
    Sample x_i from $P(X_i | Pa_{X_i})$
    // since $Pa_{X_i} \subseteq \{X_1, \cdots, X_{i-1}\}$, we already assigned values to them
return $(x_1, \cdots, x_n)$

- Estimate probability: $\mathrm{P}(\chi = e) \approx \frac{1}{M} \sum_{m=1}^{M} \mathrm{I}\{\xi[m] = e\}$
- Estimate any expectation: $E_P(f) \approx \frac{1}{M} \sum_{m=1}^{M} f(\xi[m])$

How to deal with only a subset of variables Y?

- Estimate probability: $\mathrm{P}(y) \approx \frac{1}{M} \sum_{m=1}^{M} \mathrm{I}\{\xi[m]\langle Y \rangle = y\}$

Forward Sampling

- Sampling cost
  - Per variable cost: $O(\log(Val|X_i))$
    - Sample uniformly in [0,1]
    - Find appropriate value of all $Val|X_i$ values that X_i can take
  - Per sample cost: $O(n\log(d)), d = \max_i Val|X_i$
  - **Total cost for *m* samples**: $O(Mn\log(d))$
- Number of samples needed
  - To get a relative error < ε, with probability 1-δ, we need

$$M \geq 3\frac{\ln(2/\delta)}{P(y)\varepsilon^2}$$

    - Note that number of samples grows inversely with P(y)
    - For small P(y) we need many samples, otherwise we report P(y)=0

Forward Sampling with Rejection

- In general we want to compute P(***Y***|***E=e***)
- We can do so with *sampling with rejection*
  - Generate samples as in forward sampling
  - **Reject** or discard the samples in which E≠e
  - Estimate function from accepted samples

- Problem: if evidence is unlikely or with very low probability (e.g., P(*e*)=0.001)) then we generate many rejected samples

Likelihood Weighting

- Can we ensure that all samples satisfy $E=e$?
  - Proposal: when sampling a variable $X \in E$, set $X=e$
    - Problem: we are trying to sample from the posterior $P(X|e)$ but actually we get samples from the prior $P(X)$
    - Can we sample from $P'(X,e)$ but then normalize them?
    - KEY: can we design a normalization algorithm which ensures that the normalized samples follow the target distribution $P(X|e)$?

Possible solution: *sampling* according to *the prior* distribution, but *weighting* samples with *the likelihood*!!

Likelihood Weighting: Details

- You can still generate samples based on $P(\chi)$ as forward sampling while directly set $\boldsymbol{E}=\boldsymbol{e}$

- But our aim is to generate samples from $P(\boldsymbol{Y}|\boldsymbol{e})$, which is proportional to $P(\boldsymbol{Y},\boldsymbol{e})$

- *What are the differences?*
  - The evidences will affect the posterior probability

Likelihood Weighting: Details

- If no evidence, based no BN factorization theorem, we can easily write down
  - $\mathrm{P}(\boldsymbol{X}) = \prod_i \mathrm{P}\left(X_i | \mathrm{Pa}(X_i)\right)$
- The evidences will only affect the local factors
  - $\mathrm{P}\left(E_j = e_j | \mathrm{Pa}(E_j)\right)$ for each given variable E_j
- So, if you generate a sample from the prior $\mathrm{P}(\boldsymbol{X})$ (directly set set $\boldsymbol{E}=\boldsymbol{e}$)
- It should be weighted by $\prod_j \mathrm{P}\left(E_j = e_j | \mathrm{Pa}(E_j)\right)$

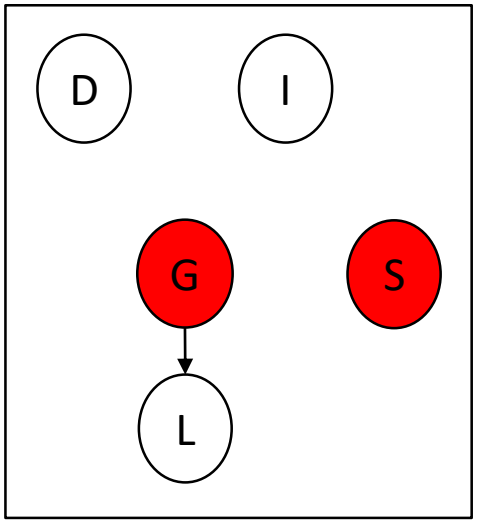
Recall: Forward Sampling with Rejection

- Sampling based on the prior P(χ), but reject the samples if $E \neq e$
- Or we can see that a sample has the probability $\prod_j \mathrm{P}\left(E_j = e_j | \mathrm{Pa}\left(E_j\right)\right)$ to be accepted

- So the samples generated by LW are equal to FS with rejection by weighting, but we can reduce the computational cost wasted by Rejection

Likelihood Weighting

Let $X_1, \cdots, X_n$ be a topological order of χ
$\omega \leftarrow 1$
for $i = 1, \cdots, n$
  if $X_i \notin \boldsymbol{E}$
    Sample x_i from $P(X_i | Pa_{X_i})$
  if $X_i \in \boldsymbol{E}$
    $x_i \leftarrow \boldsymbol{E}\langle X_i \rangle$. // Assignment to X_i in $\boldsymbol{E}$
    $\omega \leftarrow \omega \cdot P(x_i | Pa_{X_i})$
return $(x_1, \cdots, x_n)$, ω

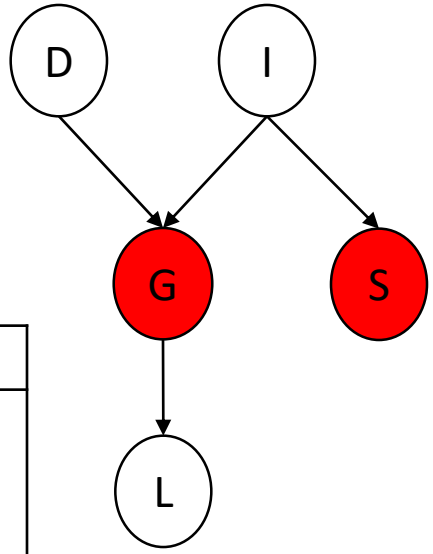


- Estimate P($\boldsymbol{Y}$=$\boldsymbol{y}$|$\boldsymbol{e}$) by:

$$P(\boldsymbol{y}|\boldsymbol{e}) \approx \frac{\sum_{m=1}^{M} \omega[m] \cdot \mathbf{I}\{\xi[m]\langle Y \rangle = y\}}{\sum_{m=1}^{M} \omega[m]}$$

Likelihood Weighting

d^0	d^1
0.4	0.6



i^0	i^1
0.7	0.3

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	0.2

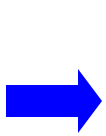
G	l^0	l^1
g^0	0.1	0.9
g^1	0.4	0.6
g^2	0.99	0.01

Samples

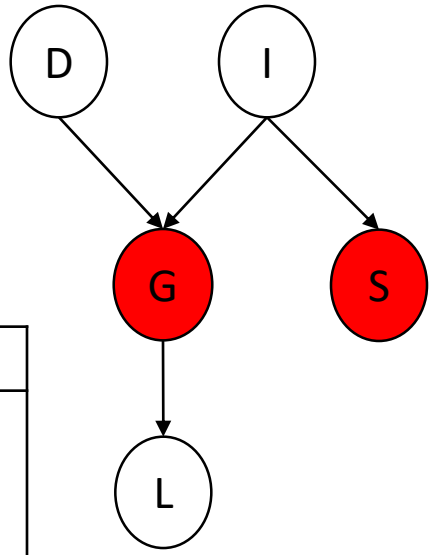
D	***I***	***G***	***S***	***L***	***W***

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

Likelihood Weighting



d^0	d^1
0.4	**0.6**



i^0	i^1
0.7	0.3

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	0.2

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

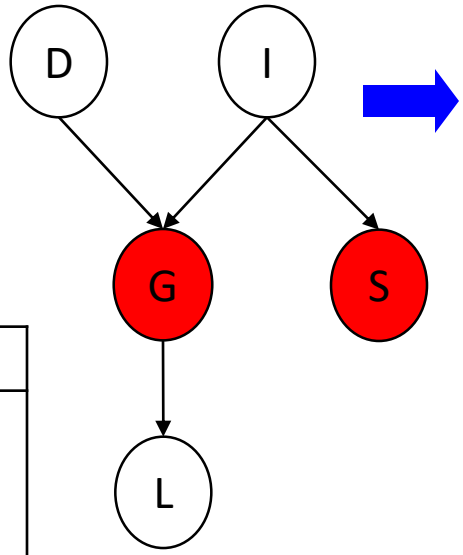
Samples

D	***I***	***G***	***S***	***L***	***W***
d^1					1

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

Likelihood Weighting

d^0	d^1
0.4	0.6



i^0	i^1
0.7	**0.3**

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	0.2

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

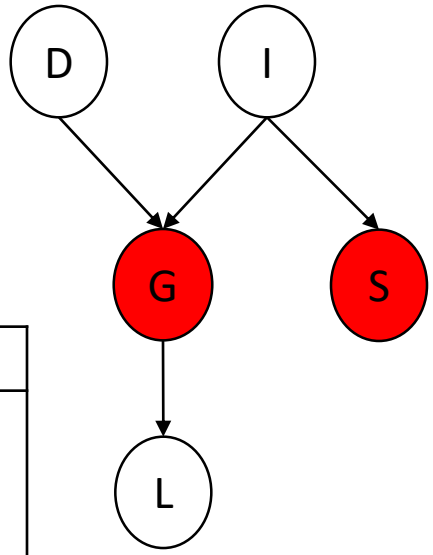
Samples

D	***I***	***G***	***S***	***L***	***W***
d^1	**i^1**				1

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

Likelihood Weighting

d^0	d^1
0.4	0.6



i^0	i^1
0.7	0.3

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	**0.2**

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

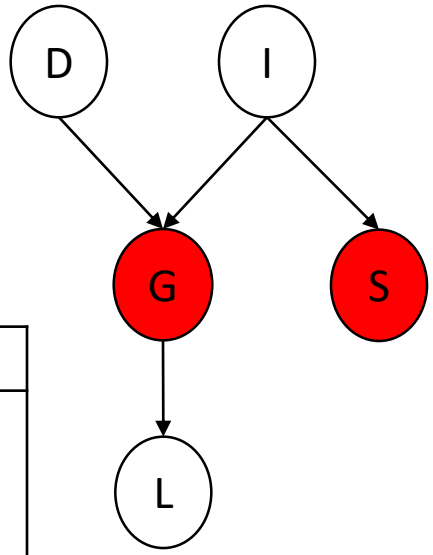
Samples

D	***I***	***G***	***S***	***L***	***W***
d^1	i^1	g^2			0.2

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

Likelihood Weighting

d^0	d^1
0.4	0.6



i^0	i^1
0.7	0.3

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	0.2

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

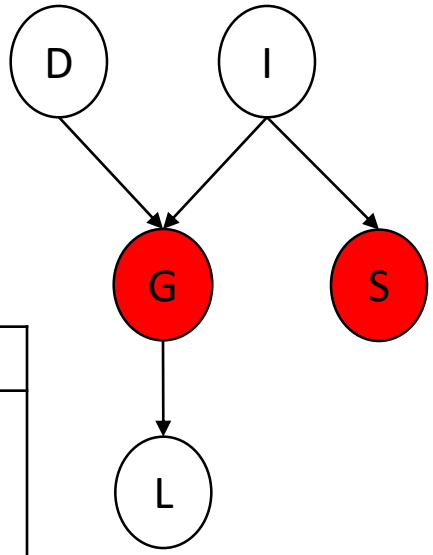
Samples

D	***I***	***G***	***S***	***L***	***W***
d^1	i^1	g^2	**s^1**		0.16

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

Likelihood Weighting

d^0	d^1
0.4	0.6



i^0	i^1
0.7	0.3

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

D	***I***	g^0	g^1	g^2
d^0	i^0	0.3	0.4	0.3
d^0	i^1	0.05	0.25	0.7
d^1	i^0	0.9	0.08	0.02
d^1	i^1	0.5	0.3	0.2

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

Sampling from the *prior*, but ***weighting*** with the *likelihood*

Samples

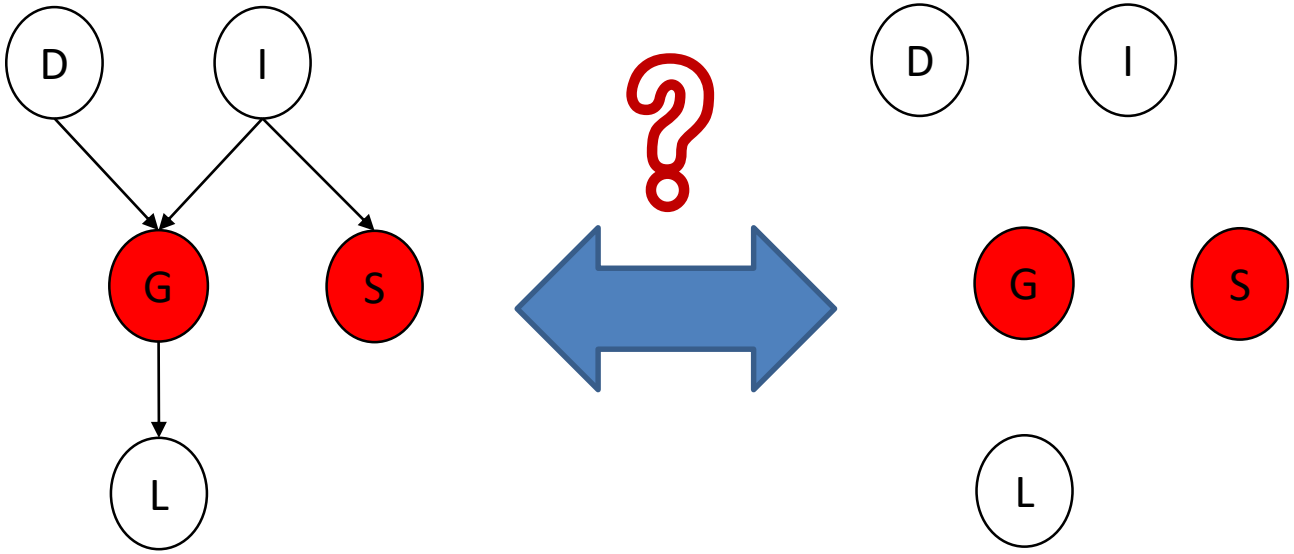
D	***I***	***G***	***S***	***L***	***W***
d^1	i^1	g^2	s^1	l^0	0.16

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

Importance Sampling

- Proposal: to estimate a function relative to P, rather than sampling from the distribution P, sample from another distribution Q
  - P is called the **target** distribution
  - Q is called the **proposal** or **sampling** distribution
  - Requirement from Q: $P(x) > 0 \rightarrow Q(x) > 0$
    - Q does not 'ignore' any non-zero probability events in P
    - In practice, performance depends on similarity between Q and P

Toy Example: Why Importance Sampling?



Directly sampling from *complete independent structure* is **much easier** than any *other structure*

Unnormalized Importance Sampling

- Since:

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

- We can estimate the expectation of $f(x)$ relative to P by generating samples from Q:

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

- Estimator variance decreases with more samples
- Q=P is the lowest variance estimator

Normalized Importance Sampling

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

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

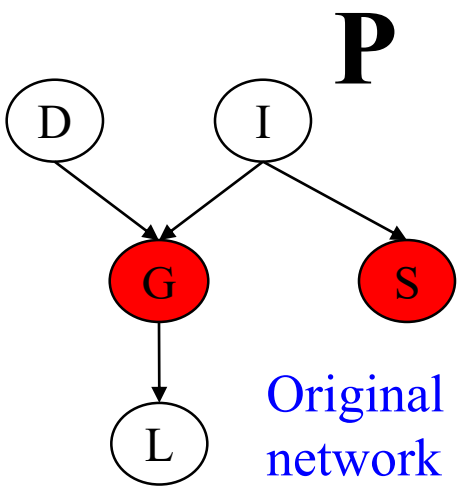
Normalized Importance Sampling

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

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

Importance Sampling for BNs

- Define *mutilated network* $G_{E=e}$ as:
  - Nodes $X \in E$ have no parents in $G_{E=e}$
  - Nodes $X \in E$ have CPD that is 1 for X=E[X] and 0 otherwise
  - The parents and CPDs for all other variables are unchanged

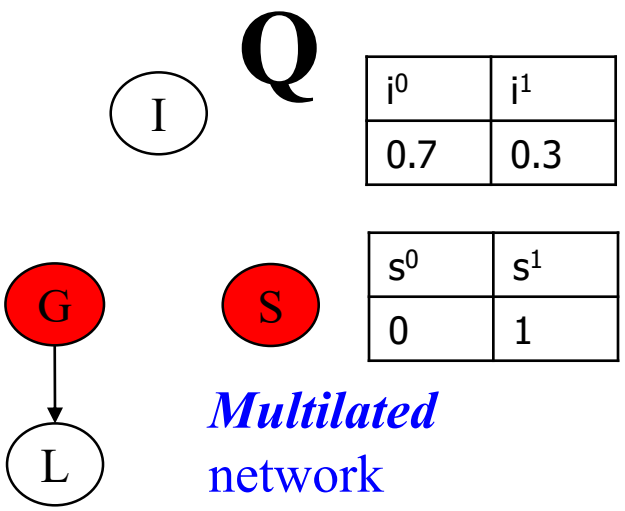


Original network

d^0	d^1
0.4	0.6

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

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



i^0	i^1
0.7	0.3

s^0	s^1
0	1

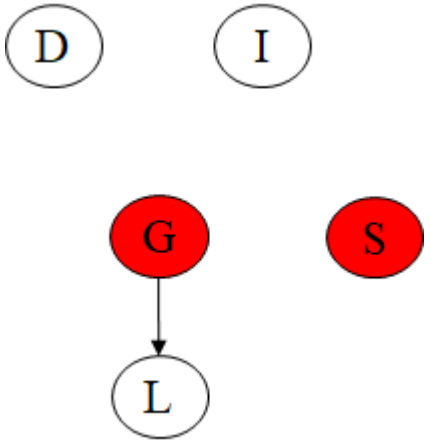
Multilated network

For given *G*, the posterior probability of *D*, *I* should be changed. But here we can directly use the prior of *D*, *I*.

G	I^0	I^1
g^0	0.1	0.9
g^1	0.4	0.6
g^2	0.99	0.01

Likelihood Weighting as Importance Sampling

- Target distribution $\mathrm{P}(\boldsymbol{X}|\boldsymbol{e}) \propto \mathrm{P}(\boldsymbol{X}, \boldsymbol{e})$
- If we set the proposal distribution based on the *multilated network*
  - $\mathrm{Q}(\boldsymbol{X}) = P_{G_{\boldsymbol{E}=\boldsymbol{e}}}(X)$



Multilated network: break the links of evidence nodes and their parents

- Likelihood weighting is precisely normalized importance sampling using the above proposal distribution (as *multilated network*)

$$P(y \mid e) \approx \frac{\sum_{m=1}^{M} w[m]\mathbf{1}\{x[m](y) = y\}}{\sum_{m=1}^{M} w[m]}$$

- Likelihood weighting
  - $w[m] = \prod_{x_i \in E} P\left(x_i = e_i | Pa_{X_i}\right)$
- Importance sampling

$$P(y \mid e) \approx \frac{\sum_{m=1}^{M} \mathbf{1}\{x[m](y) = y\} P'(x[m]) / Q(x[m])}{\sum_{m=1}^{M} P'(x[m]) / Q(x[m])}$$

$$\frac{P'(x[m])}{Q(x[m])} = \frac{\prod_{x_i \in E} P\left(x_i = e_i | Pa_{X_i}\right) \prod_{x_i \in X-E} P\left(x_i = x_i[m] | Pa_{X_i}\right)}{\prod_{x_i \in X-E} P\left(x_i = x_i[m] | Pa_{X_i}\right)}$$

$$= \prod_{x_i \in E} P\left(x_i = e_i | Pa_{X_i}\right) = w[m]$$

Number of Samples Needed

- The required number of samples is dependent on the similarity between Q and P

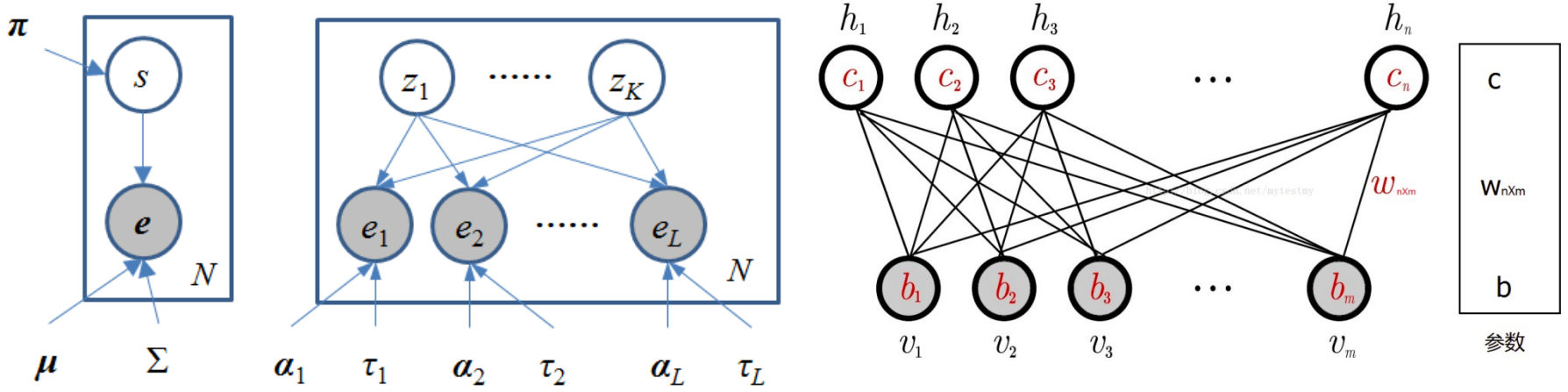
- The effective sample size

$$M_{\text{eff}} = \frac{M}{1 + \mathbb{V}ar[\mathcal{D}]}$$

$$\mathbb{V}ar[\mathcal{D}] = \sum_{m=1}^{M} w(\boldsymbol{x}[m])^2 - (\sum_{m=1}^{M} w(\boldsymbol{x}[m]))^2$$

$$\mathcal{D} = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\} \text{ from } Q$$

LW & IS on GMMs, Latent Models and Markov Networks



Try to work out the algorithms by yourself!

Limitations of LW & IS

- Likelihood weighting is inefficient for Markov networks: need to transform MN to BN (add many additional links)

- Importance sampling is hard to choose a good sampling probability Q for complex structures (*the convergence is too slow if Q is very dissimilar to target distribution P*)

Lessons from Physics

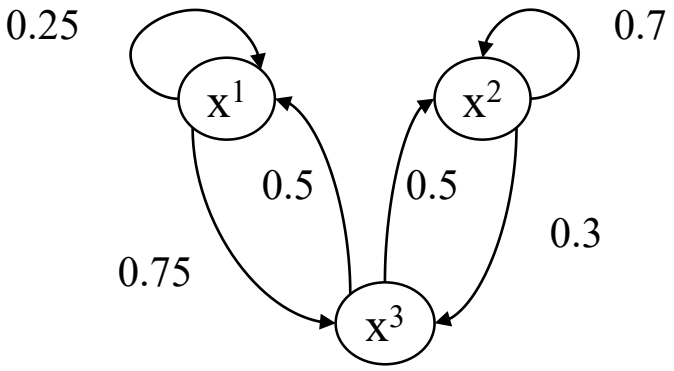
- Boltzmann Distribution
  - The current state of a molecule is probabilistically determined by its previous state
  - For an ***isolated regular*** system in ***steady*** state, the probability of a particle in a given state follows Boltzmann (Gibbs) Distribution
  - $P(X = s) \propto e^{-\frac{E_S}{kT}}$
- Can we design a novel sampling strategy?

Lessons from Markov Chains

- A **Markov chain** consists of
  - A state space Val(X)
  - Transition probability T($x \rightarrow x'$) of going from state x to x'

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

- Distribution over subsequent states:



Simple Markov chain

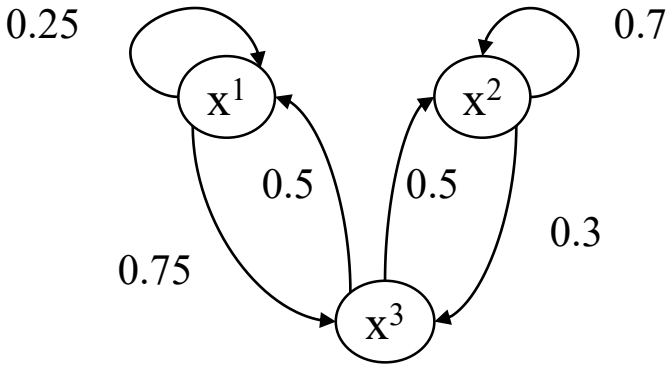
Stationary Distribution

- A distribution $\pi(X)$ is a **stationary distribution** for a Markov chain T if it satisfies

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

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

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



Simple Markov chain

Markov Chains & Stationary Dist.

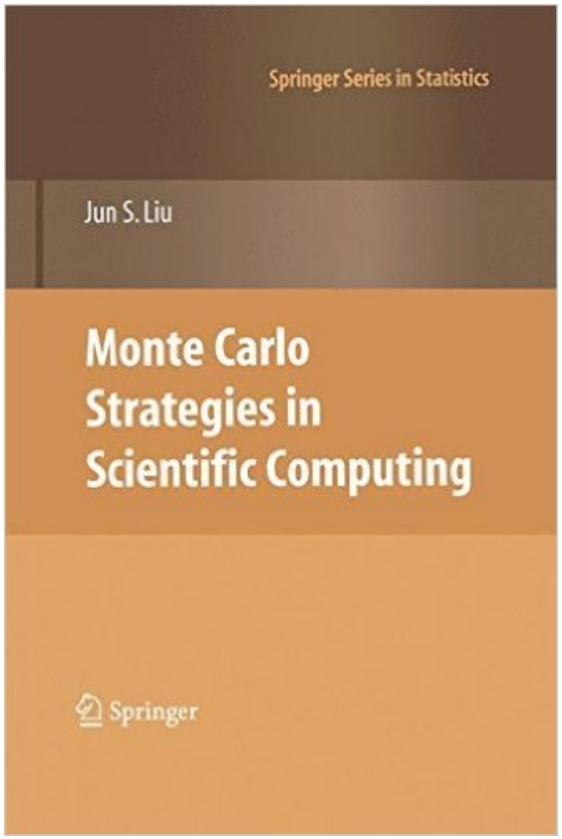
- A Markov chain is ***regular*** if there is k such that for every $x,x' \in \text{Val}(X)$, the probability of getting from x to x' in exactly k steps is greater than zero
- Theorem: A finite state Markov chain T has a **unique stationary distribution** if and only if **it is regular**

Proposal: design a Markov chain whose stationary distribution is $\text{P}(\boldsymbol{X}|\boldsymbol{e})$

Markov Chain Monte Carlo

- Generate current particle based on the previous one according to the posterior probability

- Requirements
  - ***Regular***: all the states are reachable in finite steps from any other state (all the nodes are linked in a single graph & all the local factors > 0)



Proposal for Gibbs Sampling

- **States**
  - Legal assignments to variables $X = (X_i)_{i=1\sim n}$ (consistent with evidence)
- **Transition probability**
  - $T\left(\boldsymbol{X}^{(t)} = \boldsymbol{x} \to \boldsymbol{X}^{(t+1)} = \boldsymbol{x}'\right) = P\left(\boldsymbol{X}^{(t+1)} | \boldsymbol{X}^{(t)}, \boldsymbol{e}\right)$
    - $T(\boldsymbol{x} \to \boldsymbol{x}')$ means transition probability between two states (two legal assignments to $\boldsymbol{X}$)
    - $P\left(\boldsymbol{X}^{(t+1)} | \boldsymbol{X}^{(t)}, \boldsymbol{e}\right)$ is directly defined on $P(\boldsymbol{X}|\boldsymbol{e})$

- Let's prove: P($\boldsymbol{X}|\boldsymbol{e}$) is a stationary distribution to the above chain

Proposal for Gibbs Sampling

- What does the transition probability mean?
  - $T\left(\boldsymbol{X}^{(t)} = \boldsymbol{x} \rightarrow \boldsymbol{X}^{(t+1)} = \boldsymbol{x}'\right) = P\left(\boldsymbol{X}^{(t+1)} | \boldsymbol{X}^{(t)}, \boldsymbol{e}\right)$
- Construction a Markov chain as follow
  - Given an order of updating (the order can be changed each run), update the variables one by one according to its Markov blanket (*Note: excluding itself*)
  - $P\left(X_i^{(t+1)} | X_1^{(t+1)}, \cdots, X_{i-1}^{(t+1)}, X_{i+1}^{(t)}, \cdots, X_n^{(t)}\right) =$
    $P\left(X_i^{(t+1)} | \boldsymbol{MB}\left(X_i^{(T)}\right), t+1 \; for < i \; and \; t \; for > i\right)$

Stationary Dist. of Gibbs Sampling

- According to the definition of stationary distribution of Markov chains (*regular* process)
  - $\pi(X = x) = \sum_{x'} \pi(X = x')T(x' \to x)$
    - $\pi(X = x)$: stationary distribution
    - $T(x' \to x)$: transition probability

Stationary distribution:
the probability of state X=x in time t is equal to its probability in time t+1

Stationary Dist. of Gibbs Sampling

- For GS, transition probability is set as
  - $T(x' \to x)$: $\mathrm{P}(X = x | X = x')$
- We can easily show that
  - $\pi(X = x) = \mathrm{P}(X = x)$ is a solution of
  - $\pi(X = x) = \sum_{x'} \pi(X = x') T(x' \to x)$
  - Because: $\mathbf{P}(\boldsymbol{x}) = \sum_{\boldsymbol{x}'} \mathbf{P}(\boldsymbol{x}') \mathbf{P}(\boldsymbol{x} | \boldsymbol{x}')$

Transition probability is set as the form of **target distribution**

Q: A unique solution?

Yes if **P is regular**.

Gibbs Sampling: Algorithm

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

- Sampling procedure (one sample per run)
  - Set $\boldsymbol{x}[m]=\boldsymbol{x}[m\text{-}1]$ & a variable updating order
  - For each variable $X_i \in \boldsymbol{X}\text{-}\boldsymbol{E}$
    - Set $\boldsymbol{u}_i = \boldsymbol{x}[\mathrm{m}](\boldsymbol{X}\text{-}X_i)+\boldsymbol{e}$
    - Sample from $\mathrm{P}(X_i|\boldsymbol{u}_i)$
    - Save sample $\boldsymbol{x}[m](X_i)$ = sampled value
  - Return $\boldsymbol{x}[m]$

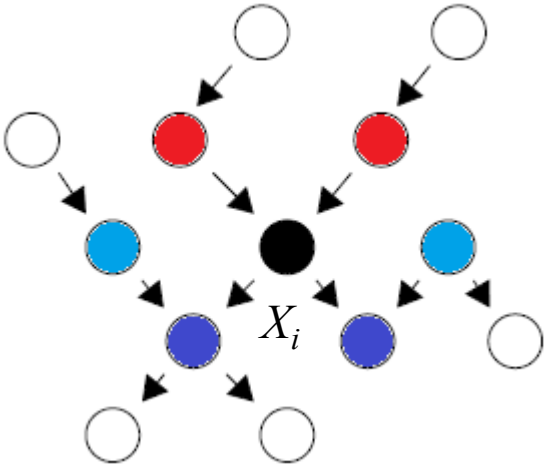
$\boldsymbol{u}_i$ can be reduced to the ***Markov Blanket*** of X_i for computing $\mathrm{P}(X_i|\boldsymbol{u}_i)$

When sampling X_i, all the previous variables $X_1 \sim X_{i-1}$ should use the ***updated*** assignment

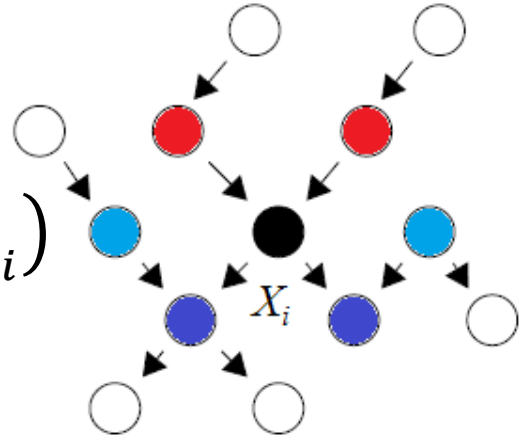
Gibbs Sampling for BNs

- How do we calculate $P(X_i \mid x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$?
- The Markov blanket of X_i, including its parents, its children and the parents of its children

- Denote the children of X_i as $\boldsymbol{Y}=(Y_1,\ldots,Y_k)$
- It is easy to show that
  - $\mathrm{P} = \mathrm{P}\left(X_i|pa(X_i),\boldsymbol{y},pa(\boldsymbol{y})_{-X_i}\right)$
  - $\propto \mathrm{P}\left(X_i|pa(X_i)\right)\mathrm{P}\left(\boldsymbol{y}|X_i,pa(\boldsymbol{y})_{-X_i}\right)$



Gibbs Sampling for BNs



- How to sample from the posterior?
  - $\tilde{\mathrm{P}}(X_i) = \mathrm{P}(X_i|pa(X_i))\mathrm{P}(\boldsymbol{y}|X_i, pa(\boldsymbol{y})_{-X_i})$
- Simply calculate
  - $\tilde{\mathrm{P}}(X_i = x_{ij})$ for each assignment of X_i
- Normalize above un-normalized posterior
  - $\mathrm{P}(X_i = x_{ij}) = \tilde{\mathrm{P}}(X_i = x_{ij}) / \sum_j \tilde{\mathrm{P}}(X_i = x_{ij})$
- Generate the updated value of X_i according to above normalized probability

Gibbs Sampling for MNs

- How do we calculate $P(X_i \mid x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$?
- The Markov blanket of X_i, including all its neighbors in the undirected graph
- We can derive that

For MN, you must use a **local ratio** rather than a local probability.

$$-\frac{P(X_i=x_{ij}|\boldsymbol{x}_{-i})}{P(X_i=x_{i0}|\boldsymbol{x}_{-i})}=\frac{\frac{1}{Z}\prod_{c_i/X_i\notin c_i}\pi(c_i)\prod_{c_i/X_i\in c_i}\pi(c_i)|_{\boldsymbol{x}_{-i},X_i=x_{ij}}}{\frac{1}{Z}\prod_{c_i/X_i\notin c_i}\pi(c_i)\prod_{c_i/X_i\in c_i}\pi(c_i)|_{\boldsymbol{x}_{-i},X_i=x_{i0}}}$$

$$-=\prod_{c_i/X_i\in c_i}\frac{\pi(c_i)|_{\boldsymbol{x}_{-i},X_i=x_{ij}}}{\pi(c_i)|_{\boldsymbol{x}_{-i},X_i=x_{i0}}}$$

Only **the cliques defined on X_i (remind Markov blanket)** will affect the posterior!

Gibbs Sampling for MNs

- Define a ground state x_{i0} for X_i
- Find all the cliques defined on X_i and its neighbors
  - $C_k|_{X_i \in C_k} \subseteq X_i \cup \mathrm{MB}(X_i)$
- For each assignment $X_i = x_{ij}$ calculate the un-normalized probability (or ratio)
  - $\tilde{P}(X_i = x_{ij}) = \prod_{c_i / X_i \in c_i} \frac{\pi(c_i)|_{x_{-i}, X_i = x_{ij}}}{\pi(c_i)|_{x_{-i}, X_i = x_{i0}}}$
- Normalized above probability and generate the updated value of X_i accordingly

Generate the hidden variables using current parameter $\boldsymbol{Y}^t \sim \mathrm{P}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}^t)$

HMM: Inference (Gibbs Sampling)

- Initialization: generate the labeling sequencing according to the prior probability

$$\pi \Rightarrow Y_t = y_i, \quad 1 \le t \le T$$

- Re-generate Y_t according to the initial setting

$$P(y_t \mid Y, X, \theta) = P(y_t \mid y_{t-1}, y_{t+1}, x_t, \theta)$$

$$= \frac{P(y_{t+1}, x_t \mid y_t, y_{t-1}, \theta) P(y_t \mid y_{t-1}, \theta)}{P(y_{t+1}, x_t \mid y_{t-1}, \theta)}$$

$$\propto P(y_{t+1} \mid y_t, \theta) P(x_t \mid y_t, \theta) P(y_t \mid y_{t-1}, \theta)$$

$$= t_{y_t, y_{t+1}} e_{y_t, e_t} t_{y_{t-1}, y_t}$$

We get a "score" proportional to the probability. So we can randomly generate Y_t based on these scores.

HMM: Parameter Re-Estimation

- Simply update the parameter in transition and emission probability matrix
  - $t_{i \to j} = \frac{\sum_{t=1}^{T} \mathrm{I}(Y^{t+1}=j, Y^{t}=i)}{\sum_{t=1}^{T} \mathrm{I}(Y^{t}=i)}$
  - $e_{i \to k} = \frac{\sum_{t=1}^{T} \mathrm{I}(Y^{t}=i, X^{t}=k)}{\sum_{t=1}^{T} \mathrm{I}(Y^{t}=i)}$

Note: $\mathrm{I}(\cdot)$ is an indicator function

- *Please consider the burn-in time issue (need to remove the first few samples)*
- *If the chain is not long enough, run multiple runs of step 1 and then do step 2*

Examples for Gibbs Sampling

- Need one volunteer to draw an acyclic directed graph with no more than nine nodes
- Let's write down Gibbs sampling for it
- Draw the moralized undirected graph and write down Gibbs sampling for it

Metropolis-Hastings Algorithm

- Unlike Gibbs sampling, *Metropolis-Hastings* algorithm can generate next-samples from any transition distribution (similar as importance sampling) rather than a particular distribution

- A new factor named *acceptance probability* is introduced to decide whether accept a transition $A(x \to x')$
  - $\mathcal{T}(x \to x') = \mathcal{T}^Q(x \to x')A(x \to x')$
  - $\mathcal{T}(x \to x) = \mathcal{T}^Q(x \to x) + \sum_{x' \neq x} \mathcal{T}^Q(x \to x')\big(1 - A(x \to x')\big)$

Metropolis-Hastings Algorithm

- When the Markov chain is stationary, the mutual transition probability should be equal
  - From x to x': $\pi(x)\mathcal{T}^{Q}(x \to x')A(x \to x')$
  - From x' to x: $\pi(x')\mathcal{T}^{Q}(x' \to x)A(x' \to x)$
  - The two probabilities are equal
- We can derive that
  - $A(x \to x') = \min\left[1, \frac{\pi(x\prime)\mathcal{T}^{Q}(x\prime \to x)}{\pi(x)\mathcal{T}^{Q}(x \to x\prime)}\right]$
  - Satisfy the above equation

MH Algorithm for Continuous Probability

- Above proposal is also correct for continuous distribution (probability density function)
- The proposal distribution transition probability is commonly set as multi-variate Gaussian
  - $\tau(x|x') = \mathcal{N}(x', \Sigma)$
  - $\pi(x) = p(x)$
  - $A(x \to x') = \min\left[1, \frac{\pi(x\prime)\tau(x\prime \to x)}{\pi(x)\tau(x \to x\prime)}\right]$

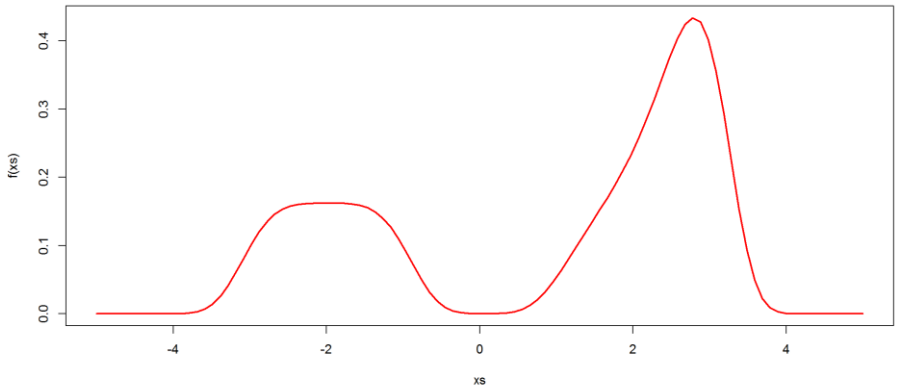
Gaussian transition probability can be regarded as *a random walk centered at the current state*

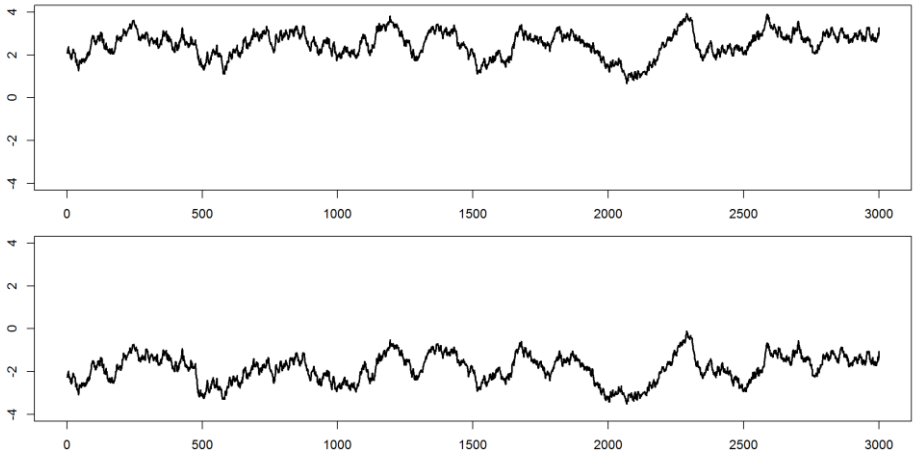
MH Algorithm for Continuous Probability

- If the proposal transition probability is commonly chosen as a Gaussian distribution centered at current state of the variables, the transition is symmetric (random walk process)!
  - $\tau(x|x') = \tau(x'|x)$
- Then, the acceptance rate is only related to its steady distribution or the target distribution
  - $A(x \to x') = \min\left[1, \frac{\pi(x\prime)}{\pi(x)}\right] = \min\left[1, \frac{p(x\prime)}{p(x)}\right]$

Disadvantages of MH Algorithm

- Inefficient transitions between different states
  - Improper transition probability: many rejections
  - The correlations of samples are high





(Hybrid) Hamiltonian Monte Carlo

- Hamiltonian
  - The total energy of a system depends not only the potential energy but also kinetic energy
  - Example: push a ball on a frictionless surface with peaks. *The height* of the ball determines its potential energy and *the speed* determines its kinetic energy
- Main idea of HMC algorithm
  - The potential energy is derived from the target distribution (the energy function of Gibbs distribution) and an artificial kinetic energy is defined on the same set of variables for improved sampling

Hamiltonian

- In quantum mechanics, a Hamiltonian is an operator corresponding to the total energy of the system in most of the cases
- In HMC, model samples are obtained by simulating a physical system, where particles move about a high-dimensional landscape, subject to potential and kinetic energies

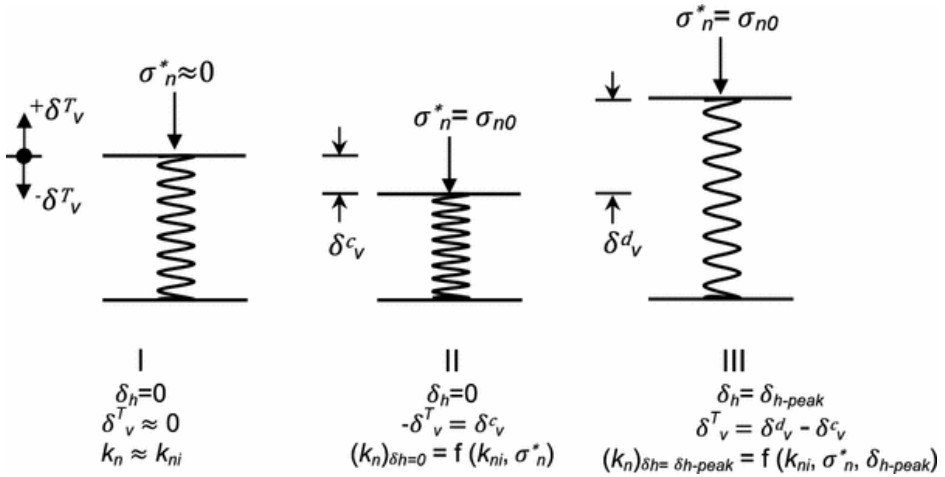
$$\mathcal{H}(s,\phi) = E(s) + K(\phi) = E(s) + \frac{1}{2}\sum_i \phi_i^2$$

E is the potential energy function as in the Gibbs distribution U(*s*); and the derivative of state variable *s* is the speed ϕ

Hamiltonian Dynamics

- For a frictionless system without any input. The total energy of the system is determined by its initial state: position (potential energy) and velocity (kinetic energy)
- We can derive

$$\frac{ds_i}{dt} = \frac{\partial \mathcal{H}}{\partial \phi_i} = \phi_i$$
$$\frac{d\phi_i}{dt} = -\frac{\partial \mathcal{H}}{\partial s_i} = -\frac{\partial E}{\partial s_i}$$



HMC: Leap-Frog Algorithm

- Theory: Hamiltonian dynamics

$$\mathcal{H}(s,\phi) = E(s) + K(\phi) = E(s) + \frac{1}{2}\sum_i \phi_i^2$$

$$\frac{ds_i}{dt} = \frac{\partial \mathcal{H}}{\partial \phi_i} = \phi_i$$
$$\frac{d\phi_i}{dt} = -\frac{\partial \mathcal{H}}{\partial s_i} = -\frac{\partial E}{\partial s_i}$$

Set $E(s)$ as the potential function of target distribution P

- Leap-Frog algorithm (inner loop)

$$\phi_i(t+\epsilon/2) = \phi_i(t) - \frac{\epsilon}{2}\frac{\partial}{\partial s_i}E(s(t))$$
$$s_i(t+\epsilon) = s_i(t) + \epsilon\phi_i(t+\epsilon/2)$$
$$\phi_i(t+\epsilon) = \phi_i(t+\epsilon/2) - \frac{\epsilon}{2}\frac{\partial}{\partial s_i}E(s(t+\epsilon))$$

Discretized time sliced for systems simulation

HMC: Leap-Frog Algorithm

- The Markov chain by the following steps
  - sample a new velocity from a univariate Gaussian distribution
  - perform *n* leap-frog steps to obtain the new state *X'*
    - According to the algorithm in the previous slide
  - perform accept/reject move of *X'*
    - according to MH algorithm

$$p_{acc}(\chi, \chi') = min\left(1, \frac{\exp(-\mathcal{H}(s', \phi')}{\exp(-\mathcal{H}(s, \phi)}\right)$$

$$\phi_i(t + \epsilon/2) = \phi_i(t) - \frac{\epsilon}{2}\frac{\partial}{\partial s_i}E(s(t))$$

$$s_i(t + \epsilon) = s_i(t) + \epsilon\phi_i(t + \epsilon/2)$$

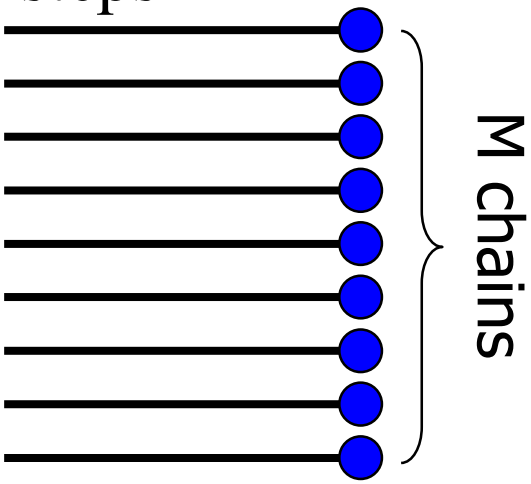
$$\phi_i(t + \epsilon) = \phi_i(t + \epsilon/2) - \frac{\epsilon}{2}\frac{\partial}{\partial s_i}E(s(t + \epsilon))$$

MCMC in Practice

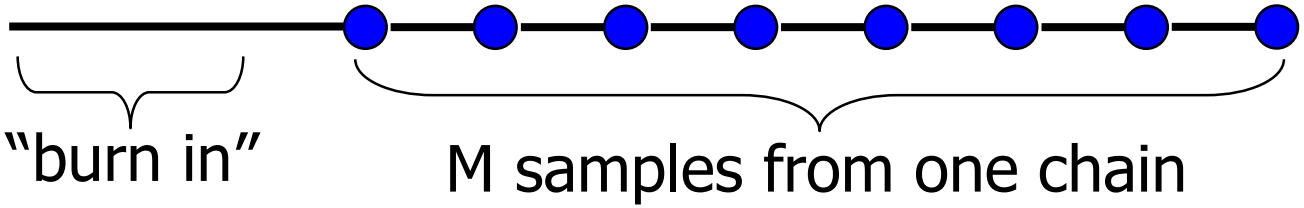
- We need to wait until the ***burn-in time*** has ended: the chain enters the stationary state
- Once the burn-in time ended, all samples are from the stationary distribution
- Generally, the burn-in time is hard to estimate
- Since no theoretical guarantees exist, application of Markov chains is somewhat of an art
- Note: samples from the same chain are *correlated*

Sampling Strategy

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



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



Comparing Strategies

Due to the increased computational capability, *Strategy I* is used more frequently

Strategy I:

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

Strategy II:

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

Hybrid strategy:

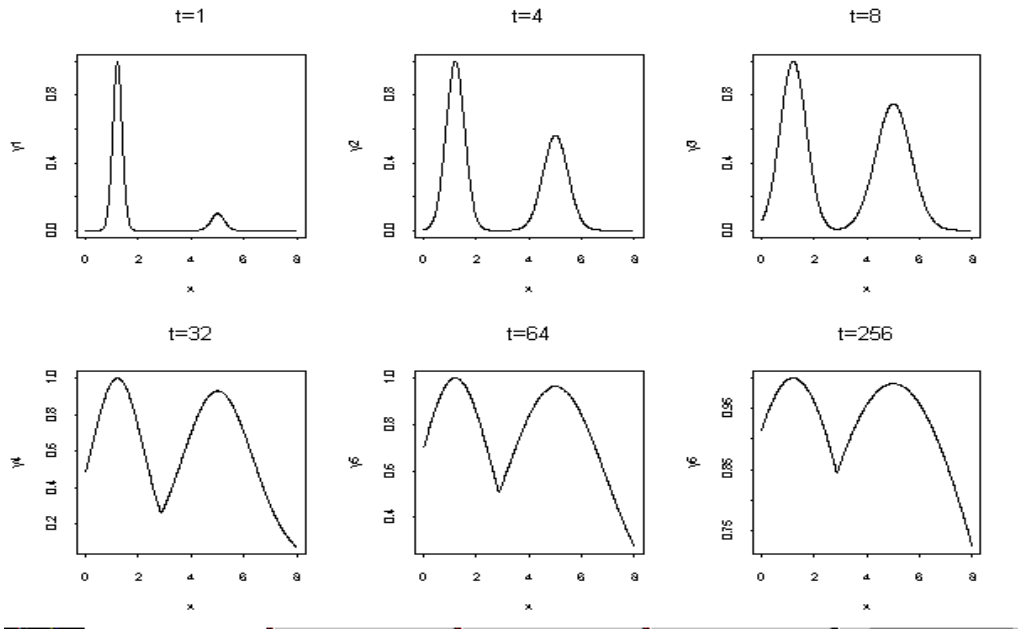
- run several chains, and sample few samples from each
- Combines benefits of both strategies

Simulated Annealing for Ergodicity

- For most applications, the target function is usually represented as a exponential family (like a energy function): $\tilde{P} \propto \exp\left(-U(\phi)\right)$
- If $U(\phi)$ has several sharp peaks, the sampling process will be *trapped* by local minimum
- At the beginning of the sampling, we want to quickly transition between different states

Simulated Annealing for Ergodicity

- A *temperature* is introduced
  - $\tilde{P} \propto \exp\left(-U(\phi)\right) \rightarrow \tilde{P} \propto \exp\left(-\frac{1}{T}U(\phi)\right)$
- At the beginning, T is set as a large number and then T is gradually reduced to 1



The shape of the function is "smooth" when T is large

The sampling can transit among different peaks and ensure *ergodicity*!

References

- Liu. **Monte Carlo Strategies in Scientific Computing**. *Springer Series in Statistics* 2001.

- Besag. **Markov Chain Monte Carlo Methods for Statistical Inference**. 2004.

- Neal, Radford M (2011). **MCMC Using Hamiltonian Dynamics**. In Steve Brooks; Andrew Gelman; Galin L. Jones; Xiao-Li Meng. Handbook of Markov Chain Monte Carlo. Chapman and Hall/CRC.

Review: Particle-Based Inference

General principle (χ denotes all of the network variables. $Y \subseteq \chi$)

- Generate samples $\xi[1], ..., \xi[M]$ from the distribution $P(\chi)$
- Estimate function by $E_P(f(\chi)) \approx \frac{1}{M}\sum_{m=1}^{M} f(\xi[m])$

- Why does this strategy work?
  - Recall the definition of probability

$$P(x) \approx \frac{n_x}{n_t}.$$

- *Key* for particle-based approximate inferences?
  - How to get samples from posteriors P(*Y*|*E*=*e*)

Review: Particle-Based Inference

- Likelihood Weighting and Importance Sampling
  - Likelihood Weighting
  - Unnormalized Importance Sampling
  - Normalized Importance Sampling
  - Importance Sampling for Bayesian Networks
- Markov Chain Monte Carlo Methods
  - Markov Chain and Stationary Distribution
  - Gibbs Sampling
  - Metropolis-Hastings Algorithm / HMC

The End of Chapter 7

Particle-based methods are powerful for inferences in PGMs