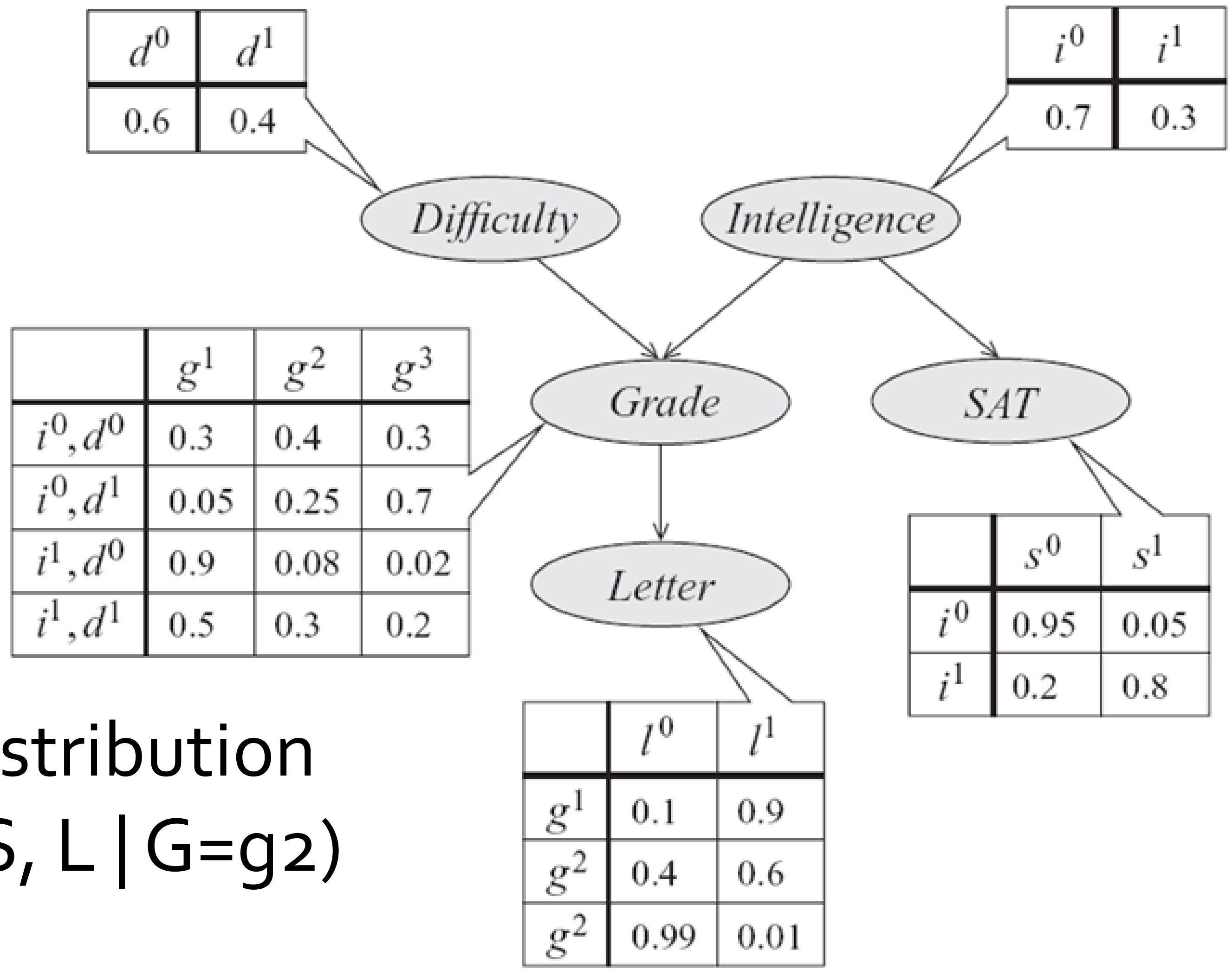
MCMC based Sampling

Our Goal (reminder)

- We need independent samples to estimate a desired distribution (usually posterior distribution, $p(Y|e)$)
- We can setup a Markov chain that converges to a stationary distribution
- Satisfying detailed balance is an easy way to guarantee convergence to equilibrium

Example



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^2	0.99	0.01

- Desired distribution
  P(D, I, S, L | G=g2)

Problem Setup

- Usually the state space is huge but in our toy example:

$$|\mathrm{I}| = |D| \times |I| \times |S| \times |L| \times |G| = 2 \times 2 \times 2 \times 2 \times 3 = 48$$

- Samples are shown as **x**:(d,i,s,l,g)
- Given our graphical model we can simply evaluate every sample as:

$$p(x) = p(d)p(i)p(s \mid i)p(g \mid d,i)p(l \mid g)$$

Average of Samples Converge to the Expectation

- Ergodicity (special case of law of large numbers). If a Markov process is positive recurrent with invariant distribution π then

$$P\left(\frac{1}{n}\sum_{k=0}^{n-1} f\left(X_k\right) \to \overline{f} \; as \; n \to \infty\right) = 1$$

- Where

$$\overline{f} = \mathrm{E}_\pi\left(f\right) = \sum_{i \in I} \pi_i f_i$$

- And π is the unique invariant distribution

Proposal Distribution (reminder)

- We cannot always sample efficiently from *P(X)*
- But we might be able to evaluate *P*(X) efficiently
- In that case, we could sample efficiently from some other "simpler" distribution called the *proposal distribution* Q(X)

Proposal Distribution (cont.)

- if $Q(X) \neq 0$ where $P(X) \neq 0$

$$
\begin{aligned}
E_{P(x)}[f(x)] &= E_{Q(x)}\left[f(x)\frac{P(x)}{Q(x)}\right] \\
&= \sum_x Q(x) f(x) \frac{P(x)}{Q(x)} \\
&= \sum_x f(x) P(x)
\end{aligned}
$$

How to find desired distribution?

- Several ways we can do this with MCMC
  - Metropolis
  - Metropolis Hasting
  - Gibbs Sampling

General form of MCMC

1. Sample a point from a *proposal distribution*

$$q(y \mid x)$$

2. Compute the *importance ratio*

$$r = \frac{p(y)\,q(x \mid y)}{p(x)\,q(y \mid x)}$$

3. Move to the new state with an *transition probability* (related to importance ratio)

$$P(x \to y) = q(y \mid x)\{r \wedge 1\}$$

Metropolis Algorithm

- The probability of moving from one state to another *must be symmetric*:

$$q(x \mid y) = q(y \mid x)$$

Metropolis Algorithm

- Importance ratio

$$r = \frac{\mathrm{p}(y)\,\mathrm{q}(x \mid y)}{\mathrm{p}(x)\,\mathrm{q}(y \mid x)} = \frac{\mathrm{p}(y)}{\mathrm{p}(x)}$$

- Transition probability

$$\mathrm{P}(\mathrm{x} \to \mathrm{y}) = \mathrm{q}(\mathrm{y} \mid \mathrm{x})\{\mathrm{r} \wedge 1\}$$

Metropolis Algorithm

```
X<-randomValue()
while(1):
    Y=generateSample(q(Y|X))
    r=p(Y)/p(X)
    if(r>1):
        X=Y
    else:
        t=generateSample(uniform(0,1))
        if(t<r):
            X=Y
```

Convergence of Metropolis Algorithm

$$p(x)\mathrm{P}(x \to y) = p(x)\,q(y \mid x)\left\{\frac{p(y)}{p(x)} \wedge 1\right\}$$

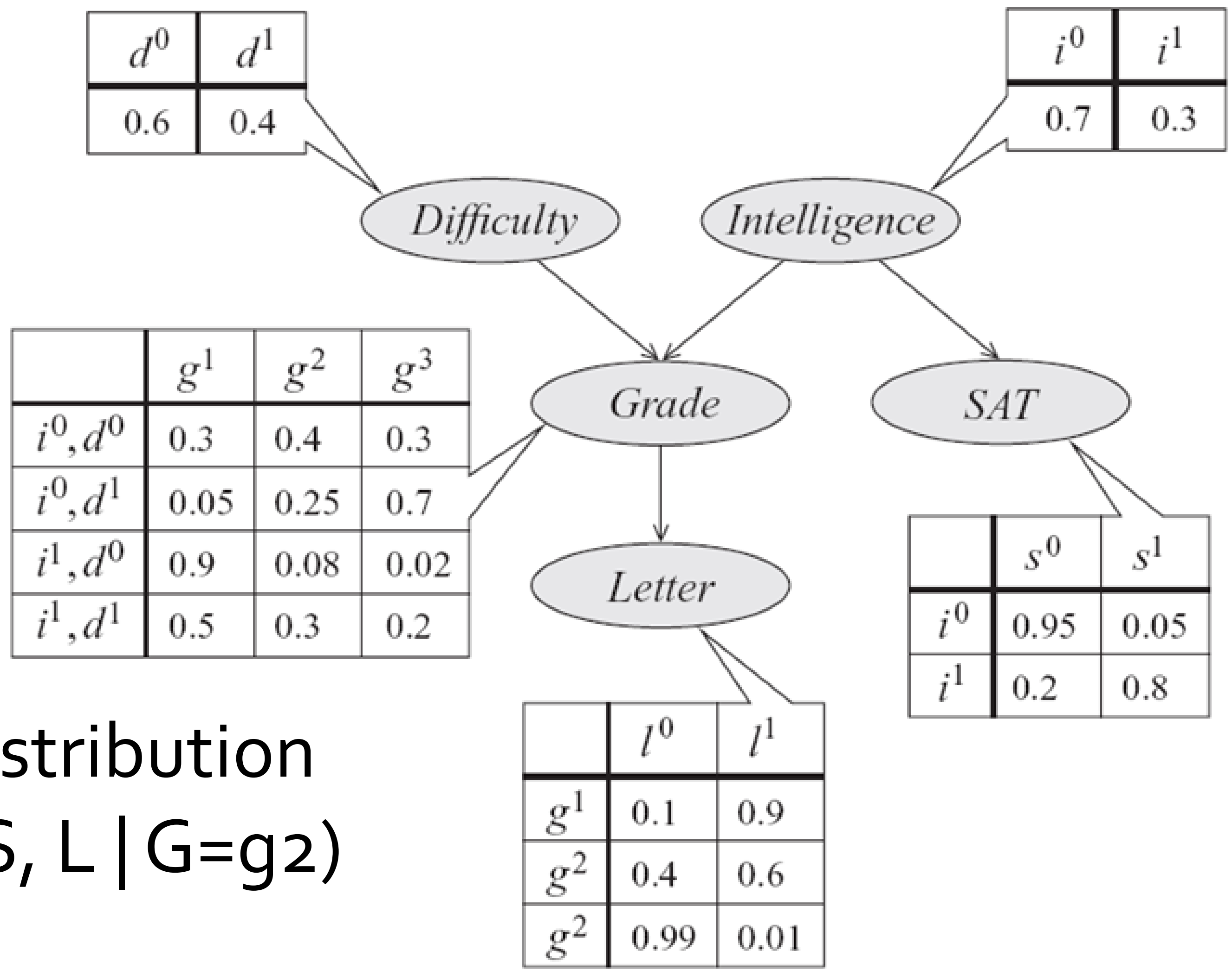
$$assume\ p(y) > p(x)\ then$$

$$= p(x)q(y \mid x) \times 1 = p(x)q(x \mid y)$$

$$= p(x)q(x \mid y)\frac{p(y)}{p(y)} = p(y)q(x \mid y)\frac{p(x)}{p(y)}$$

$$= p(y)q(x \mid y)\left\{\frac{p(x)}{p(y)} \wedge 1\right\} = p(y)\mathrm{P}(y \to x)$$

Example



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^2	0.99	0.01

- Desired distribution

  P(D, I, S, L | G=g2)

Problem Setup

- Usually the state space is huge but in our toy example:

$$|\mathrm{I}| = |D| \times |I| \times |S| \times |L| \times |G| = 2 \times 2 \times 2 \times 2 \times 3 = 48$$

- Samples are shown as **x**:(d,i,s,l,g)
- Given our graphical model we can simply evaluate every sample as:

$$p(x) = p(d)p(i)p(s|i)p(g|d,i)p(l|g)$$

Metropolis(example)

- Let proposal distribution be uniform
- Start with random **x**:$(d^0, i^1, s^1, l^1, g^2)$

$$p(x) = 0.6 \times 0.3 \times 0.8 \times 0.08 \times 0.6 = 0.007$$

- Obtain **y** by uniformly sampling from I

y:$(d^1, i^1, s^0, l^0, g^2)$

$$p(y) = 0.4 \times 0.3 \times 0.2 \times 0.3 \times 0.4 = 0.003$$

$$\frac{p(y)}{p(x)} \wedge 1 = \frac{0.003}{0.007} \wedge 1 = 0.42$$

- Draw a random value between [0,1]. If it is ">" than 0.42 reject it. Let say 0.32, so accept **y**.

Metropolis(example)

- Again obtain the next **y** by uniformly sampling from I.
  y:(d^0,i^0 ,s^0,l^0 ,g^2)

$$p(y) = 0.6 \times 0.7 \times 0.95 \times 0.4 \times 0.4 = 0.064$$

$$\frac{p(y)}{p(x)} \wedge 1 = \frac{0.064}{0.003} \wedge 1 = 1$$

- So accept **y** and so on

Metropolis-Hastings Algorithm

- The proposal distribution need not be symmetric
- Now the proposal distribution is factored into the importance ratio
- Follows the general form introduced earlier
- This is more general (and useful) than Metropolis algorithm

Metropolis-Hastings

- Importance ratio

$$r = \frac{p(y)q(x|y)}{p(x)q(y|x)}$$

- Transition probability

$$\mathrm{P}(x \rightarrow y) = q(y|x)\{r \wedge 1\}$$

Metropolis-Hastings Algorithm

```
X<-randomValue()
while(1):
    Y=generateSample(q(Y|X))
    r=p(Y)q(X|Y)/p(X)q(Y|X)
    if(r>1):
        X=Y
    else:
        t=generateSample(uniform(0,1))
        if(t<r):
            X=Y
```

Convergence of Metropolis-Hasting Algorithm

$$p(x)\mathrm{P}(x \to y) = p(x)q(y|x)\{\frac{p(y)q(x|y)}{p(x)q(y|x)} \wedge 1\}$$

$$assum\ p(y)q(x|y) > p(x)q(y|x)$$

$$= p(x)q(y|x) = p(x)q(y|x)\frac{p(y)q(x|y)}{p(y)q(x|y)}$$

$$= p(y)q(x|y)\frac{p(x)q(y|x)}{p(y)q(x|y)} = p(y)q(x|y)\frac{p(x)q(y|x)}{p(y)q(x|y)}$$

$$= p(y)q(x|y)\{\frac{p(x)q(y|x)}{p(y)q(x|y)} \wedge 1\}$$

$$= p(y)\mathrm{P}(y \to x)$$

Gibbs Sampling Algorithm

- Special case of Metropolis-Hastings algorithm
- The proposal distribution has a given form (i.e. it is not designed on a problem by problem basis)
- Samples the components of the outcome vector one at a time using the marginal distribution, where all other components are fixed to values from previous samples

Proposal Distribution of Gibbs Sampling

- Proposal distribution

$$q(y\,|\,x) = \begin{cases} p(y_j\,|\,x_{-j}) & y_{-j} = x_{-j} \\ 0 & otherwise \end{cases}$$

- Importance ratio is unity. We *always* accept.

$$r = \frac{\mathrm{p}(y)\,\mathrm{q}(x\,|\,y)}{\mathrm{p}(x)\,\mathrm{q}(y\,|\,x)} = 1$$

Gibbs Sampling Algorithm

```
X<-randomValue()
while(1):
     for(j=0;j<len(X);j++)
          y=generateSample(p(y|X[0:j],X[j+1:len(x)]))
          X[j]=y
```

Proof that acceptance rate is unity

$$r = \frac{p(y)q(x|y)}{p(x)q(y|x)}$$

$$= \frac{p(y)p(x_j|y_{-j})}{p(x)p(y_j|x_{-j})} = \frac{p(y)p(x_j|x_{-j})}{p(x)p(y_j|y_{-j})}$$

$$= \frac{p(y)p(x_j,x_{-j})p(y_{-j})}{p(x)p(y_j,y_{-j})p(x_{-j})} = \frac{p(y)p(x)p(y_{-j})}{p(x)p(y)p(x_{-j})}$$

$$= \frac{p(y_{-j})}{p(x_{-j})} = 1$$

Gibbs Sampling

- Let start with **x**:$(d^0, i^1, s^1, l^1, g^2)$
- We will sample D,I,S,L and G in a round robin manner
- Sample D:
  - *s1:p*$(d^0, i^1, s^1, l^1, g^2)$= 0.6x0.3x0.8x0.08x0.6=0.007
  - *s2:p*$(d^1, i^1, s^1, l^1, g^2)$= 0.4x0.3x0.8x0.3x0.6 =0.017
  - Generate a random number between [0,0.007+0.017], if it is greater than 0.007 move to s2 otherwise move to s1. let say random value is 0.011. so current state will become **x:**$(d^1, i^1, s^1, l^1, g^2)$

Gibbs Sampling

- Sample I:
  - *s1:p*(d^1, i^0, s^1, l^1, g^2)= 0.4x0.7x0.05x0.25x0.6=0.002
  - *s2:p*(d^1, i^1, s^1, l^1, g^2)= 0.4x0.3x0.8x0.3x0.6 =0.017
  - Generate a random number between [0,0.002+0.017], if it is greater than 0.002 move to s2 otherwise move to s1. let say random value is 0.001. so current state will become **x:**(d^1, i^0, s^1, l^1, g^2)
- **Sample S and so on**

Practical issues

- Learning how a tool works is one thing, using it in a practical situation is another.
- MCMC is no different.

Convergence

- Convergence and ergodicity theorems state

$$P(X_n = j) \to \pi_j \text{ as } n \to \infty$$

$$\frac{1}{n}\sum_{k=0}^{n-1} f(X_k) \to \overline{f} \; as \; n \to \infty$$

- This is nice, but they don't say anything about *how fast* they converge

Convergence

- How can we tell if our sample has accurately characterized our desired distribution?
- How big should n be before we trust our result?
- How long do is this MCMC thing going to take?

Mixing time

- Mixing is a measure of how long a Markov process takes to get near its equilibrium.
- There are many analytic ways to calculate this, but only for completely characterized Markov chains.
- For sampling, we would like our process to converge quickly from an arbitrary point in the space. This is called "mixing well."

Burn In

- We start the Markov chain from a random point in the sample space.
- This point and the points in its neighborhood might be very unlikely according to our distribution.
- Run the Markov chain for many iterations before using the sampled points. This is called *burn in.*

Burn In

- How long should our burn in last?
  1. Analytically evaluate the convergence rate of our chain
     1. Usually results in overly pessimistic estimates
  2. Use convergence diagnostics
     1. Do not guarantee convergence
  3. Use perfect simulation
     1. Only valid for specific types of problems

Thinning

- Consecutive outputs from the chain can be highly correlated
- Saving all of the sampled points can be expensive
- We can save only every k outputs, which is called *thinning*
- Sample of our samples

Review

- Markov chains are ergodic

$$P\left(\frac{1}{n}\sum_{k=0}^{n-1} f\left(X_k\right) \rightarrow \overline{f} \; as \; n \rightarrow \infty\right) = 1$$

- MCMC is a technique for importance sampling
- Metropolis requires a symmetric proposal distribution
- Metropolis-Hasting does not

Review

- Gibbs sampling is special case of Metropolis Hasting that always accepts the proposal
- In implementing MCMC, convergence and independence are major concerns
  - Burn in (convergence rate, convergence diagnostics , perfect sampling)
  - Thinning