CSCI 699 - ProbGen

# Probabilistic and Generative Models

Willie Neiswanger

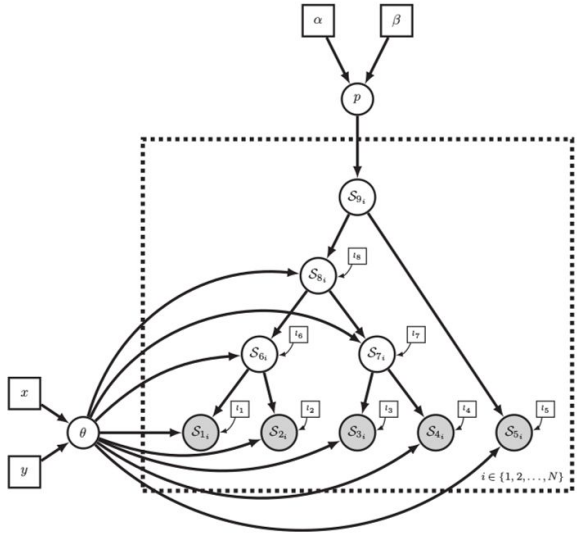
# Lecture 2 - Introduction to PGMs

# Plan for Today

# Plan for Today

**Lecture:** PGMs — using ideas from both probability and graph theory to derive efficient machine learning algorithms

- Directed vs undirected graphical models: Bayesian networks and Markov random fields.
- Plate notation, generative process notation.
- Observed variables vs. latent variables vs. (hyper)parameters.
- Famous/classic Bayes Nets (HMMs, GMMs, LDA, VAE, etc.)
- Pros and Cons of Bayes Nets vs MRFs.
- Famous/classic MRFs (Ising Model, CRF).




Source: "Probabilistic Graphical Model Representation in Phylogenetics", Höhna 2014

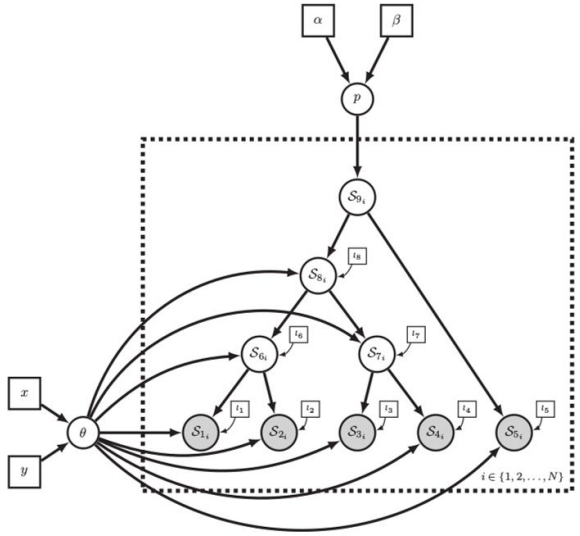
# Plan for Today

**Lecture:** PGMs — using ideas from both probability and graph theory to derive efficient machine learning algorithms

- Directed vs undirected graphical models: Bayesian networks and Markov random fields.
- Plate notation, generative process notation.
- Observed variables vs. latent variables vs. (hyper)parameters.
- Famous/classic Bayes Nets (HMMs, GMMs, LDA, VAE, etc.)
- Pros and Cons of Bayes Nets vs MRFs.
- Famous/classic MRFs (Ising Model, CRF).

**After:**

- Class introductions and research interests, for group project.
- Compute resources.




Source: "Probabilistic Graphical Model Representation in Phylogenetics", Höhna 2014

# **Lecture 2 (January 24)** — Intro to Probabilistic Graphical Models (PGMs)

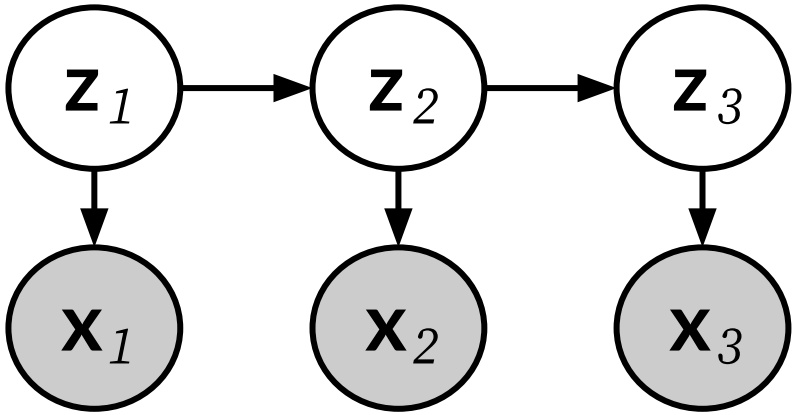
At a high level:  two main “types” of PGMs.

# **Lecture 2 (January 24)** — Intro to Probabilistic Graphical Models (PGMs)

At a high level: two main "types" of PGMs.

**Bayesian Networks (directed)**

*E.g.*, Hidden Markov Model (HMM)

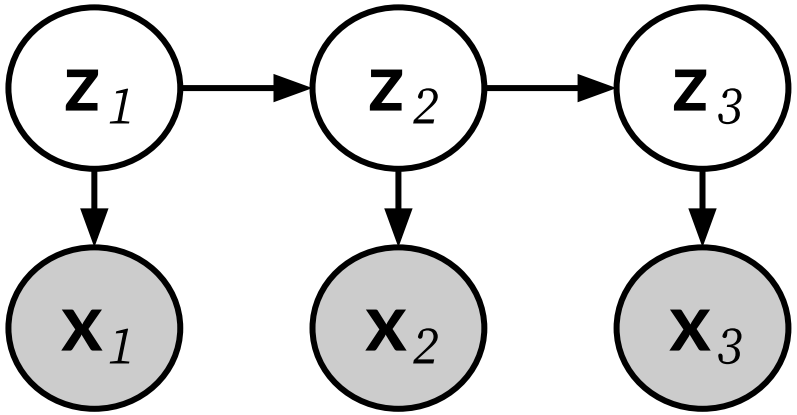


# **Lecture 2 (January 24)** — Intro to Probabilistic Graphical Models (PGMs)

At a high level: two main "types" of PGMs.

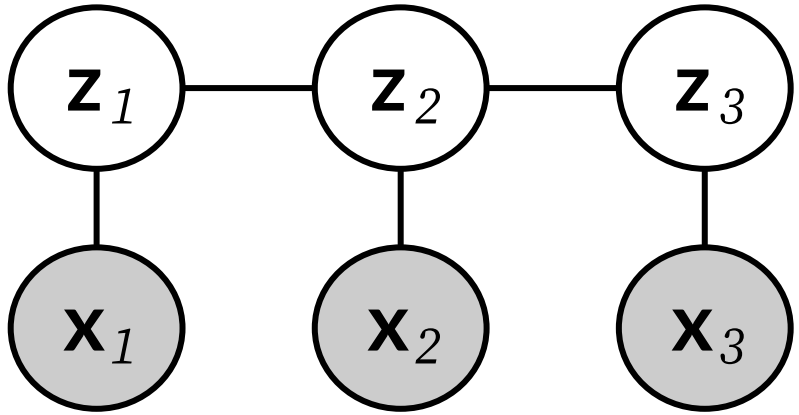
**Bayesian Networks (directed)**

*E.g.*, Hidden Markov Model (HMM)




**Markov Random Fields (undirected)**

*E.g.*, Conditional Random Field (CRF)

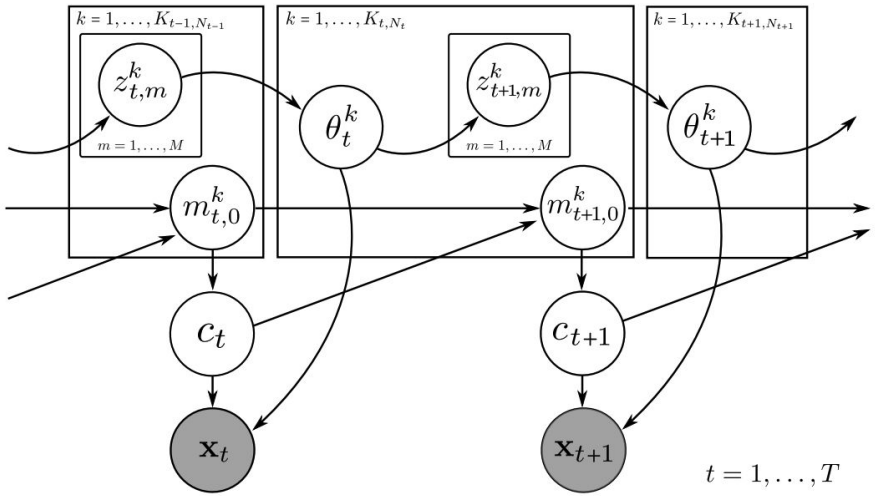


# **Lecture 2 (January 24)** — Intro to Probabilistic Graphical Models (PGMs)

At a high level: two main "types" of PGMs.

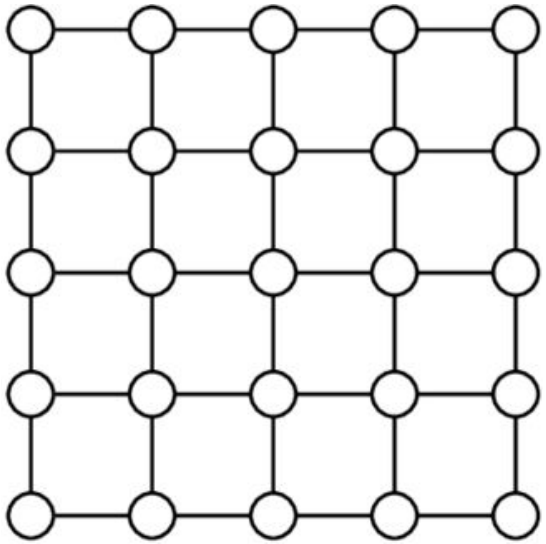
## Bayesian Networks (directed)

*E.g.*, Dependent Dirichlet Process Mixture of Experts (DDPMO)




## Markov Random Fields (undirected)

*E.g.*, Ising Model



# **Lecture 2 (January 24)** — Intro to Probabilistic Graphical Models (PGMs)

Today, we will focus most of our time on **Bayesian networks**.

# **Lecture 2 (January 24)** — Intro to Probabilistic Graphical Models (PGMs)

Today, we will focus most of our time on **Bayesian networks**.

They have certain advantages over Markov Random Fields, which we will discuss.

The converse is also true: MRFs have certain advantages.

# **Lecture 2 (January 24)** — Intro to Probabilistic Graphical Models (PGMs)

Today, we will focus most of our time on **Bayesian networks**.

They have certain advantages over Markov Random Fields, which we will discuss.

The converse is also true: MRFs have certain advantages.

And also:

Bayesian networks are more relevant to some of the approximate inference algorithms and deep generative models we will cover a bit later in the class.

# Directed PGMs:  Bayesian Networks

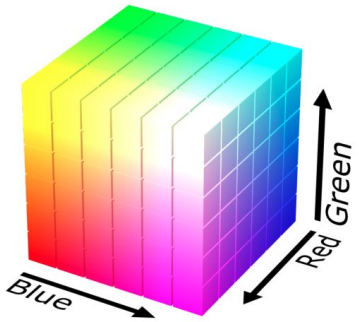
# Image Example

Modeling a single pixel in an image

# Image Example

Modeling a single pixel in an image $\Rightarrow$ 3 discrete random variables:

- R - Red Channel. $\text{Val}(R) = \{0, \ldots, 255\}$
- G - Green Channel. $\text{Val}(G) = \{0, \ldots, 255\}$
- B - Blue Channel. $\text{Val}(B) = \{0, \ldots, 255\}$

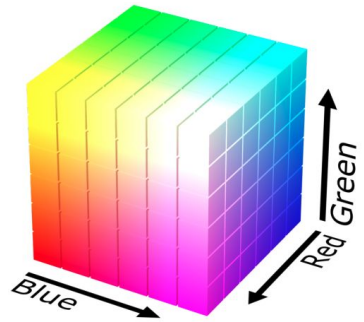



Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

Modeling a single pixel in an image ⇒ 3 discrete random variables:

- R - Red Channel. $\text{Val}(R) = \{0, \ldots, 255\}$
- G - Green Channel. $\text{Val}(G) = \{0, \ldots, 255\}$
- B - Blue Channel. $\text{Val}(B) = \{0, \ldots, 255\}$




Source: Stefano Ermon, Deep Generative Models (CS236) Class

Sampling from the joint distribution randomly generates a color for the pixel:

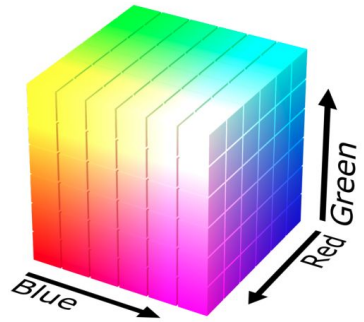
$$r, g, b \sim p(R, G, B)$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

Modeling a single pixel in an image ⇒ 3 discrete random variables:

- R - Red Channel. $\text{Val}(R) = \{0, \ldots, 255\}$
- G - Green Channel. $\text{Val}(G) = \{0, \ldots, 255\}$
- B - Blue Channel. $\text{Val}(B) = \{0, \ldots, 255\}$




Source: Stefano Ermon, Deep Generative Models (CS236) Class

Sampling from the joint distribution randomly generates a color for the pixel:

$$r, g, b \sim p(R, G, B)$$

Recall from prob overview:
a sample from the joint PDF.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** How many parameters do we need to specify the joint distribution

$$p(R = r,\ G = g,\ B = b) \quad ?$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** How many parameters do we need to specify the joint distribution

$$p(R = r,\ G = g,\ B = b) \quad ?$$

**Answer:** It's a discrete distribution with $256 \times 256 \times 256$ possible values ⇒

$256 \times 256 \times 256 - 1 = 16{,}777{,}215$ parameters

A lot!

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

Modeling an image

# Image Example

Modeling an image ⇒ For simplicity, consider black & white images with $n$ pixels.

# Image Example

Modeling an image ⇒ For simplicity, consider black & white images with $n$ pixels.

Model the image as a set of n random variables, $X_1, X_2, \ldots, X_n$, where

$$\text{Val}(X_i) = \{0, 1\} = \{\text{Black}, \text{White}\}$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

Modeling an image ⇒ For simplicity, consider black & white images with $n$ pixels.

Model the image as a set of n random variables, $X_1, X_2, \ldots, X_n$, where

$$\text{Val}(X_i) = \{0, 1\} = \{\text{Black}, \text{White}\}$$

How many possible images?

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

Modeling an image ⇒ For simplicity, consider black & white images with $n$ pixels.

Model the image as a set of n random variables, $X_1, X_2, \ldots, X_n$, where

$$\text{Val}(X_i) = \{0, 1\} = \{\text{Black}, \text{White}\}$$

How many possible images? ⇒ $\underbrace{2 \times \ldots \times 2}_{n \text{ times}} = 2^n$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** How many parameters do we need to specify the joint distribution

$$p(X_1 = x_1, \ \ldots, \ X_n = x_n) \ ?$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** How many parameters do we need to specify the joint distribution

$$p(X_1 = x_1, \ \ldots, \ X_n = x_n) \ ?$$

**Answer:** It's a discrete distribution with $2^n$ possible values ⇒

$$2^n - 1 \quad \text{parameters}$$

# Image Example

**Question:** If we simplify our joint density, can we reduce this number?

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** What if we assume that $X_1, X_2, \ldots, X_n$ are independent?

# Image Example

**Question:** What if we assume that $X_1, X_2, \ldots, X_n$ are independent?

**Answer:** Then $p(x_1, x_2, \ldots, x_n) = p(x_1)p(x_2)\cdots p(x_n)$.

Note: this is still a discrete distribution with $2^n$ possible values.

# Image Example

**Question:** What if we assume that $X_1, X_2, \ldots, X_n$ are independent?

**Answer:** Then $p(x_1, x_2, \ldots, x_n) = p(x_1)p(x_2)\cdots p(x_n)$.

Note: this is still a discrete distribution with $2^n$ possible values.

However, to specify a marginal distribution $p(x_i)$ requires *1* parameter.

Binary random variable specified by 1 parameter.

# Image Example

**Question:** What if we assume that $X_1, X_2, \ldots, X_n$ are independent?

**Answer:** Then $p(x_1, x_2, \ldots, x_n) = p(x_1)p(x_2)\cdots p(x_n)$.

Note: this is still a discrete distribution with $2^n$ possible values.

However, to specify a marginal distribution $p(x_i)$ requires *1* parameter.

⇒ Specifying $p(x_1, x_2, \ldots, x_n)$ requires only $n$ parameters.

⇒ The frequency of $2^n$ values can be described using only $n$ numbers

⇒ Big reduction!

Binary random variable specified by 1 parameter.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** What if we assume that $X_1, X_2, \ldots, X_n$ are independent?

**However**, independence might be too strong of an assumption!

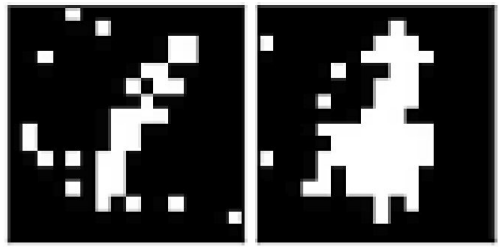
# Image Example

**Question:** What if we assume that $X_1, X_2, \ldots, X_n$ are independent?

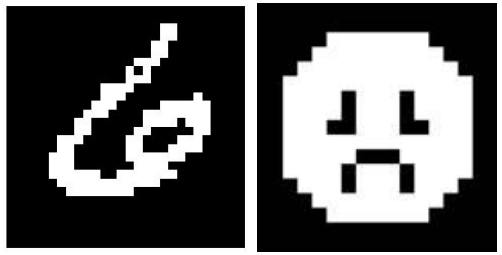
**However**, independence might be too strong of an assumption!

Model likely to be **not useful** .

For example, each pixel chosen independently when sampling does not yield realistic images.



vs.



Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** Can we think of another assumption that may be “in between”?

# Image Example

**Question:** Can we think of another assumption that may be “in between”?

To answer this, remember a couple of properties from last class:

# Image Example

**Question:** Can we think of another assumption that may be “in between”?

To answer this, remember a couple of properties from last class:

Joint PDF Factorization

Definition of conditional probability density function:

$$p(x_1 \mid x_2, \ldots, x_n) = \frac{p(x_1, x_2, \ldots, x_n)}{p(x_2, \ldots, x_n)}$$

# Image Example

**Question:** Can we think of another assumption that may be "in between"?

To answer this, remember a couple of properties from last class:

Joint PDF Factorization

Definition of conditional probability density function:

$$p(x_1 \mid x_2, \ldots, x_n) = \frac{p(x_1, x_2, \ldots, x_n)}{p(x_2, \ldots, x_n)}$$

Leads to the **factorization** (*i.e.*, way of writing) the joint PDF as:

$$p(x_1, x_2, \ldots, x_n) = p(x_1 \mid x_2, \ldots, x_n)\, p(x_2, \ldots, x_n)$$

# Image Example

**Question:** Can we think of another assumption that may be “in between”?

To answer this, remember a couple of properties from last class:

Chain Rule

# Image Example

**Question:** Can we think of another assumption that may be “in between”?

To answer this, remember a couple of properties from last class:

Chain Rule

Joint PDF factorization

$$p(x_1, x_2, \ldots, x_n) = p(x_n \mid x_1, x_2, \ldots, x_{n-1}) p(x_1, x_2, \ldots, x_{n-1})$$

# Image Example

**Question:** Can we think of another assumption that may be "in between"?

To answer this, remember a couple of properties from last class:

Chain Rule

Joint PDF factorization

(repeatedly!)

$$
\begin{aligned}
p(x_1, x_2, \ldots, x_n) &= p(x_n \mid x_1, x_2, \ldots, x_{n-1}) p(x_1, x_2, \ldots, x_{n-1}) \\
&= p(x_n \mid x_1, x_2, \ldots, x_{n-1}) p(x_{n-1} \mid x_1, x_2, \ldots, x_{n-2}) p(x_1, x_2, \ldots, x_{n-2}) \\
&= \ldots
\end{aligned}
$$

# Image Example

**Question:** Can we think of another assumption that may be "in between"?

To answer this, remember a couple of properties from last class:

Chain Rule

Joint PDF factorization

(repeatedly!)

$$
\begin{aligned}
p(x_1, x_2, \ldots, x_n) &= p(x_n \mid x_1, x_2, \ldots, x_{n-1}) p(x_1, x_2, \ldots, x_{n-1}) \\
&= p(x_n \mid x_1, x_2, \ldots, x_{n-1}) p(x_{n-1} \mid x_1, x_2, \ldots, x_{n-2}) p(x_1, x_2, \ldots, x_{n-2}) \\
&= \ldots \\
&= p(x_1) \prod_{i=2}^{n} p(x_i \mid x_1, \ldots, x_{i-1}) \\
&= p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2, x_1) \cdots p(x_n \mid x_{n-1}, \ldots, x_1)
\end{aligned}
$$

# Image Example

**Question:** Can we think of another assumption that may be “in between”?

# Image Example

**Question:** Can we think of another assumption that may be “in between”?

**Answer:**

⇒ First, Consider the chain rule:

$$p(x_1, x_2, \ldots, x_n) = p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2, x_1) \cdots p(x_n \mid x_{n-1}, \ldots, x_1)$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** Can we think of another assumption that may be "in between"?

**Answer:**

$p(x_1)$ requires 1 parameter

⇒ First, Consider the chain rule:

$$p(x_1, x_2, \ldots, x_n) = p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2, x_1) \cdots p(x_n \mid x_{n-1}, \ldots, x_1)$$

How many parameters? $1 + 2 + \ldots + 2^{n-1} = 2^n - 1$

# Image Example

**Question:** Can we think of another assumption that may be "in between"?

**Answer:**

$p(x_2 \mid x_1 = 0)$ requires 1 parameter

$p(x_2 \mid x_1 = 1)$ requires 1 parameter

⇒ First, Consider the chain rule:

$$p(x_1, x_2, \ldots, x_n) = p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2, x_1) \cdots p(x_n \mid x_{n-1}, \ldots, x_1)$$

How many parameters? $1 + 2 + \ldots + 2^{n-1} = 2^n - 1$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** Can we think of another assumption that may be “in between”?

**Answer:**

⇒ First, Consider the chain rule:

$$p(x_1, x_2, \ldots, x_n) = p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2, x_1) \cdots p(x_n \mid x_{n-1}, \ldots, x_1)$$

How many parameters? $1 + 2 + \ldots + 2^{n-1} = 2^n - 1$

(So this is the same as before, when we had no independence assumptions!)

⇒ the chain rule does not buy us anything.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** Can we think of another assumption that may be “in between”?

**Answer:**

Now suppose each $X_i$ is independent of variables $X_1 \dots X_{i-2}$ given $X_{i-1}$, *i.e.*,

$$X_i \perp (X_1 \dots X_{i-2}) \mid X_{i-1}$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** Can we think of another assumption that may be "in between"?

**Answer:**

Now suppose each $X_i$ is independent of variables $X_1 \dots X_{i-2}$ given $X_{i-1}$, *i.e.*,

$$X_i \perp (X_1 \dots X_{i-2}) \mid X_{i-1}$$

Could call this a "Markovian" dependence structure.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** Can we think of another assumption that may be "in between"?

**Answer:**

Now suppose each $X_i$ is independent of variables $X_1 \dots X_{i-2}$ given $X_{i-1}$, *i.e.*,

$$X_i \perp (X_1 \dots X_{i-2}) \mid X_{i-1}$$

Then:

$$\begin{aligned} p(x_1, x_2, \dots, x_n) &= p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2, x_1) \cdots p(x_n \mid x_{n-1}, \dots, x_1) \\ &= p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2, \cancel{x_1}) \cdots p(x_n \mid x_{n-1}, \cancel{\dots}, \cancel{x_1}) \\ &= p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2) \cdots p(x_n \mid x_{n-1}) \end{aligned}$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Image Example

**Question:** Can we think of another assumption that may be "in between"?

**Answer:**

Now suppose each $X_i$ is independent of variables $X_1 \dots X_{i-2}$ given $X_{i-1}$, *i.e.*,

$$X_i \perp (X_1 \dots X_{i-2}) \mid X_{i-1}$$

Then:

$$p(x_1, x_2, \dots, x_n) = p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2) \cdots p(x_n \mid x_{n-1})$$

$\Rightarrow$ In total, $2n - 1$ parameters.

$\Rightarrow$ Exponential reduction!

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Conditional Independence Assumptions

We can incorporate structure (simplifications) through conditional independence!

*I.e.*, conditional independence assumptions

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Conditional Independence Assumptions

We can incorporate structure (simplifications) through conditional independence!

*I.e.*, conditional independence assumptions

Bayesian Networks — the general idea:

- Represent a joint PDF (probability model) in terms of a set of conditional parameterizations.
- For each random variable $X_i$, specify a distribution $p(x_i \mid \mathbf{x}_{P_i})$ for a set of variables $\mathbf{x}_{P_i}$.
- Then write the joint parameterization as:

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid \mathbf{x}_{P_i})$$

(Need to guarantee this is a "legal" probability distribution → must correspond to some chain rule factorization with factors simplified due to assumed conditional independencies).

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Bayesian Networks

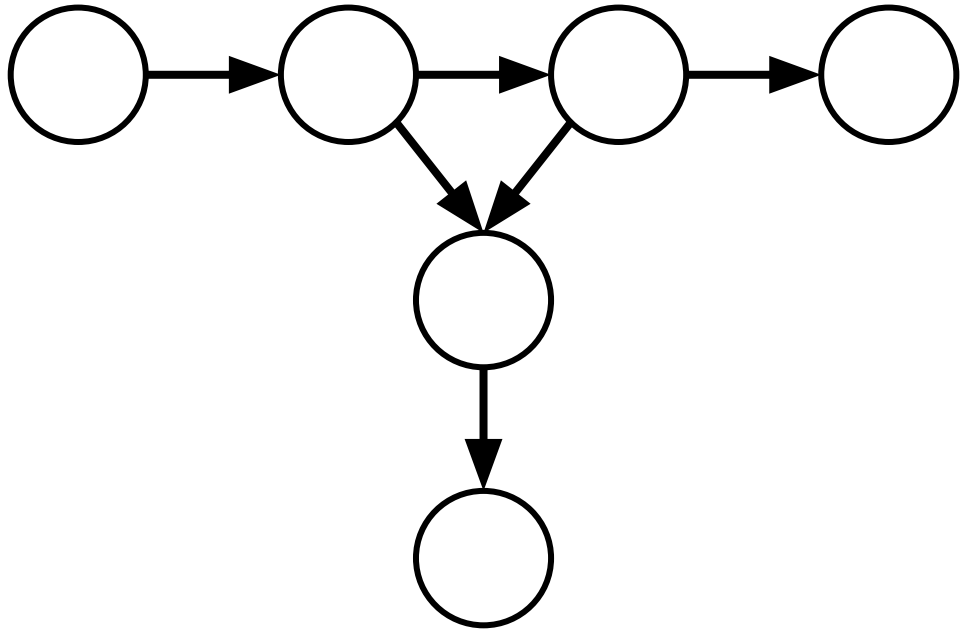
**Definition.**

# Bayesian Networks

**Definition**. A Bayesian Network is defined by a directed acyclic graph (DAG) $G = (V, E)$, where

1. There is one node $i \in V$ for each random variable $X_i$.

2. There is one conditional probability distribution per node, $p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$, specifying the variable's probability conditioned on its parents' values.

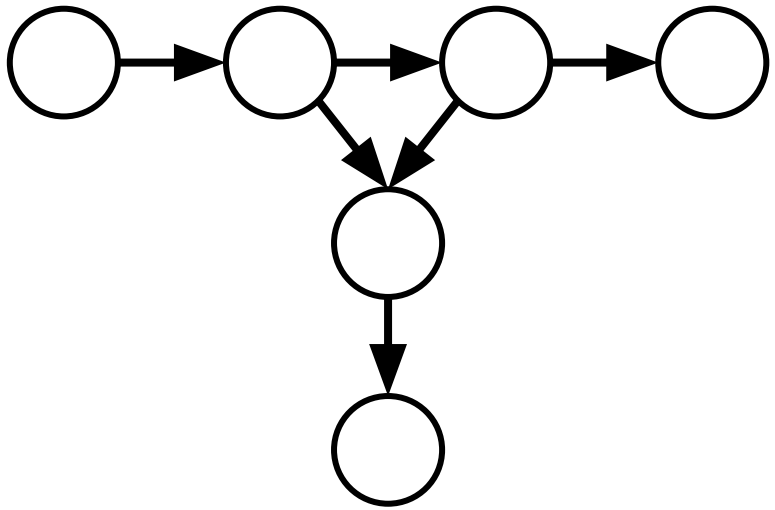


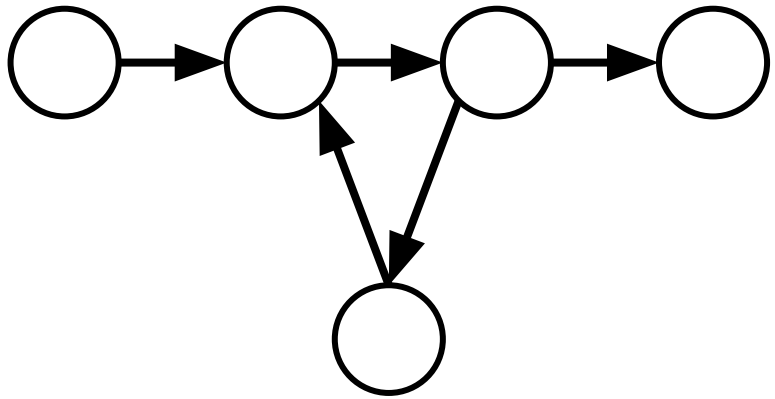
Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Bayesian Networks

**Note**.

Directed acyclic graph (DAG) vs. Not-a-DAG.





Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Bayesian Networks

The graph $G = (V, E)$ is called the structure of the Bayesian network.

This defines a joint distribution:

$$p(x_1, \ldots, x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Bayesian Networks

The graph $G = (V, E)$ is called the structure of the Bayesian network.

This defines a joint distribution:

$$p(x_1, \ldots, x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$$

**Claim:**

$p(x_1, \ldots, x_n)$ is a valid probability distribution (simplifies some chain rule rep'n), because of the ordering implied by the DAG.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# Bayesian Networks

The graph $G = (V, E)$ is called the structure of the Bayesian network.

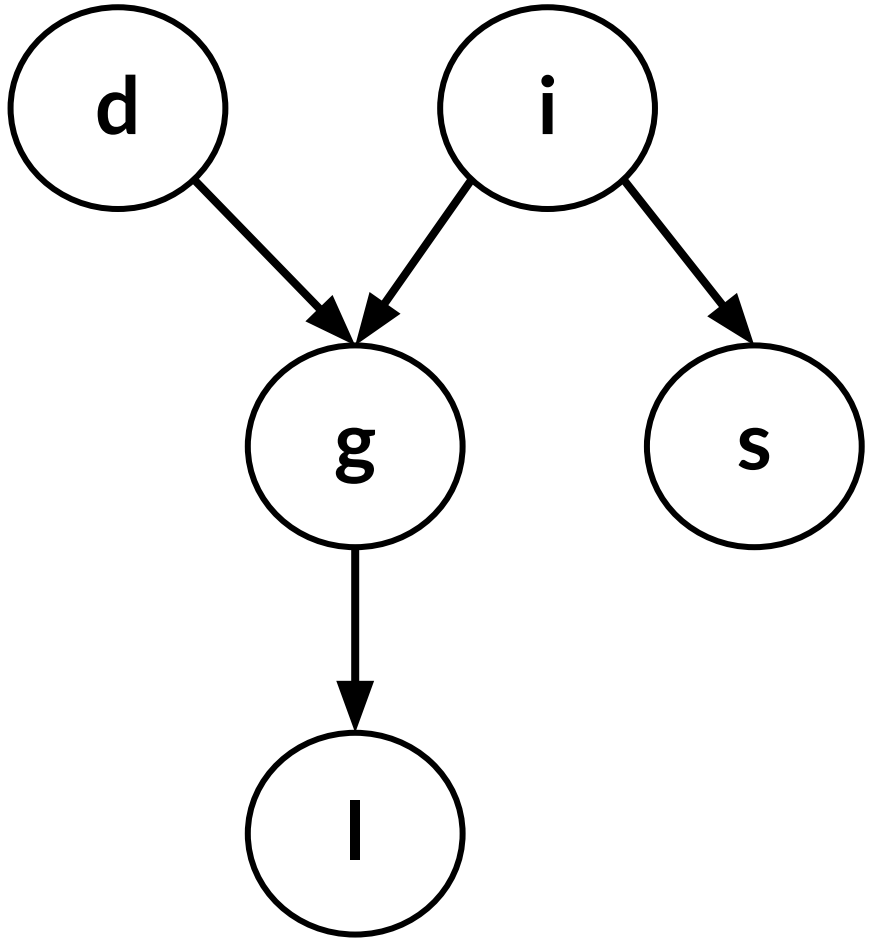
This defines a joint distribution:

$$p(x_1, \ldots, x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$$

**Economical representation** $\Rightarrow$ Number of parameters is exponential in $\max_{i \in V} |\mathrm{Pa}(i)|$, not in $|V|$.

# **Bayesian Networks** — Example

Consider the following Bayesian network:

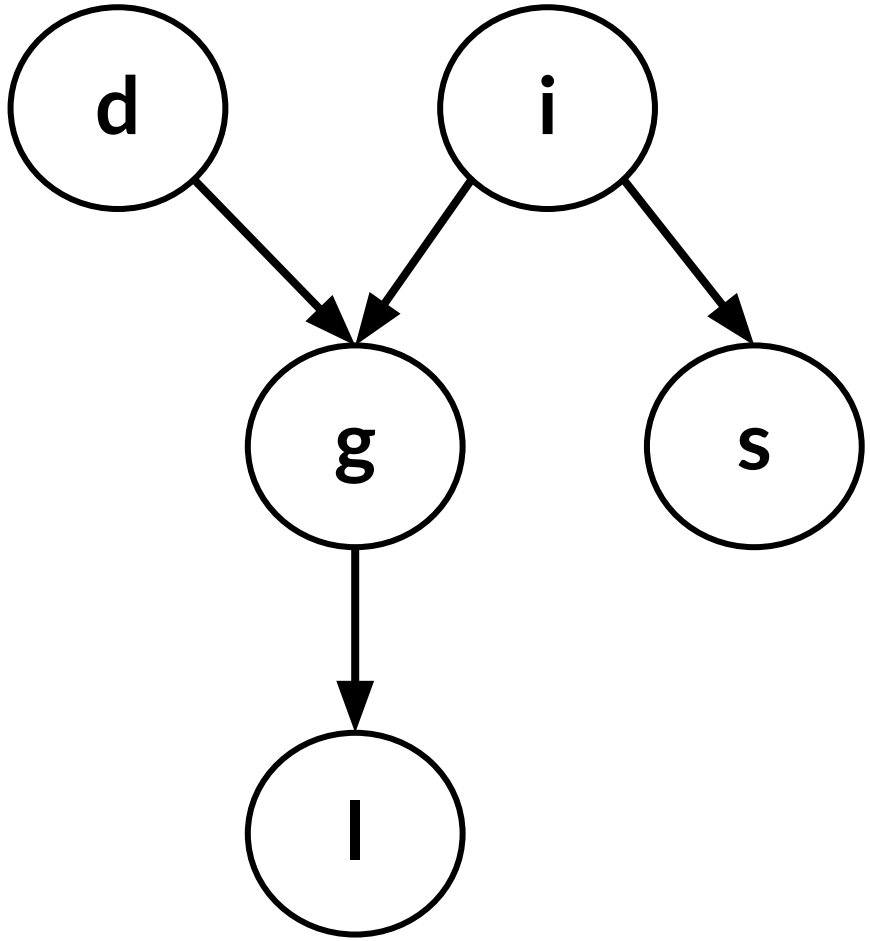



Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Example

Consider the following Bayesian network:

(A model of student grades and recommendation letters)



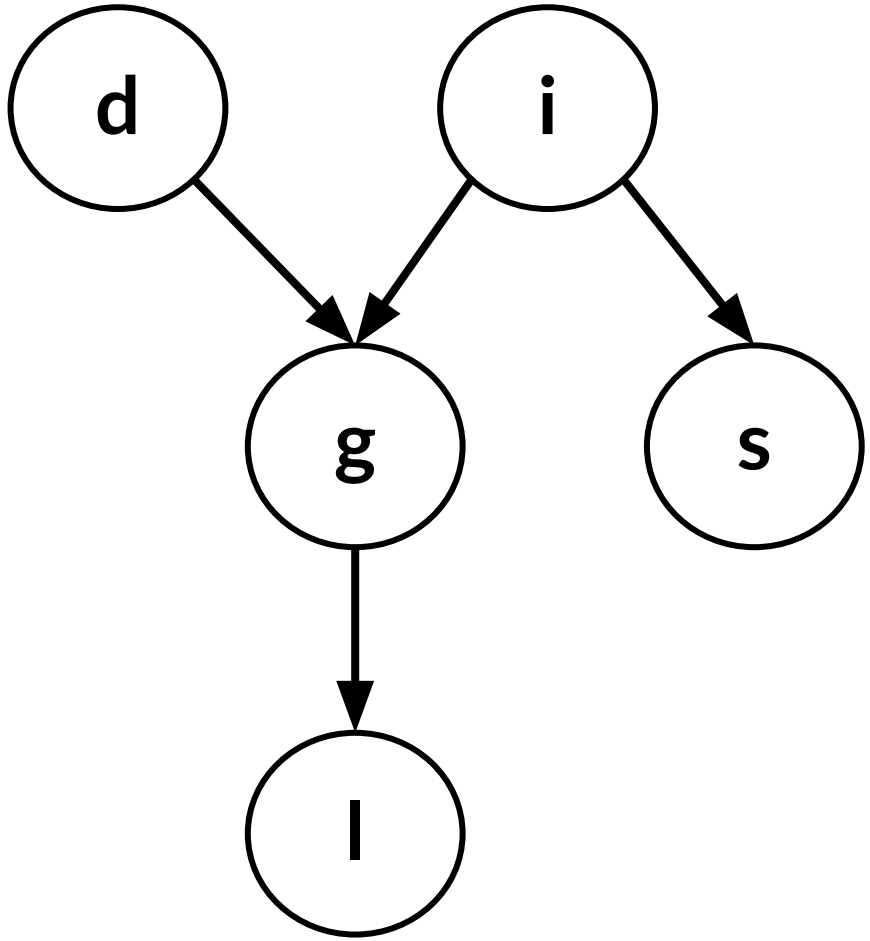

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Example

Consider the following Bayesian network:

(A model of student grades and recommendation letters)

- d - Class difficulty.
- i - Student's intelligence.
- g - Student's grade in the class.
- s - Student's SAT test score.
- l - Professor's letter of recommendation.
  (good vs bad letter)

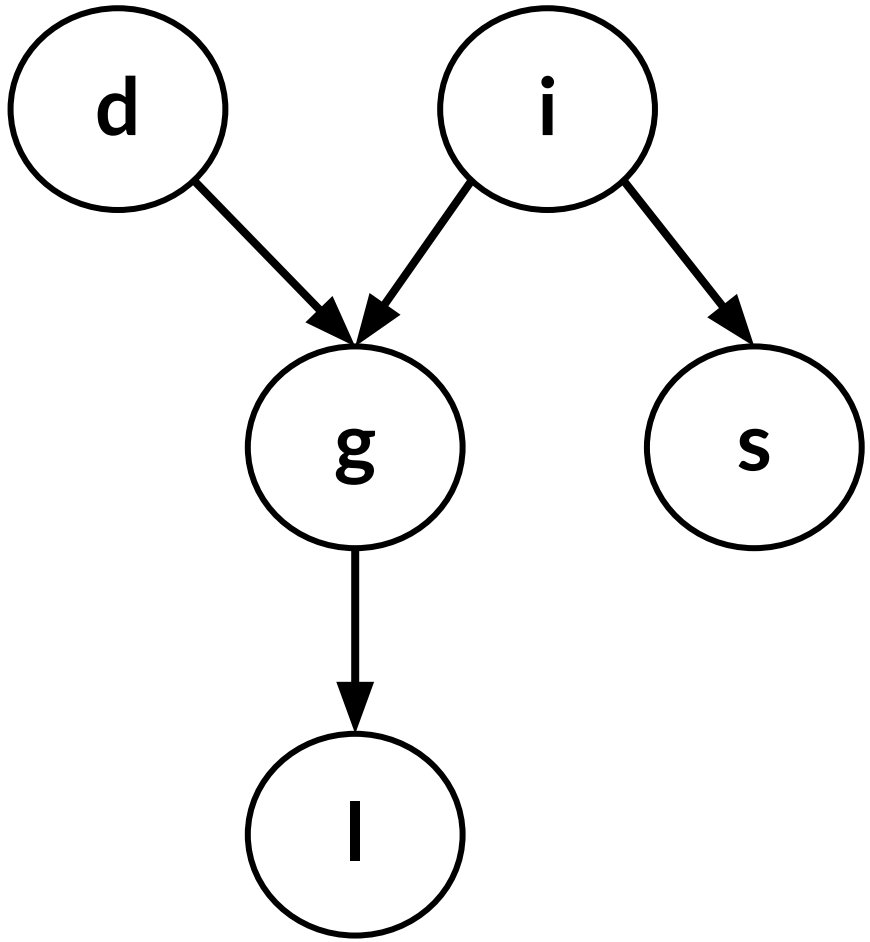



Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Example

Think of it as a generative process that generates data from top to bottom

(*i.e.*, from parents to children)




Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Example

Think of it as a generative process that generates data from top to bottom

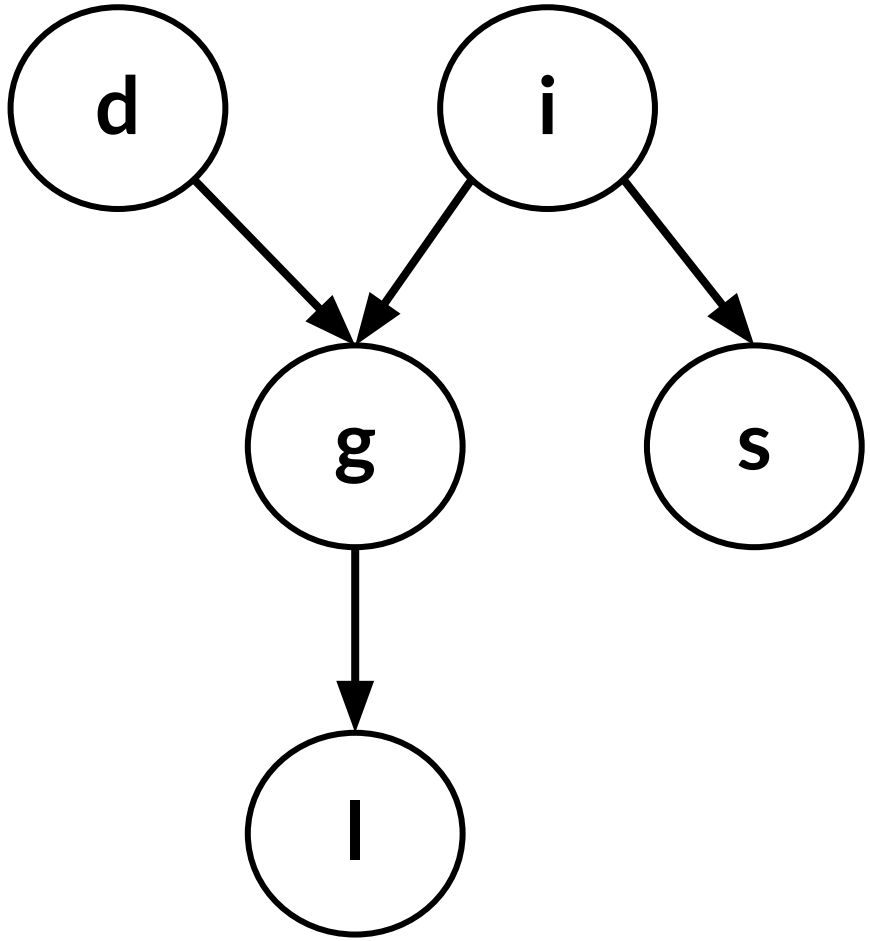
(*i.e.*, from parents to children)

Namely:

Class difficulty d and student's intelligence i → student's grade g.

Student's grade g → letter of recommendation l.

Student's intelligence → SAT score s

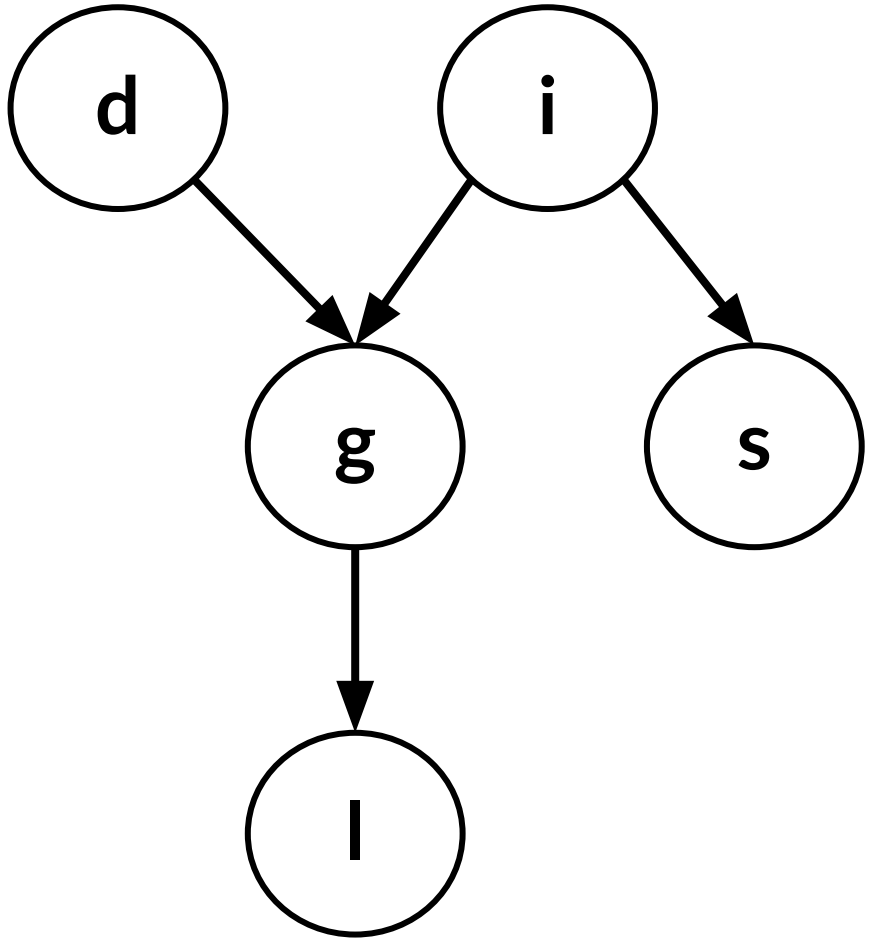



Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Example

How can we parameterize these distributions?

*I.e.*, "define" or "specify"




Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Example

How can we parameterize these distributions?

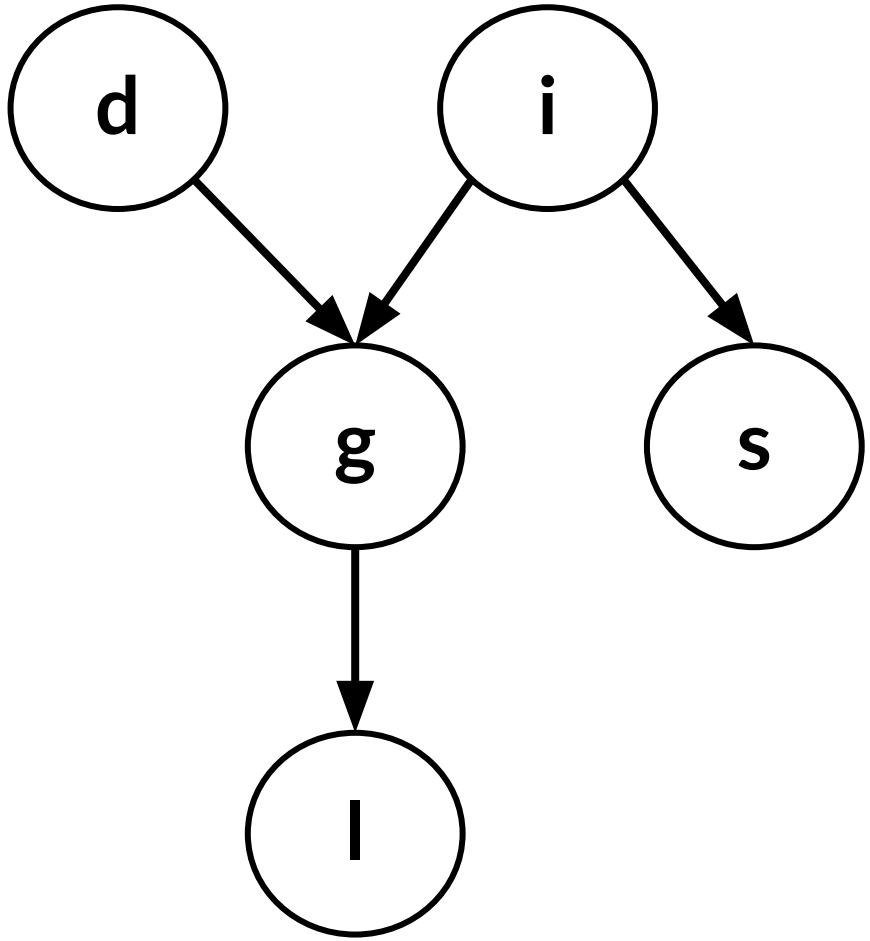
Suppose

d, i, s, l — binary random variables

- d - easy vs hard class.
- i - intelligent vs not-as-intelligent.
- s - good SAT score vs bad SAT score.
- l - good letter of rec vs bad letter of rec.

g — has three values.

- g - good grade vs average grade vs bad grade.




Source: Stefano Ermon, Deep Generative Models (CS236) Class

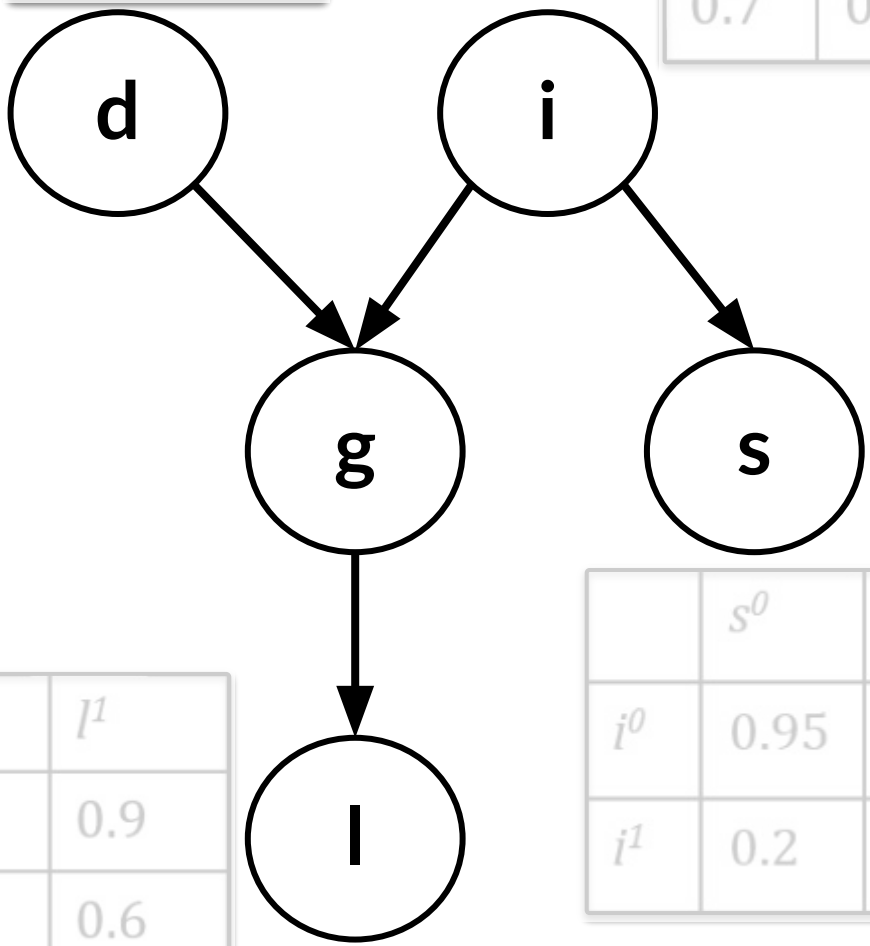
# **Bayesian Networks** — Example

Can define conditional probability distributions!

(In this case — tables).

| $d^0$ | $d^1$ |
|---|---|
| 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 |

Source: Stefano Ermon, Deep Generative Models (CS236) Class

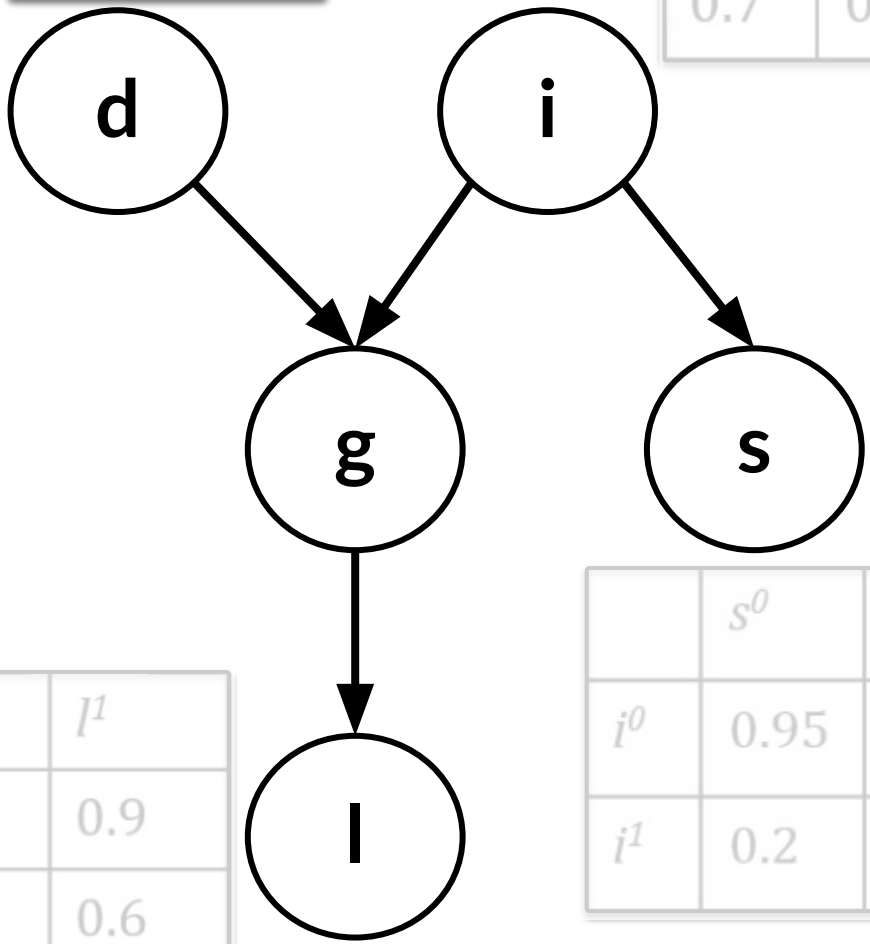
# **Bayesian Networks** — Example

Can define conditional probability distributions!

(In this case — tables).

| $d^0$ | $d^1$ |
|---|---|
| 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 |

Source: Stefano Ermon, Deep Generative Models (CS236) Class

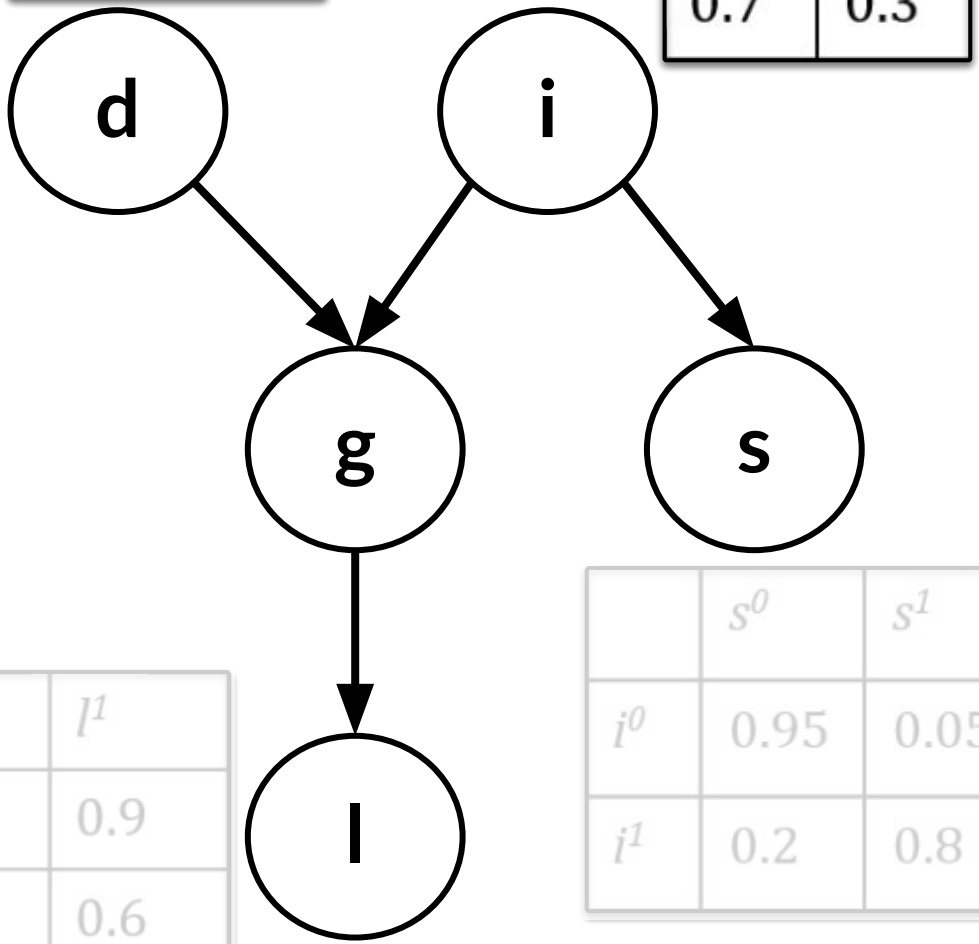
# **Bayesian Networks** — Example

Can define conditional probability distributions!

(In this case — tables).

| $d^0$ | $d^1$ |
|---|---|
| 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 |

Source: Stefano Ermon, Deep Generative Models (CS236) Class

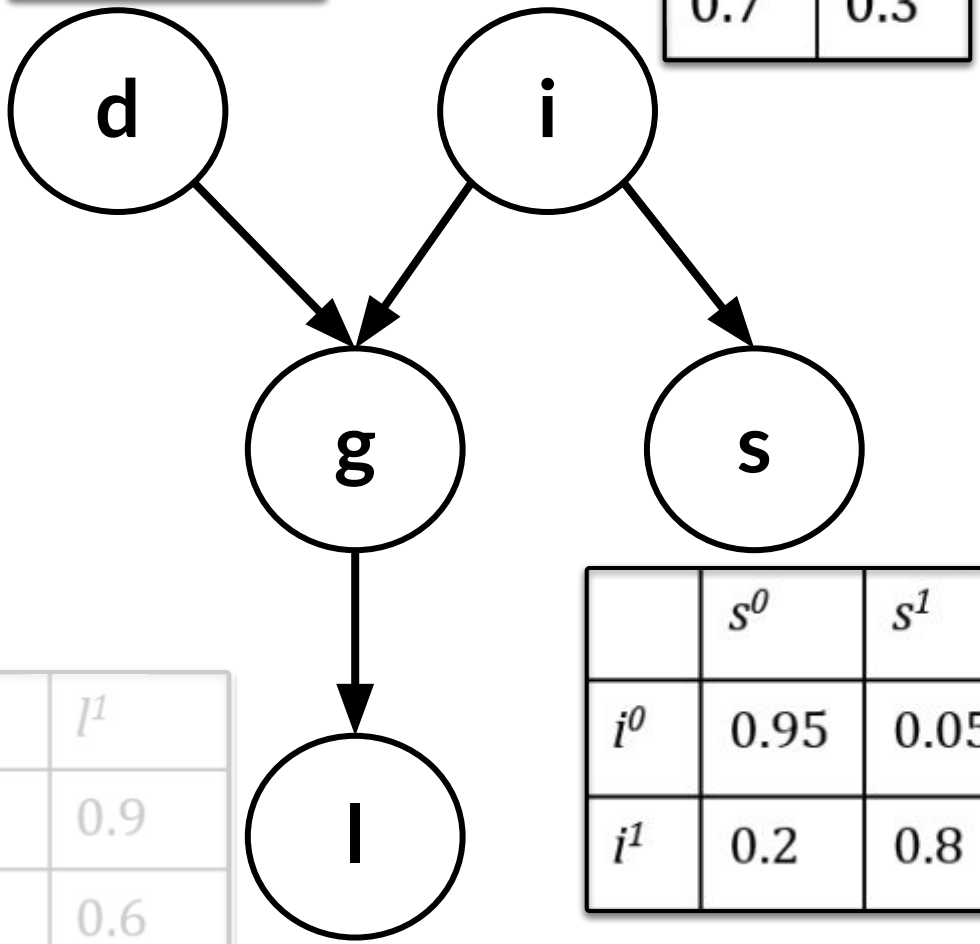
# **Bayesian Networks** — Example

Can define conditional probability distributions!

(In this case — tables).

| $d^0$ | $d^1$ |
|---|---|
| 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 |

Source: Stefano Ermon, Deep Generative Models (CS236) Class

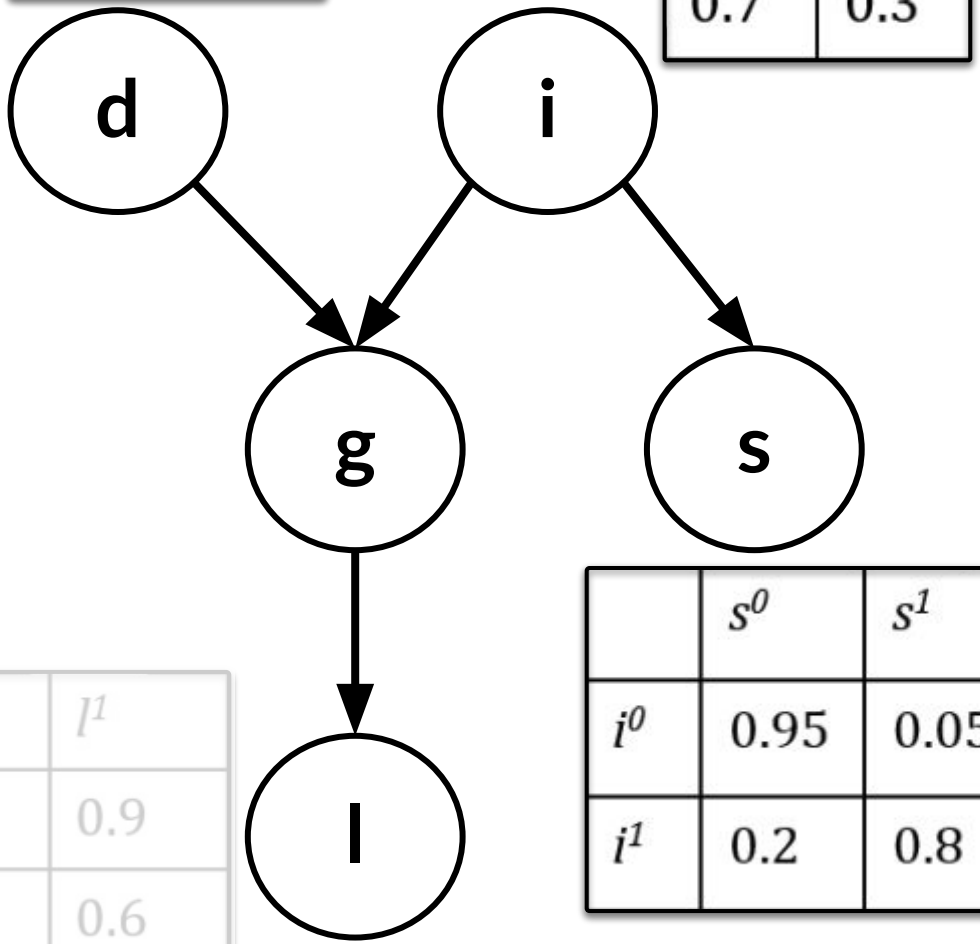
# **Bayesian Networks** — Example

Can define conditional probability distributions!

(In this case — tables).

| $d^0$ | $d^1$ |
|---|---|
| 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 |

Source: Stefano Ermon, Deep Generative Models (CS236) Class

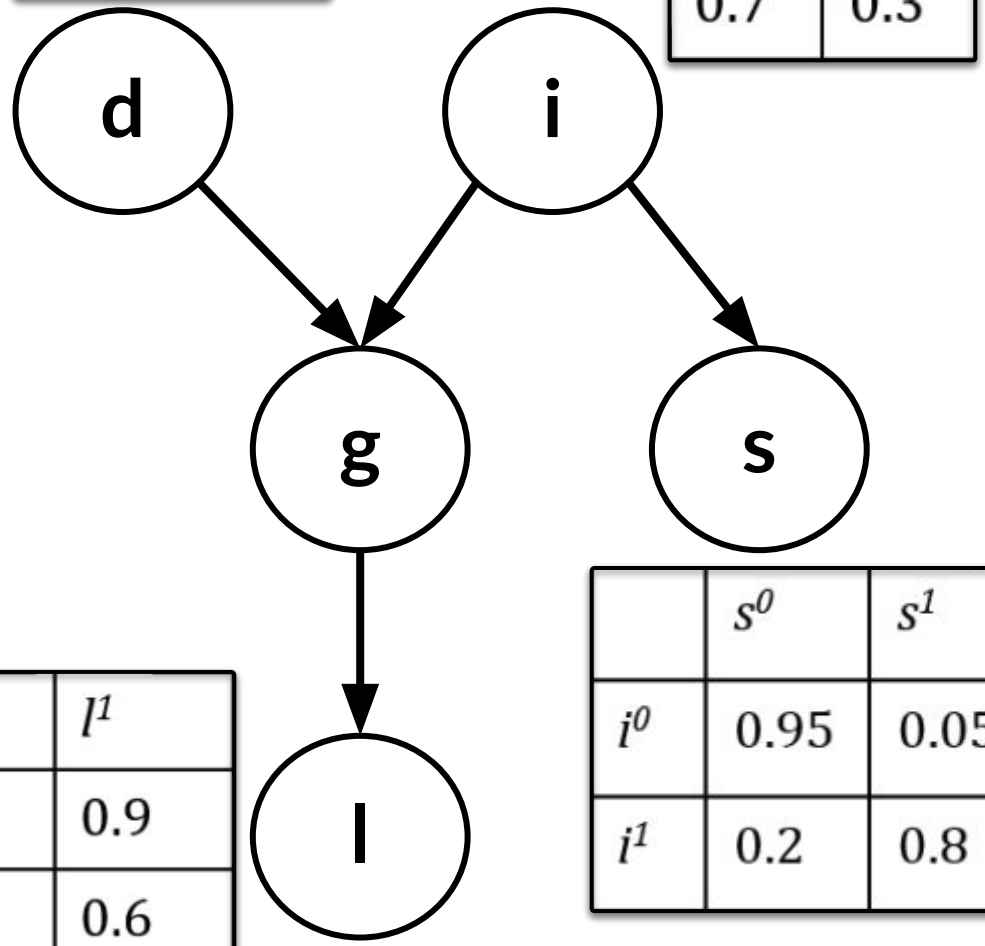
# **Bayesian Networks** — Example

Can define conditional probability distributions!

(In this case — tables).

| $d^0$ | $d^1$ |
|---|---|
| 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 |

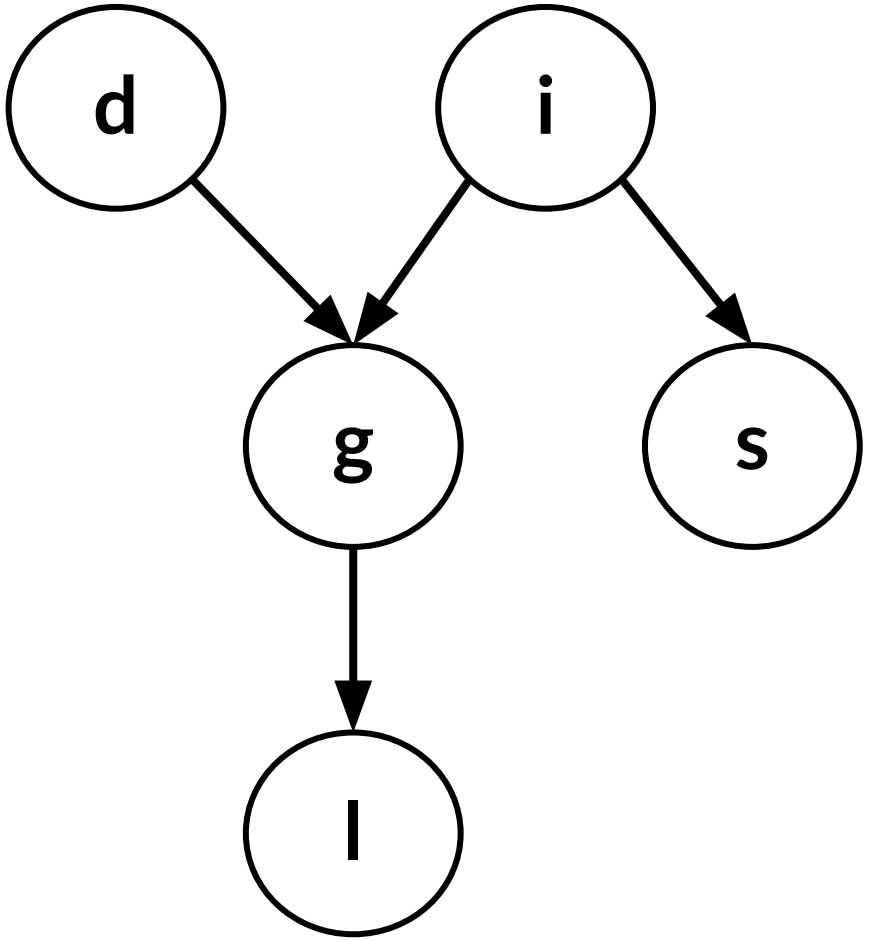
| | $s^0$ | $s^1$ |
|---|---|---|
| $i^0$ | 0.95 | 0.05 |
| $i^1$ | 0.2 | 0.8 |

| | $l^0$ | $l^1$ |
|---|---|---|
| $g^1$ | 0.1 | 0.9 |
| $g^2$ | 0.4 | 0.6 |
| $g^3$ | 0.99 | 0.01 |

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Example

What is the joint distribution?



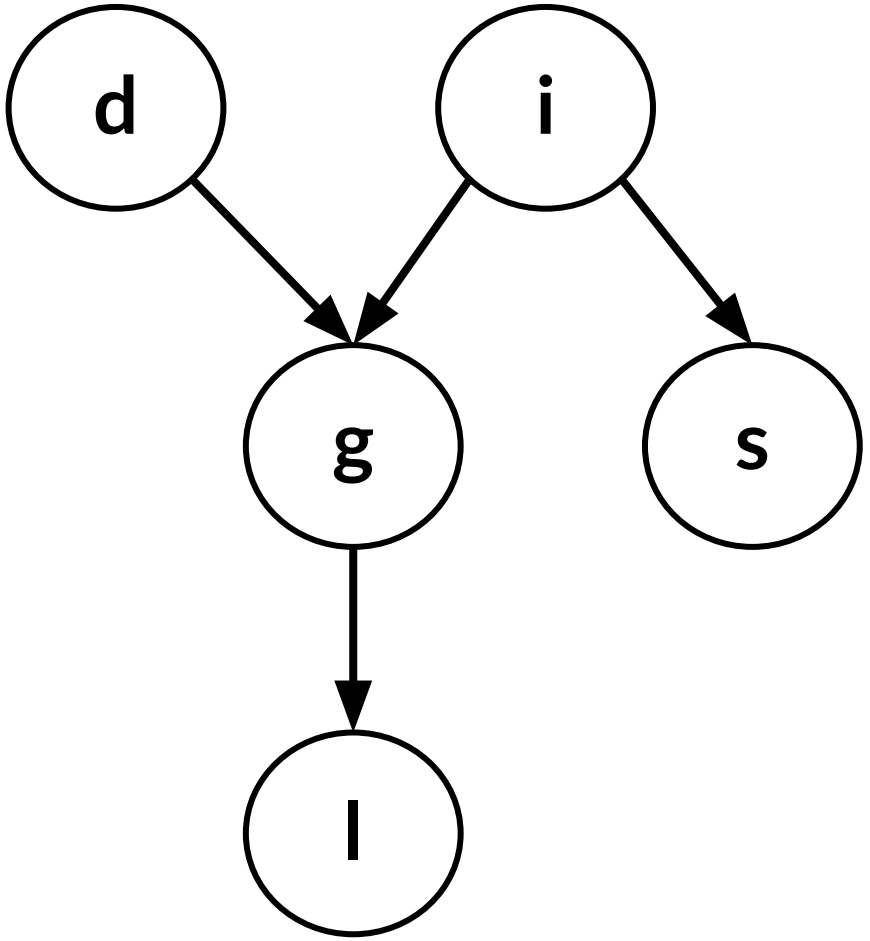

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Example

What is the joint distribution?

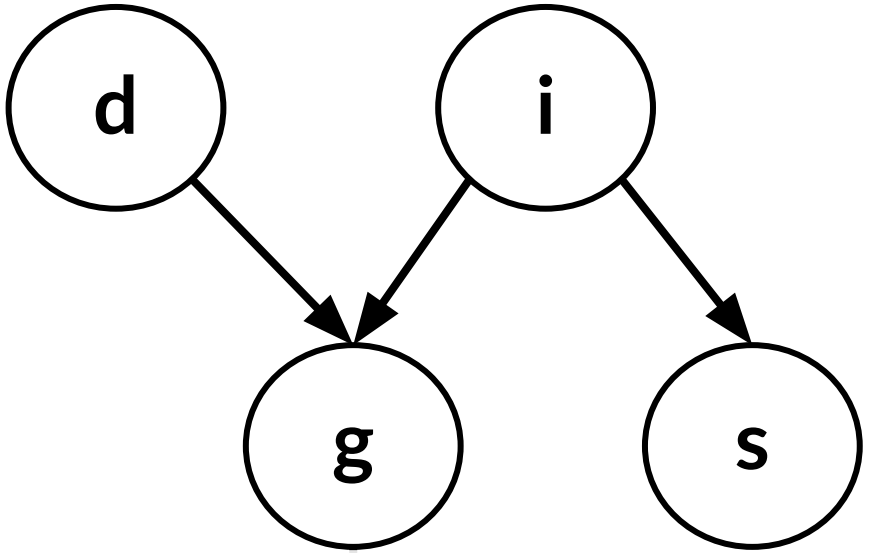
$$p(x_1, \ldots, x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$$

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




Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Example




Joint distribution (re-written):

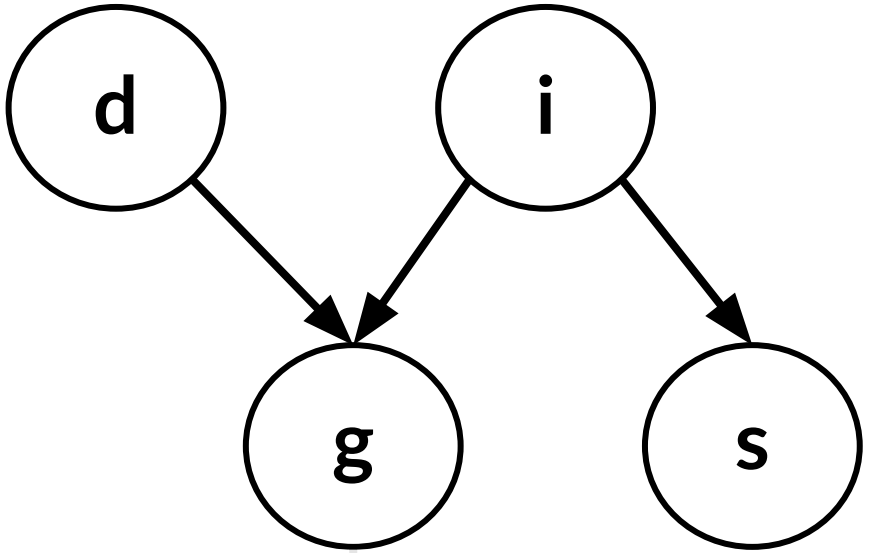
$$p(d, i, g, s, l) = p(d)\ p(i)\ p(g \mid i, d)\ p(s \mid i)\ p(l \mid g)$$

By chain rule (without any assumptions), could write joint distribution as:

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

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Example




Joint distribution (re-written):

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

By chain rule (without any assumptions), could write joint distribution as:

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

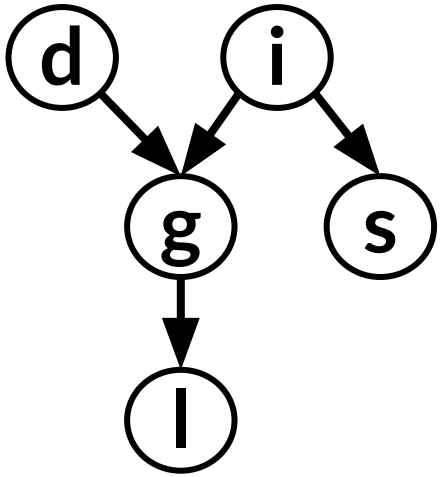
Thus, we are assuming the following conditional independencies:

$$d \perp i \qquad s \perp \{d, g\} \mid i \qquad l \perp \{i, d, s\} \mid g$$

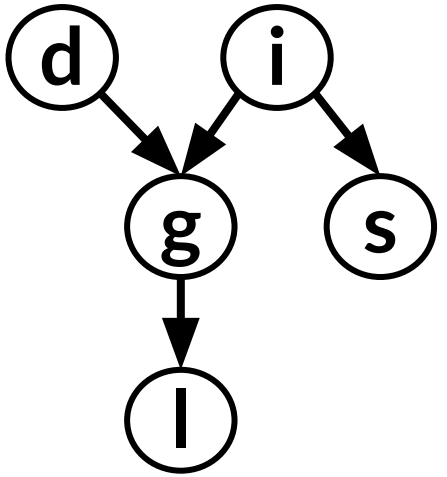
Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Generative Process View

Generative process view.



# **Bayesian Networks** — Generative Process View

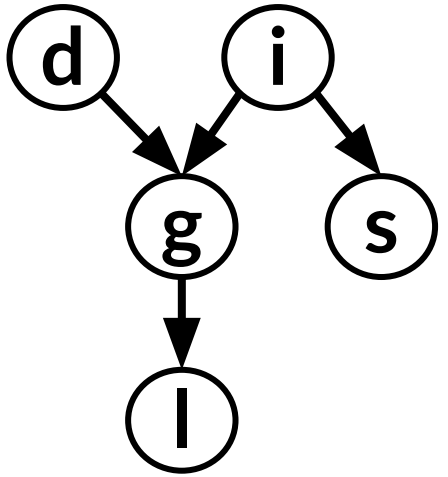



Generative process view.

View a Bayes net as “generating” from root to leaf variables.

- *I.e.*, view it as defining a *sampling procedure* for all variables in the graph.

# **Bayesian Networks** — Generative Process View




Generative process view.

View a Bayes net as "generating" from root to leaf variables.

- *I.e.*, view it as defining a *sampling procedure* for all variables in the graph.

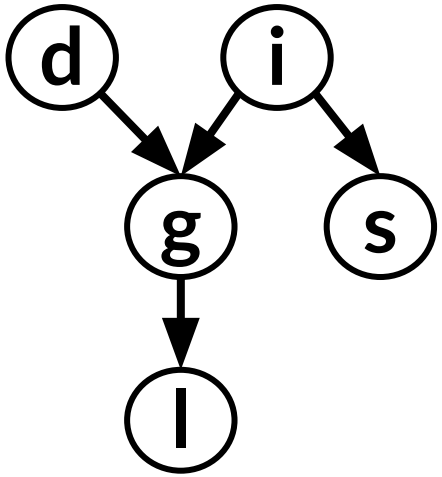
Iteratively sample from conditional probability distribution according to a topological sort of the nodes.

**Topological Sort** (of a directed graph)**:**

A linear ordering of the vertices such that for every directed edge $(u, v)$ from vertex $u$ to vertex $v$, $u$ comes before $v$ in the ordering

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Generative Process View




Generative process view.

View a Bayes net as "generating" from root to leaf variables.

- *I.e.*, view it as defining a *sampling procedure* for all variables in the graph.

Iteratively sample from conditional probability distribution according to a topological sort of the nodes.

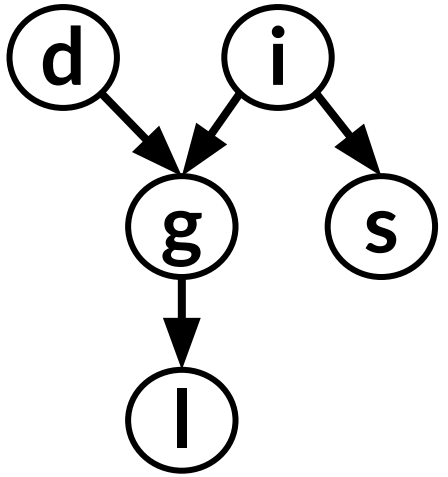
$$d \sim p(d) \qquad i \sim p(i)$$

$$g \sim p(g \mid i, d) \qquad s \sim p(s \mid i)$$

$$l \sim p(l \mid g)$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Generative Process View




Generative process view.

View a Bayes net as "generating" from root to leaf variables.

- *I.e.*, view it as defining a *sampling procedure* for all variables in the graph.

Iteratively sample from conditional probability distribution according to a topological sort of the nodes.

$$
\begin{aligned}
d &\sim p(d) \\
i &\sim p(i) \\
g &\sim p(g \mid i, d) \\
s &\sim p(s \mid i) \\
l &\sim p(l \mid g)
\end{aligned}
$$

Often called the **"generative process" view** of the PGM/joint PDF.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Summary

# **Bayesian Networks** — Summary

Bayesian network is define by a tuple $(G = \{V, E\}, P)$ where $P$ is specified as a set of local conditional probability distributions (CPDs) associated with nodes $V$

⇒ Efficient representation using a graph-based data structure.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Summary

Bayesian network is define by a tuple $(G = \{V, E\}, P)$ where $P$ is specified as a set of local conditional probability distributions (CPDs) associated with nodes $V$

⇒ Efficient representation using a graph-based data structure.

Computing joint probability of any assignment obtained by multiplying CPDs.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Summary

Bayesian network is define by a tuple $(G = \{V, E\}, P)$ where $P$ is specified as a set of local conditional probability distributions (CPDs) associated with nodes $V$

⇒ Efficient representation using a graph-based data structure.

Computing joint probability of any assignment obtained by multiplying CPDs.

Can sample from joint by sampling from CPDs according to DAG ordering.

# **Bayesian Networks** — Summary

Bayesian network is define by a tuple $(G = \{V, E\}, P)$ where $P$ is specified as a set of local conditional probability distributions (CPDs) associated with nodes $V$

⇒ Efficient representation using a graph-based data structure.

Computing joint probability of any assignment obtained by multiplying CPDs.

Can sample from joint by sampling from CPDs according to DAG ordering.

Can identify some conditional independencies by looking at graph properties.

# **Bayesian Networks** — Summary

Bayesian network is define by a tuple $(G = \{V, E\}, P)$ where $P$ is specified as a set of local conditional probability distributions (CPDs) associated with nodes $V$

⇒ Efficient representation using a graph-based data structure.

Computing joint probability of any assignment obtained by multiplying CPDs.

Can sample from joint by sampling from CPDs according to DAG ordering.

Can identify some conditional independencies by looking at graph properties.

Nodes in graph can correspond to random variables or random vectors.

# **Bayesian Networks** — Summary

Bayesian network is define by a tuple $(G = \{V, E\}, P)$ where $P$ is specified as a set of local conditional probability distributions (CPDs) associated with nodes $V$

⇒ Efficient representation using a graph-based data structure.

Computing joint probability of any assignment obtained by multiplying CPDs.

Can sample from joint by sampling from CPDs according to DAG ordering.

Can identify some conditional independencies by looking at graph properties.

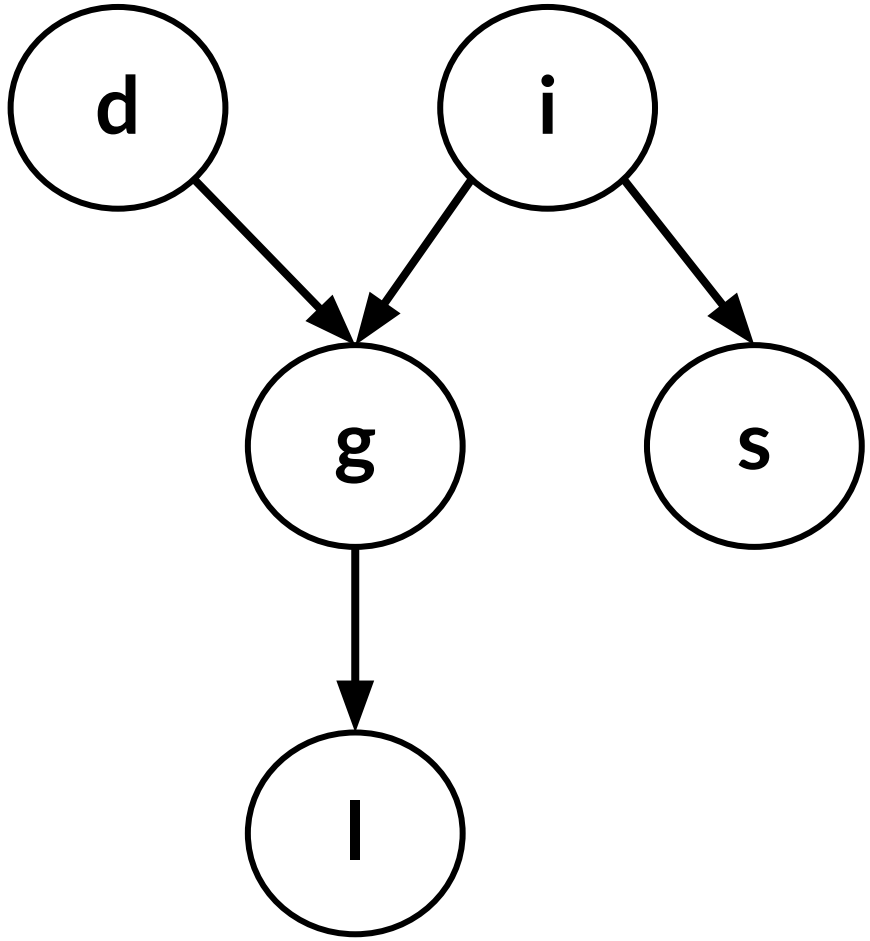
Nodes in graph can correspond to random variables or random vectors.

**Next**: observed vs. latent variables; plate notation; famous Bayesian networks.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Bayesian Networks** — Observed vs Latent

Back to the running example Bayesian network…



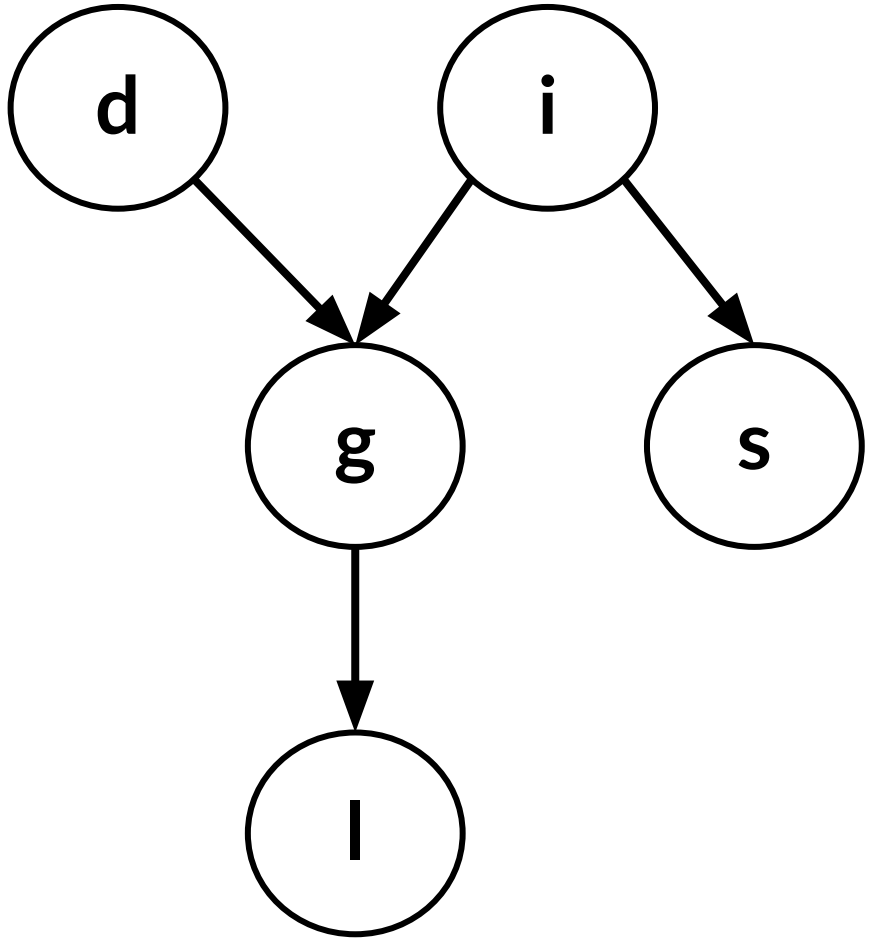
# **Bayesian Networks** — Observed vs Latent

Observed Variables

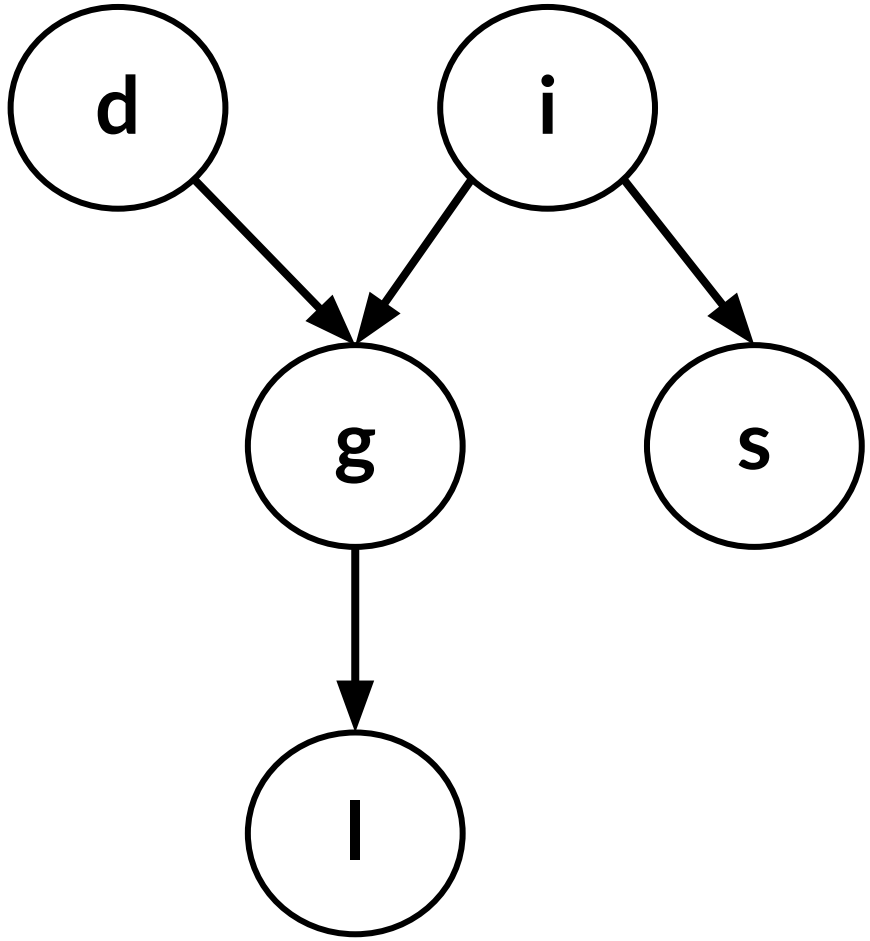
- Variables that we observe (as samples from joint PDF)

Latent Variables

- Variables in the model that are unobserved.
- ⇒ but we may want to infer them given observed variables!



# **Bayesian Networks** — Observed vs Latent




Observed Variables

- Variables that we observe (as samples from joint PDF)

Latent Variables

- Variables in the model that are unobserved.
- ⇒ but we may want to infer them given observed variables!

$$p(d \mid g, s, l)$$

# **Bayesian Networks** — Observed vs Latent

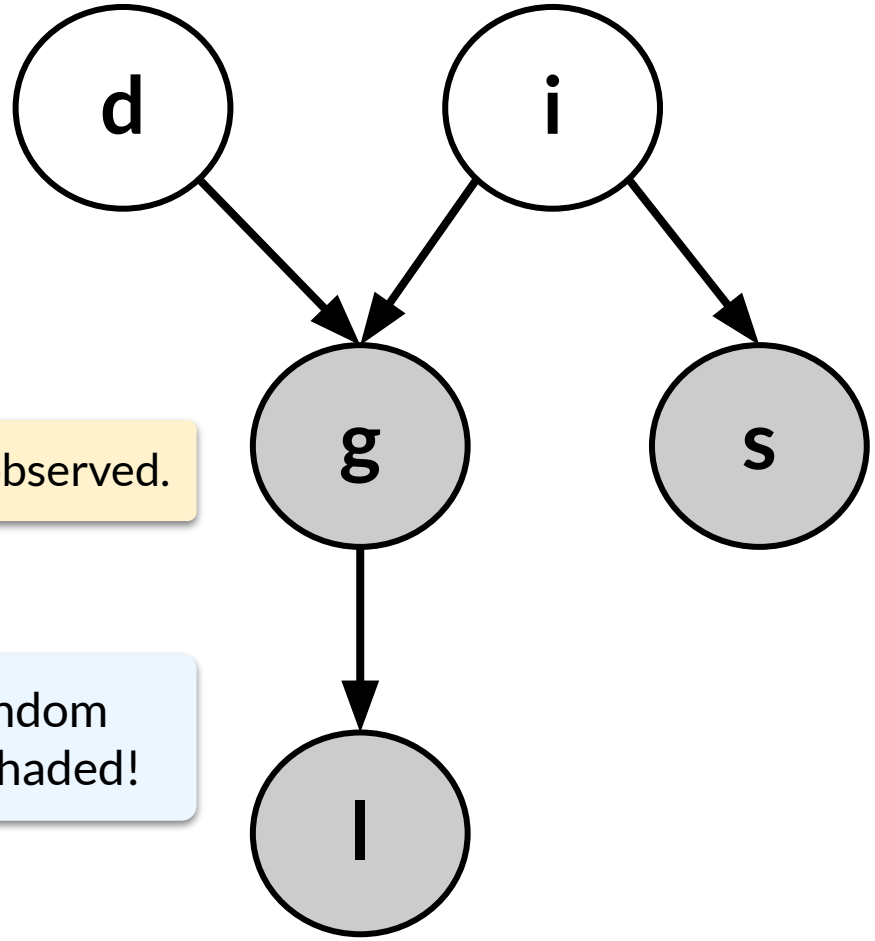
## Observed Variables

- Variables that we observe (as samples from joint PDF)

## Latent Variables

- Variables in the model that are unobserved.
- ⇒ but we may want to infer them given observed variables!

$$p(d \mid g, s, l)$$




*E.g.*, $\{d, i\}$ latent, $\{g, s, l\}$ observed.

Observed random variables are shaded!

# **Bayesian Networks** — Observed vs Latent

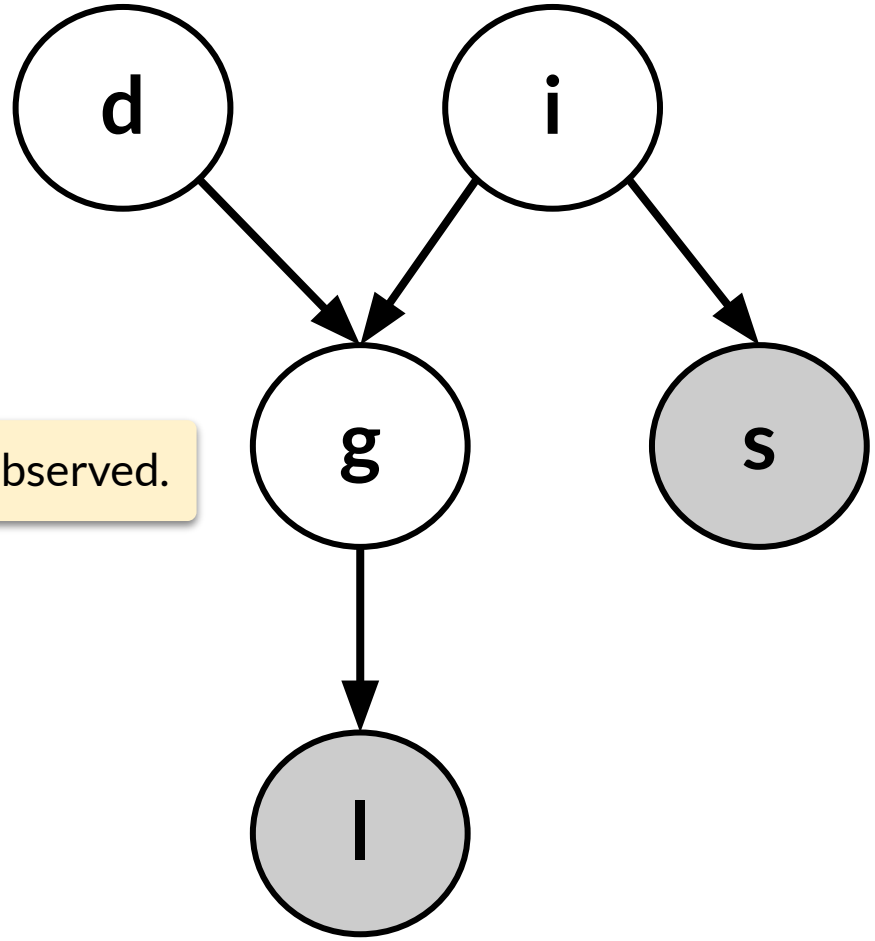
## Observed Variables

- Variables that we observe (as samples from joint PDF)

## Latent Variables

- Variables in the model that are unobserved.
- ⇒ but we may want to infer them given observed variables!

$$p(g \mid s, l)$$




*E.g.*, $\{d, i, g\}$ latent, $\{s, l\}$ observed.

# **Bayesian Networks** — Observed vs Latent

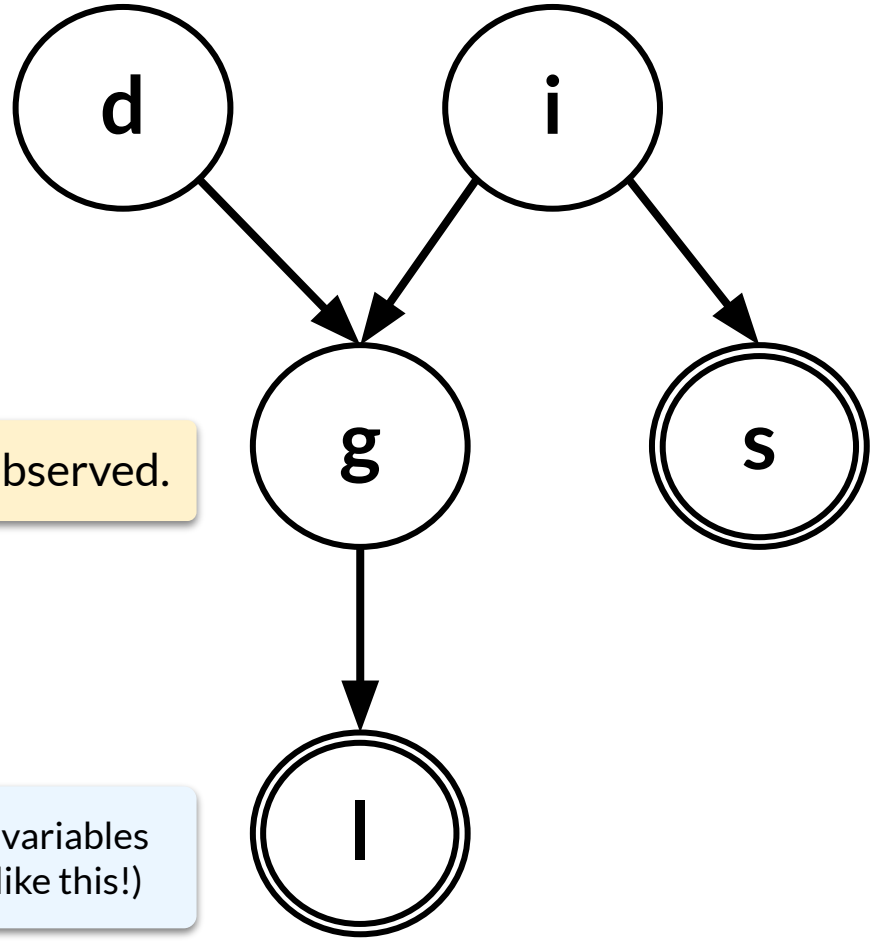
## Observed Variables

- Variables that we observe (as samples from joint PDF)

## Latent Variables

- Variables in the model that are unobserved.
- ⇒ but we may want to infer them given observed variables!

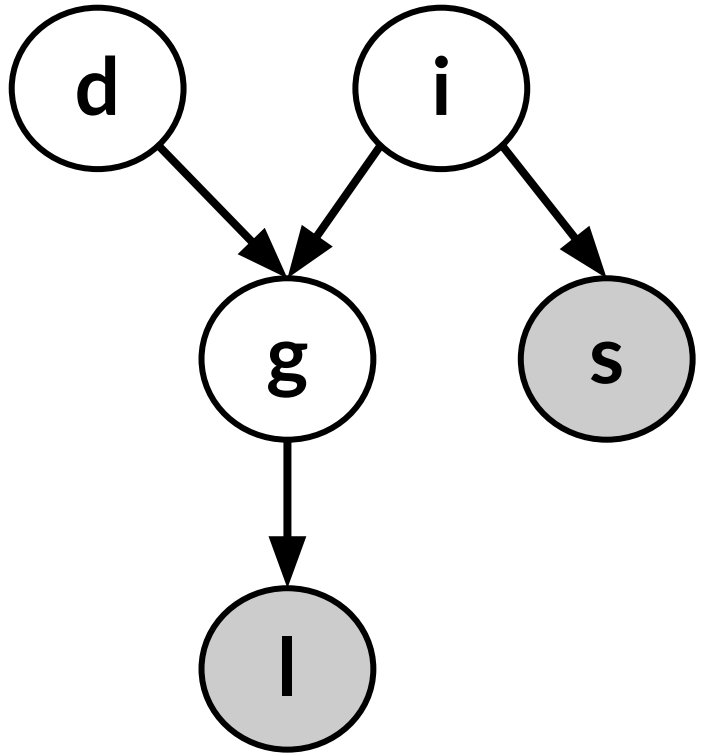
$$p(g \mid s, l)$$




*E.g.*, $\{d, i, g\}$ latent, $\{s, i\}$ observed.

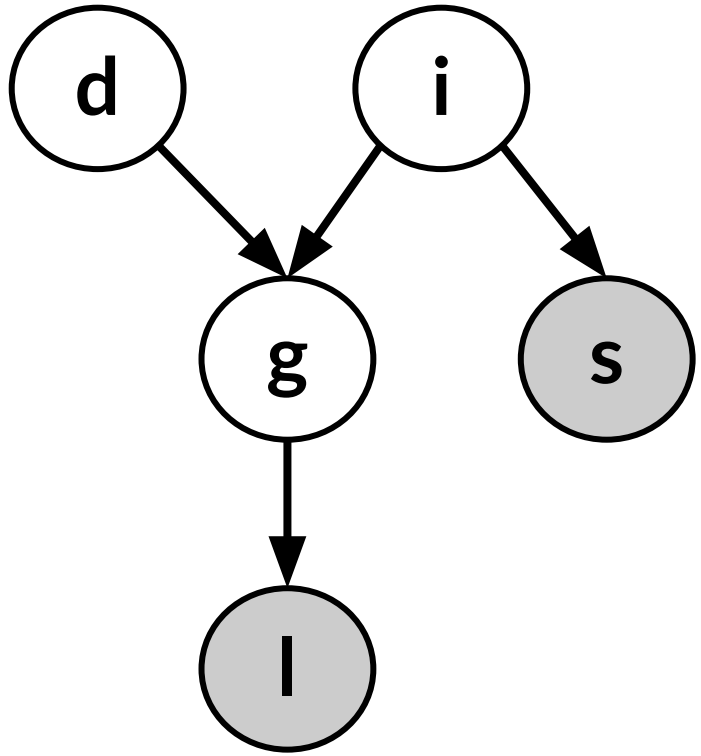
(Observed random variables sometimes shown like this!)

# **Bayesian Networks** — Plate Notation



# **Bayesian Networks** — Plate Notation

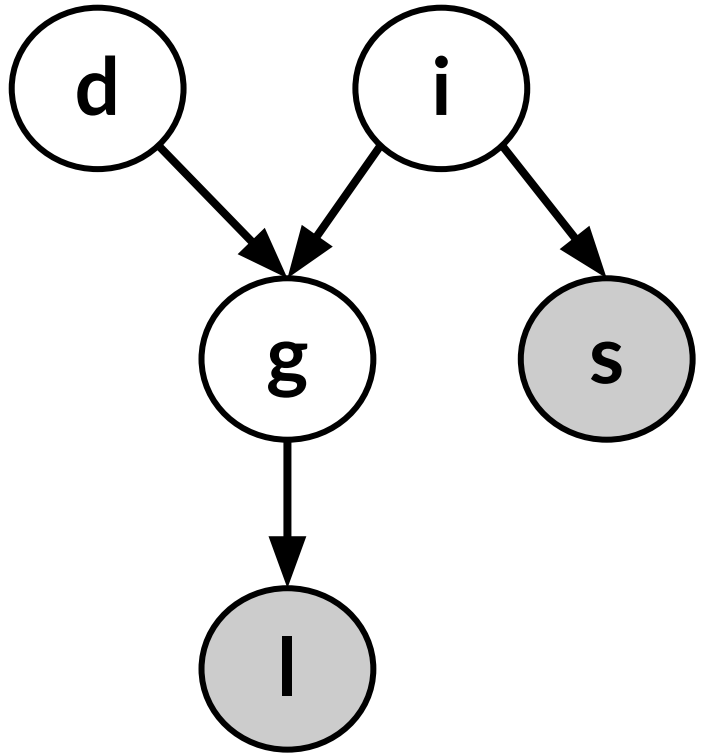
Plate notation is a “visual language” for describing more-complex Bayesian networks.



# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

⇒ Allows us to construct more-complex BNs with compressed notation!



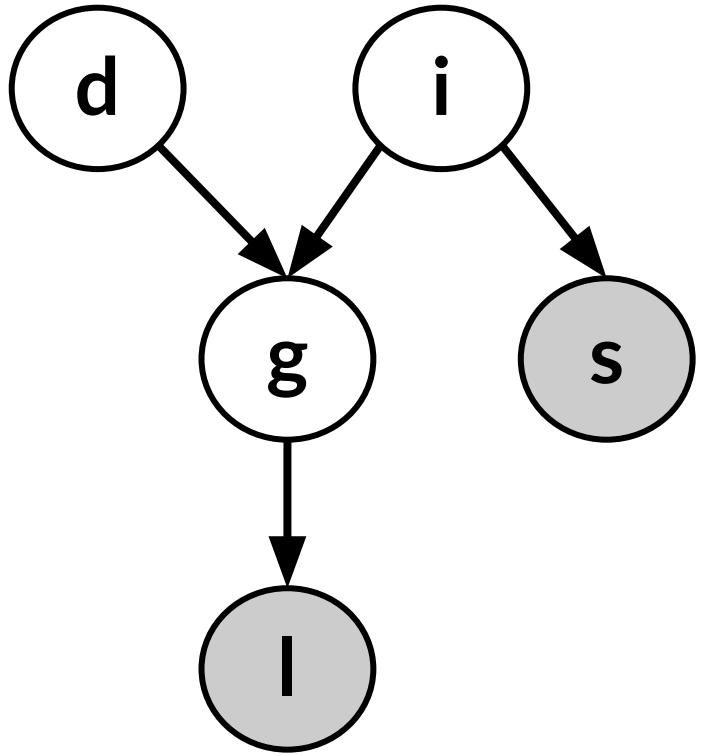
# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

⇒ Allows us to construct more-complex BNs with compressed notation!

***E.g.*, what if we want to model 2 students?**

(Each taking a separate class, an SAT test, and getting a letter-of-rec).



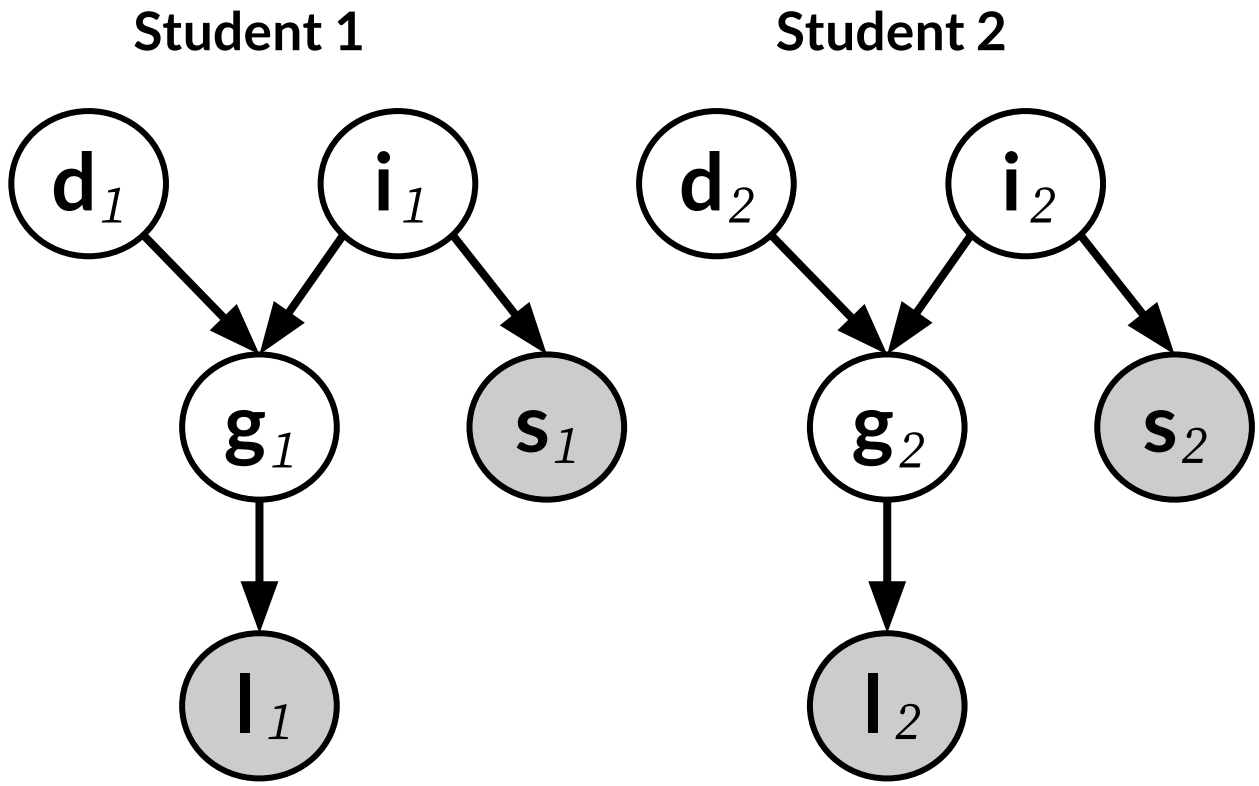
# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

⇒ Allows us to construct more-complex BNs with compressed notation!

***E.g.*, what if we want to model 2 students?**

(Each taking a separate class, an SAT test, and getting a letter-of-rec).



# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

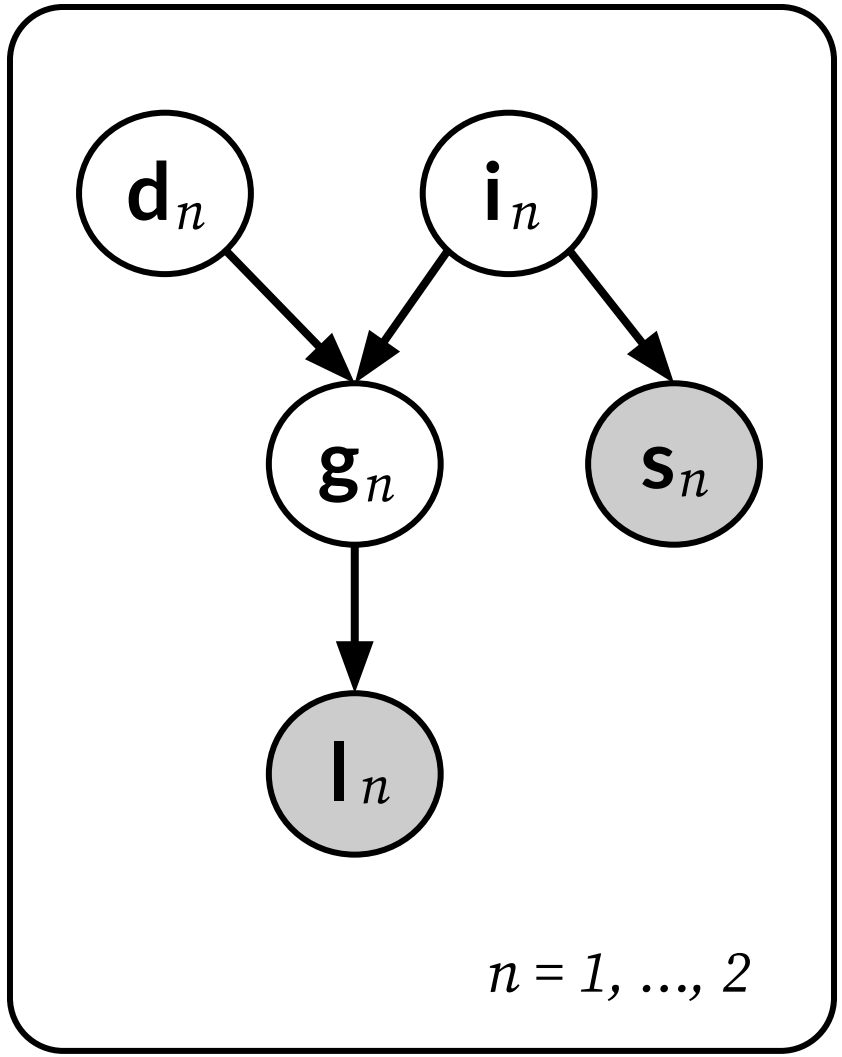
⇒ Plate with indices denotes sets of random variables.

# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

⇒ Plate with indices denotes sets of random variables.

Two (independent) sets of random variables.

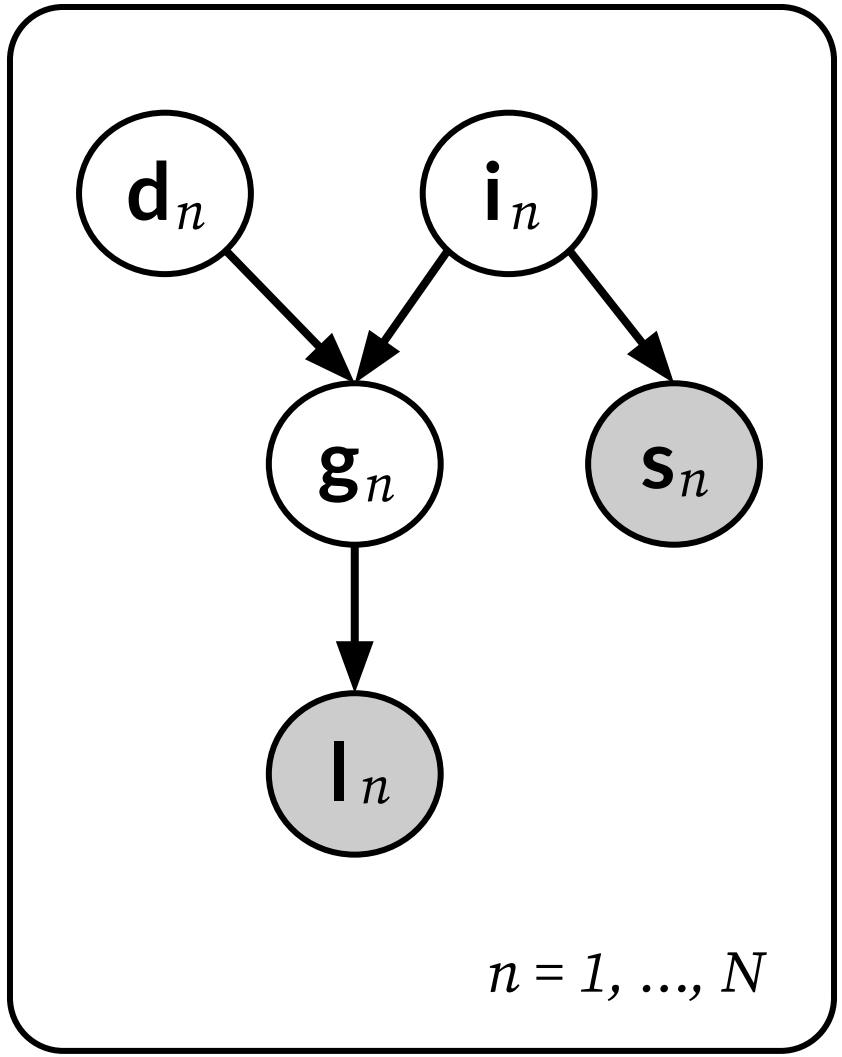


# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

⇒ Plate with indices denotes sets of random variables.

$N$ (independent) sets of random variables.

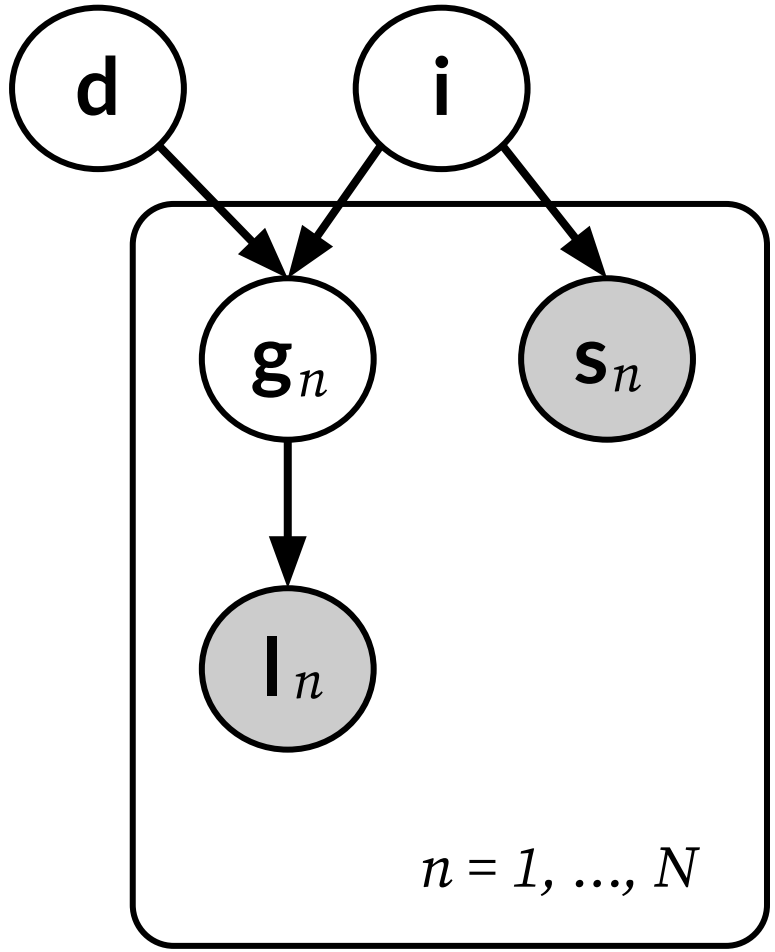


# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

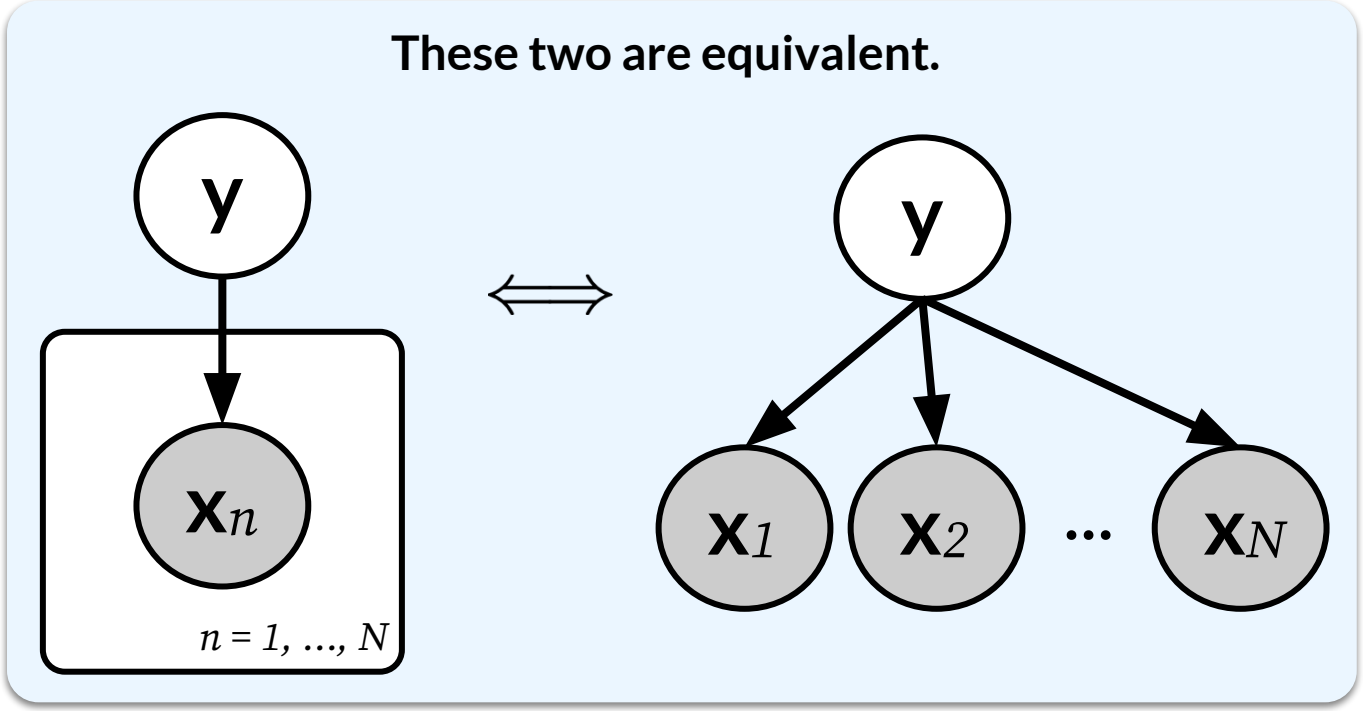
**What does this plate mean?**

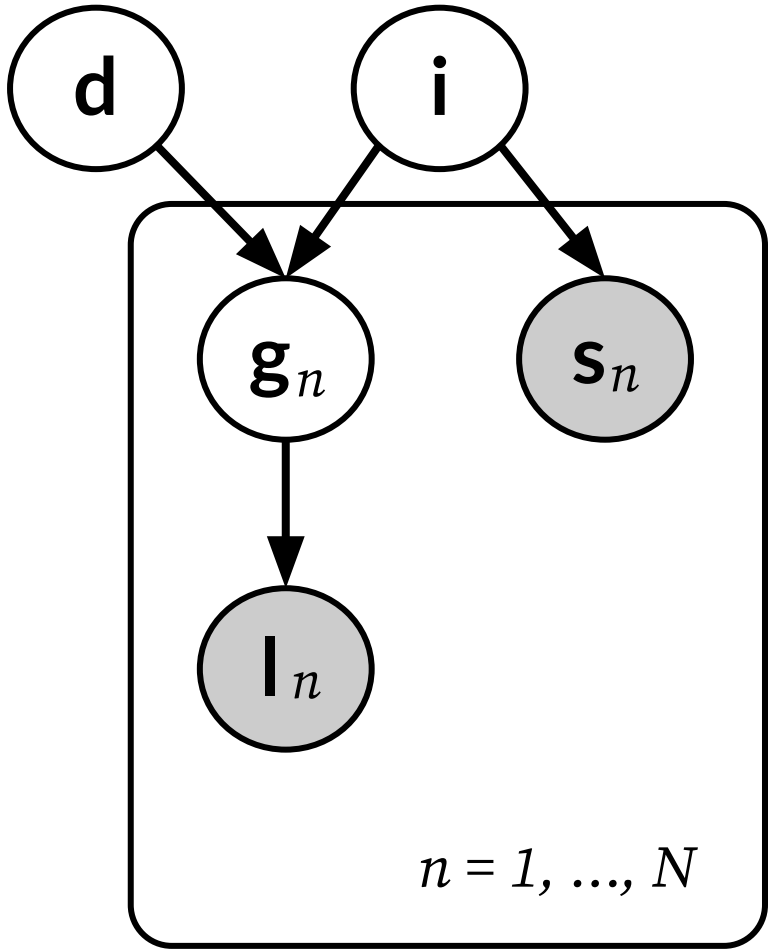
(Where there are a subset of variables within the plate 🤔 …)



# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.



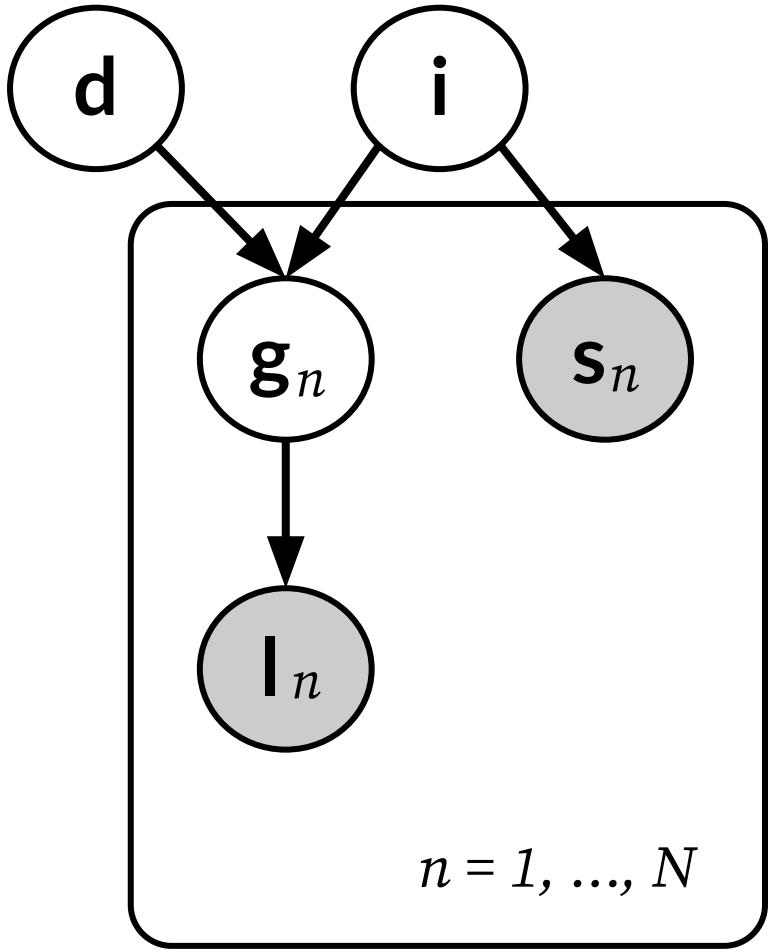



# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

This could be a model of (for example):

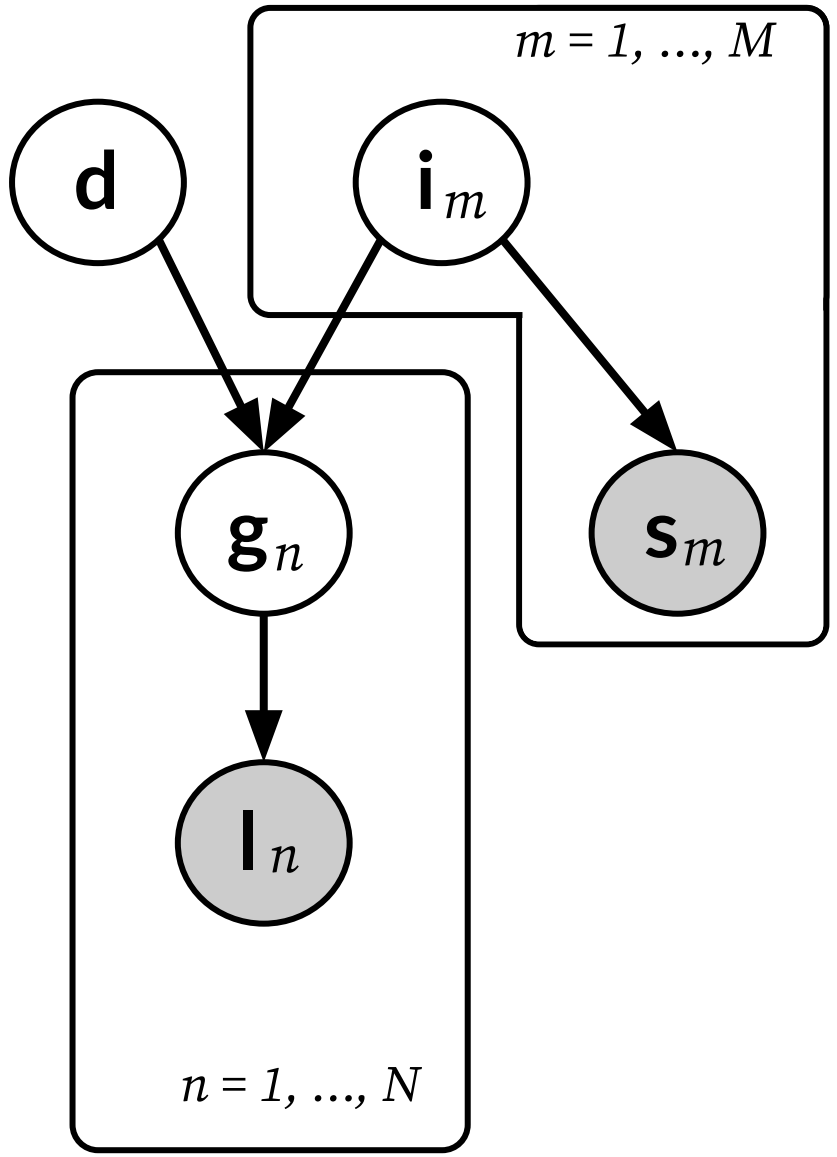
- A single student, with intelligence i, taking a single class with difficulty d.
- At $N$ times during the semester, they:
  - Receive a grade g.
  - Get a letter of rec based on that grade.
  - Separately, take an SAT test.



# **Bayesian Networks** — Plate Notation

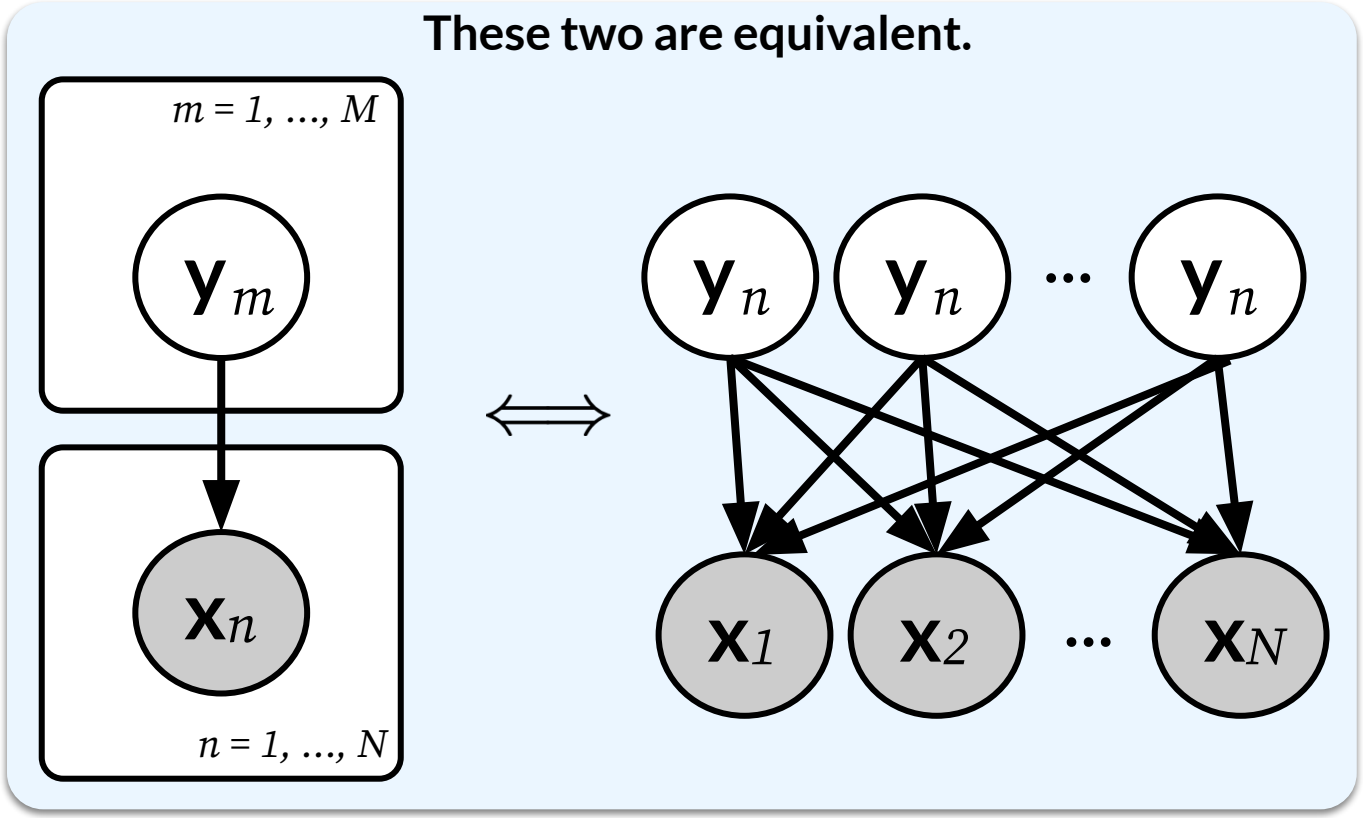
Plate notation is a “visual language” for describing more-complex Bayesian networks.

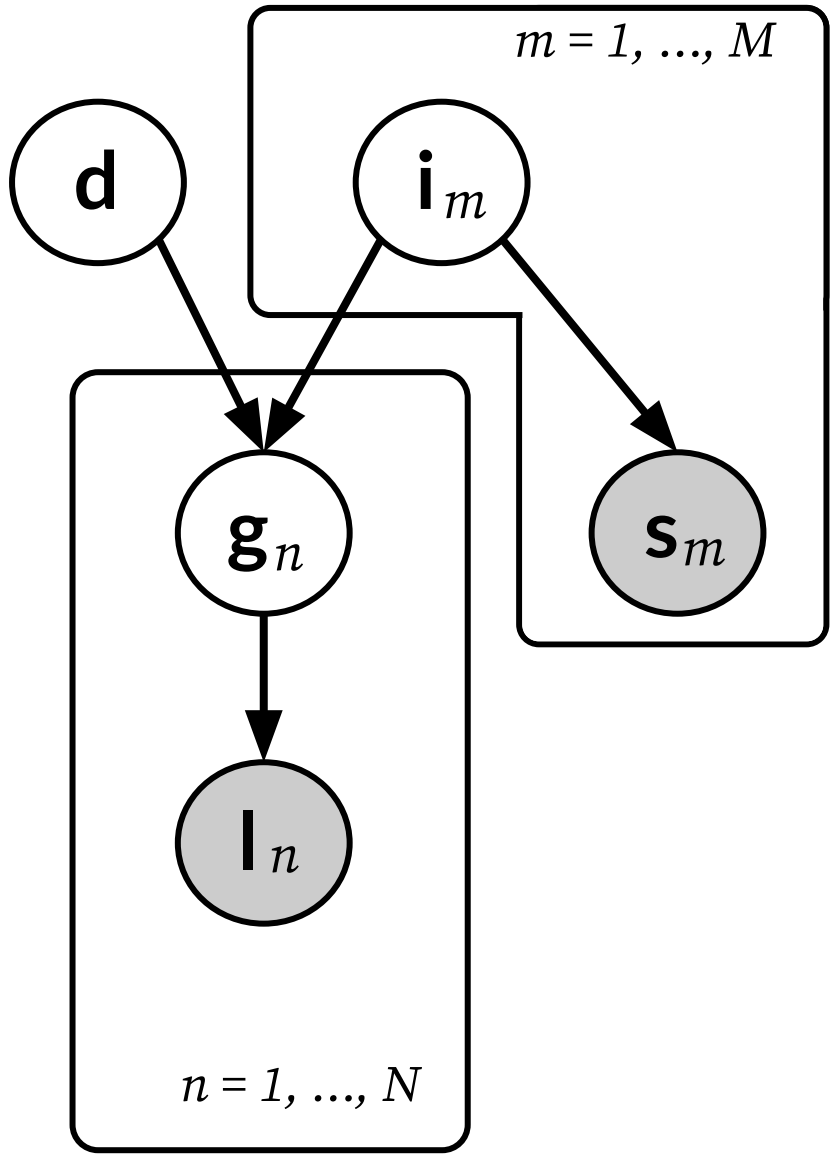
**What about this structure?**



# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.



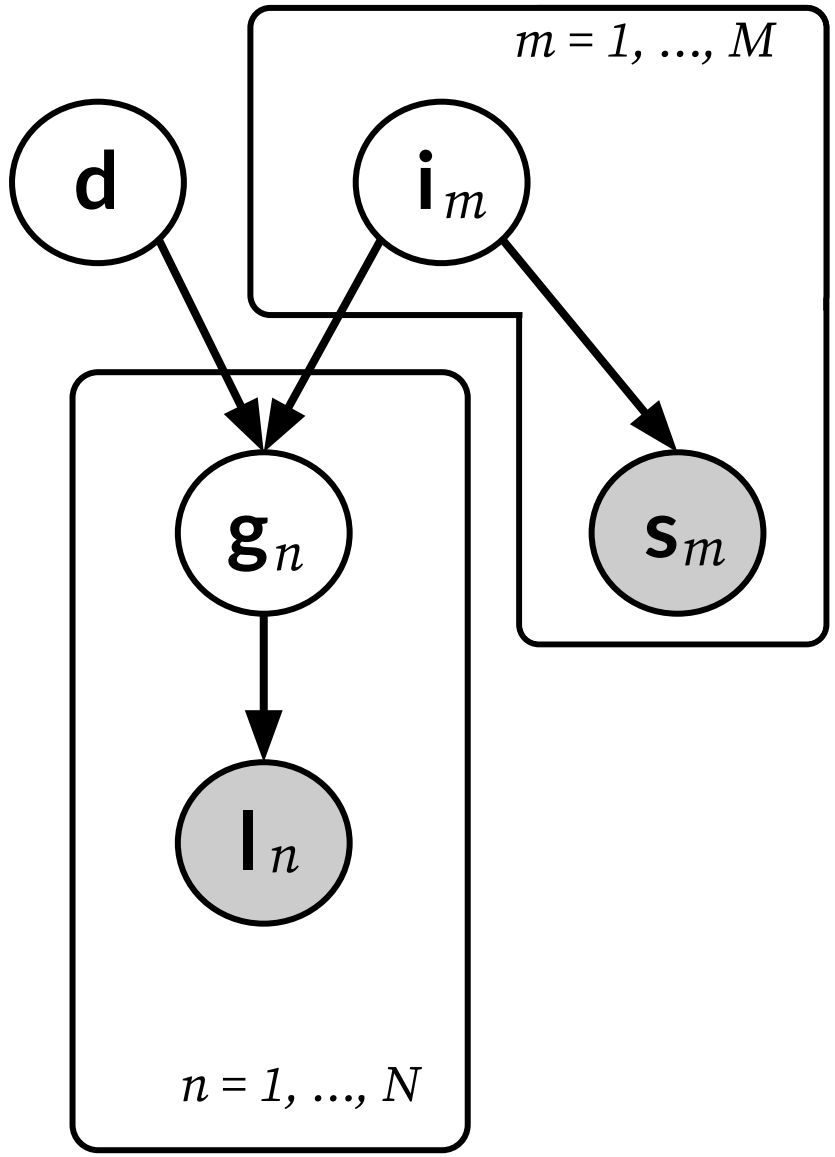



# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

This could be a model of (for example):

- $M$ students, each with an intelligence i, in a group for a course project.
  - Each student in the group takes SAT test.
- At $N$ times during the semester, the group:
  - Receives a grade g.
  - Gets a letter of rec based on that grade.

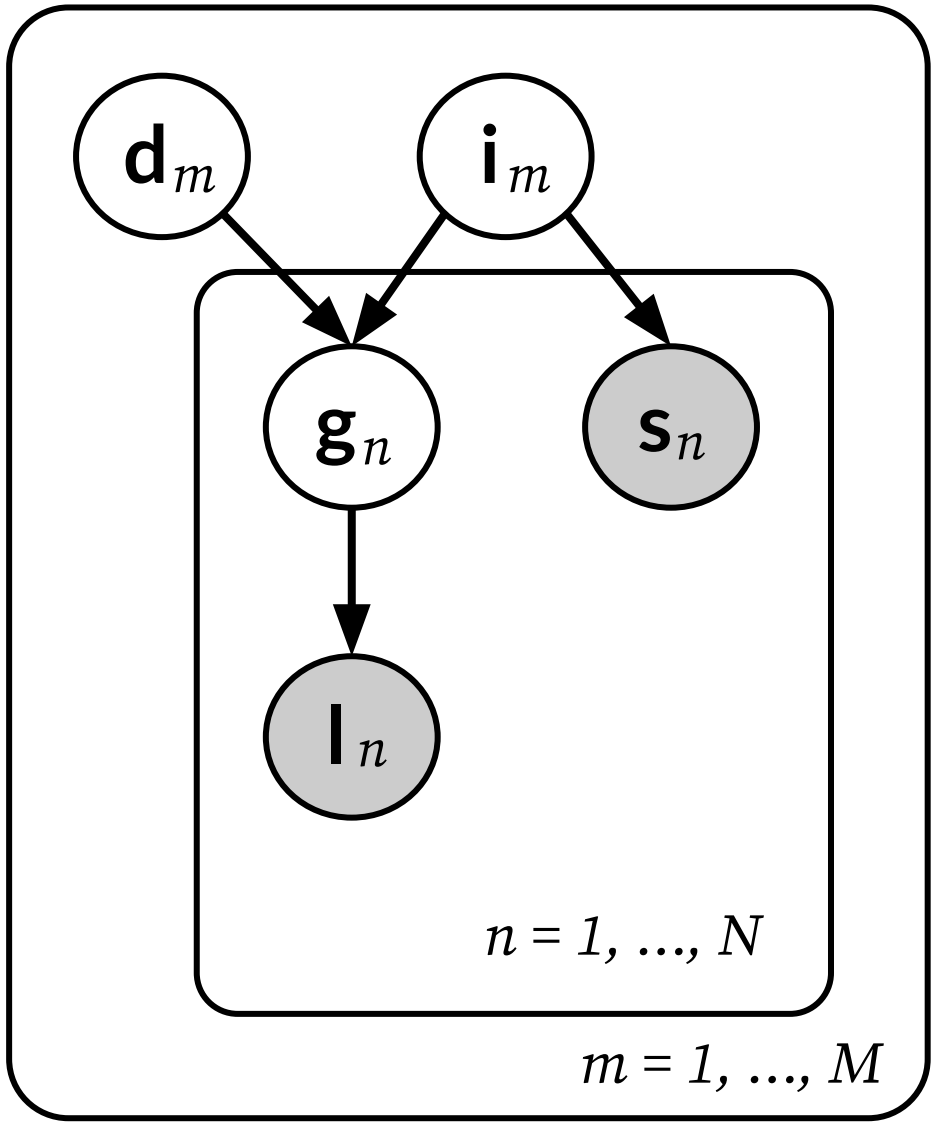


# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

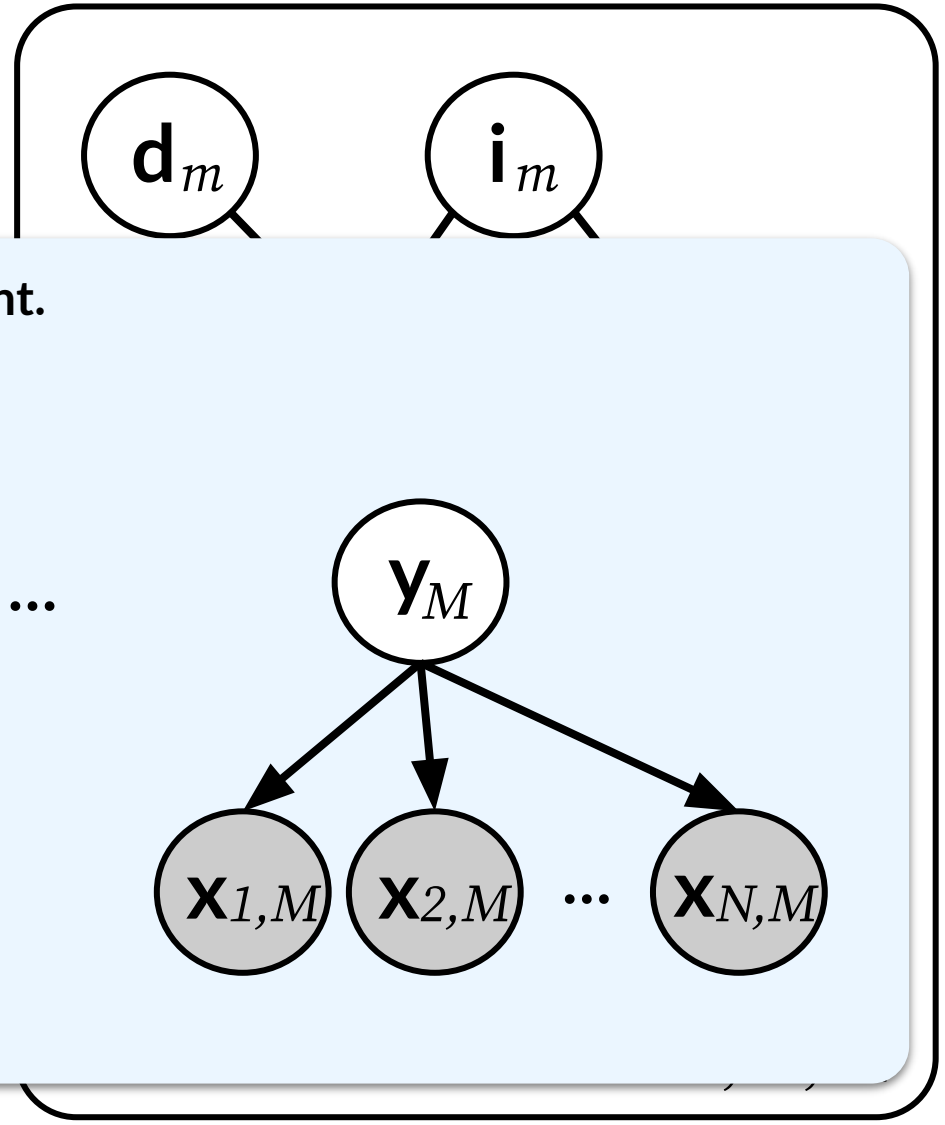
**How about multiple (concentric) plates?**

⇒ Yes!



# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian




**These two are equivalent.**

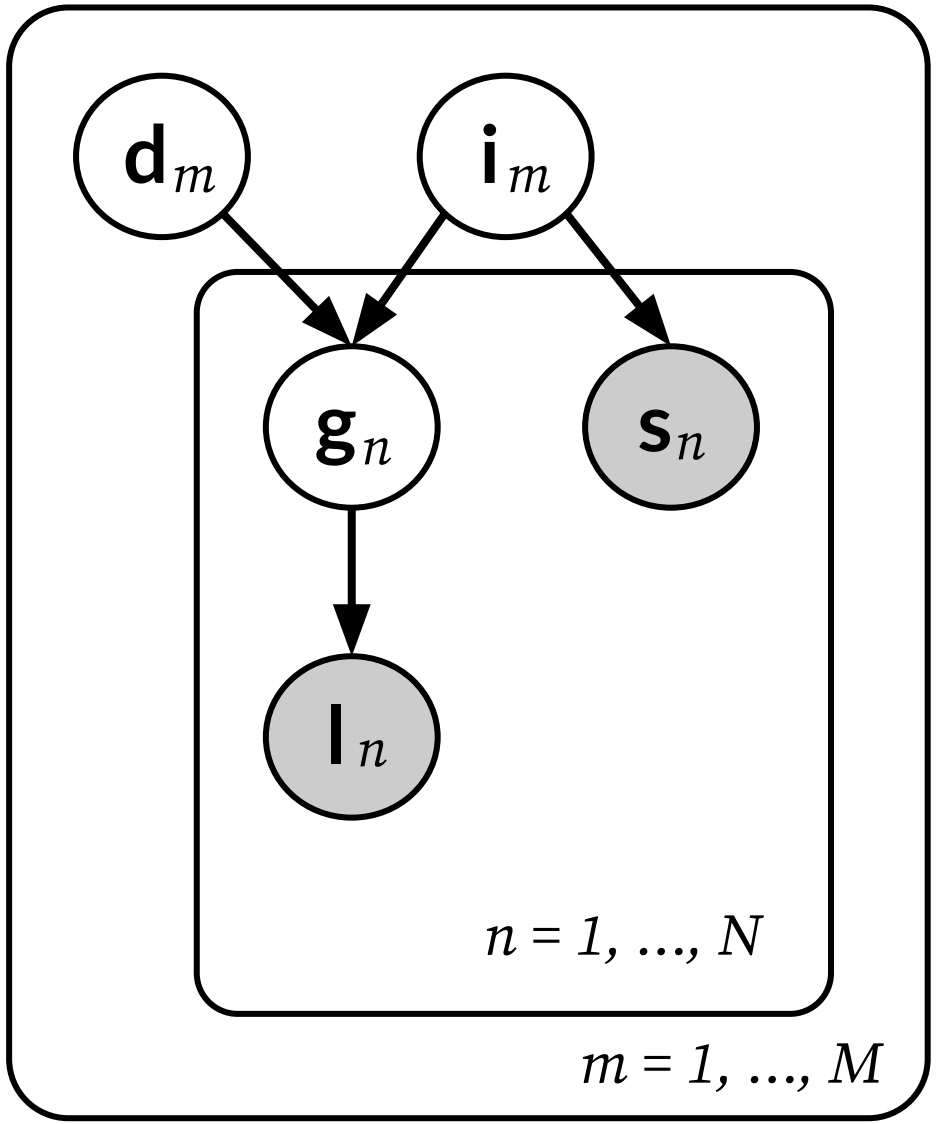
# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

Could be viewed as a model of…

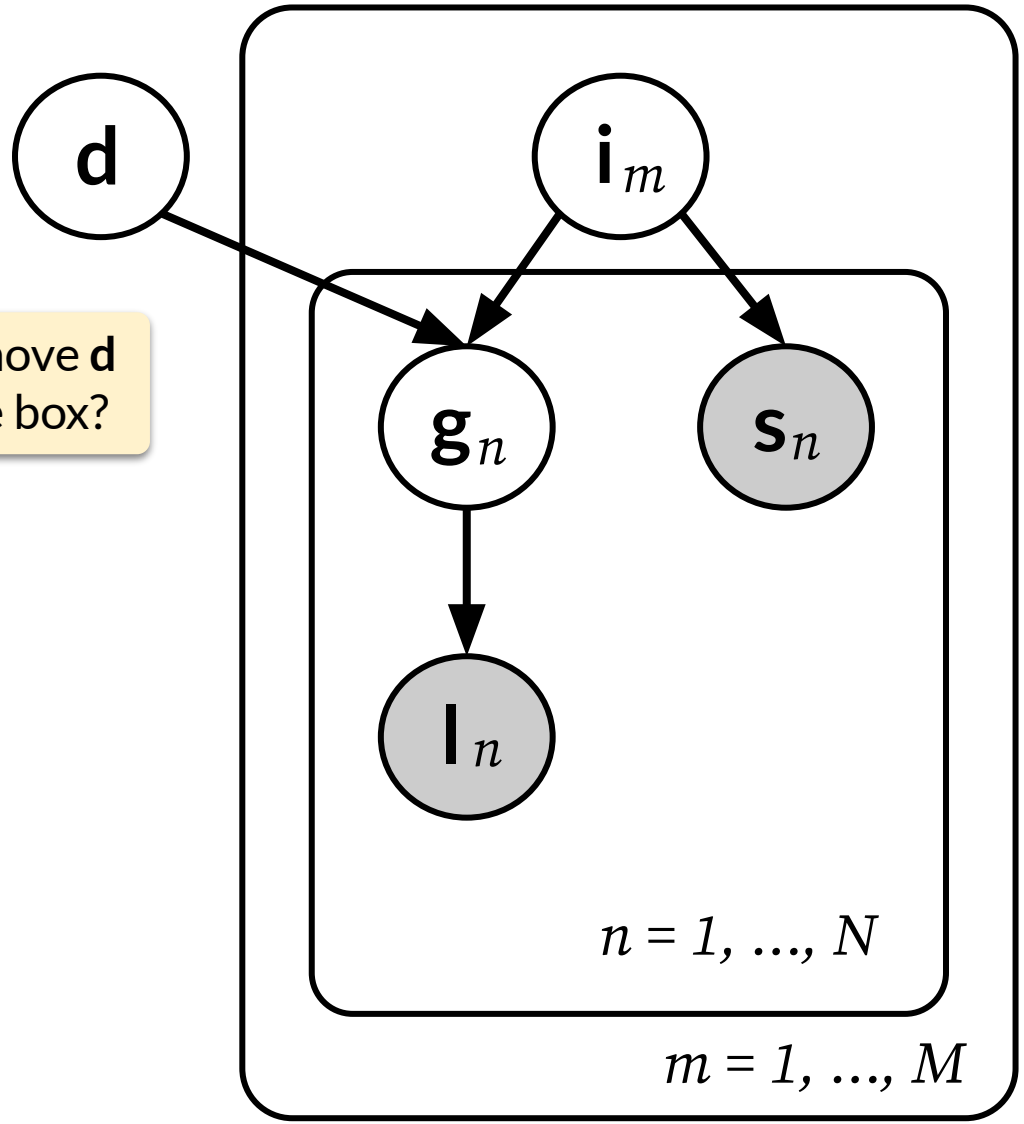
Repeat the following for $M$ students:

- Each student, with intelligence i, is taking a (separate) class with difficulty d.
- At $N$ times during the semester, they:
  - Receive a grade g.
  - Get a letter of rec based on that grade.
  - Separately, take an SAT test.



# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.




And if we move **d** outside the box?

# **Bayesian Networks** — Plate Notation

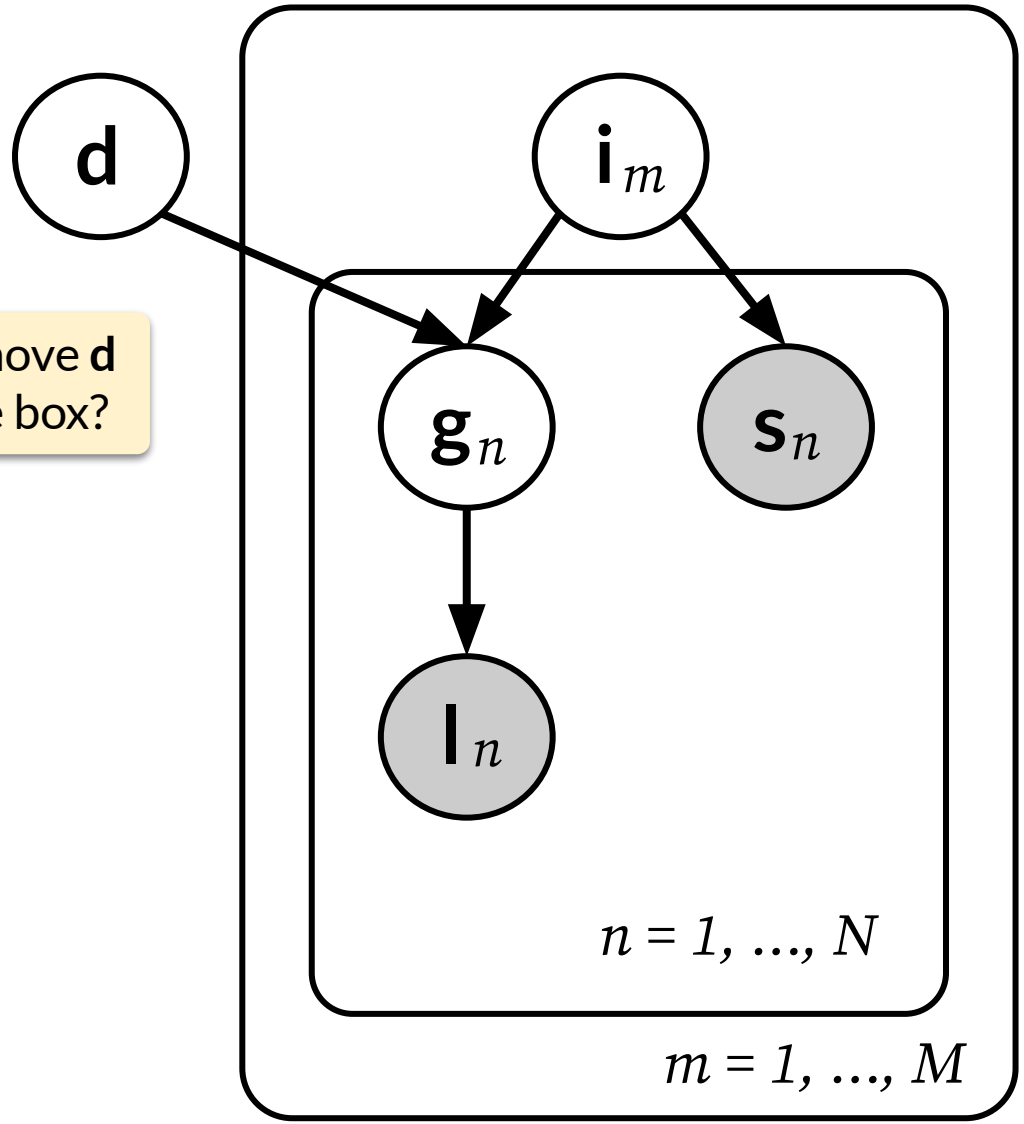
Plate notation is a “visual language” for describing more-complex Bayesian networks.

Could be viewed as a model of…

Repeat the following for $M$ students:

- A single student, with intelligence i, taking **the same** class with difficulty d.
- At $N$ times during the semester, they:
  - Receive a grade g.
  - Get a letter of rec based on that grade.
  - Separately, take an SAT test.

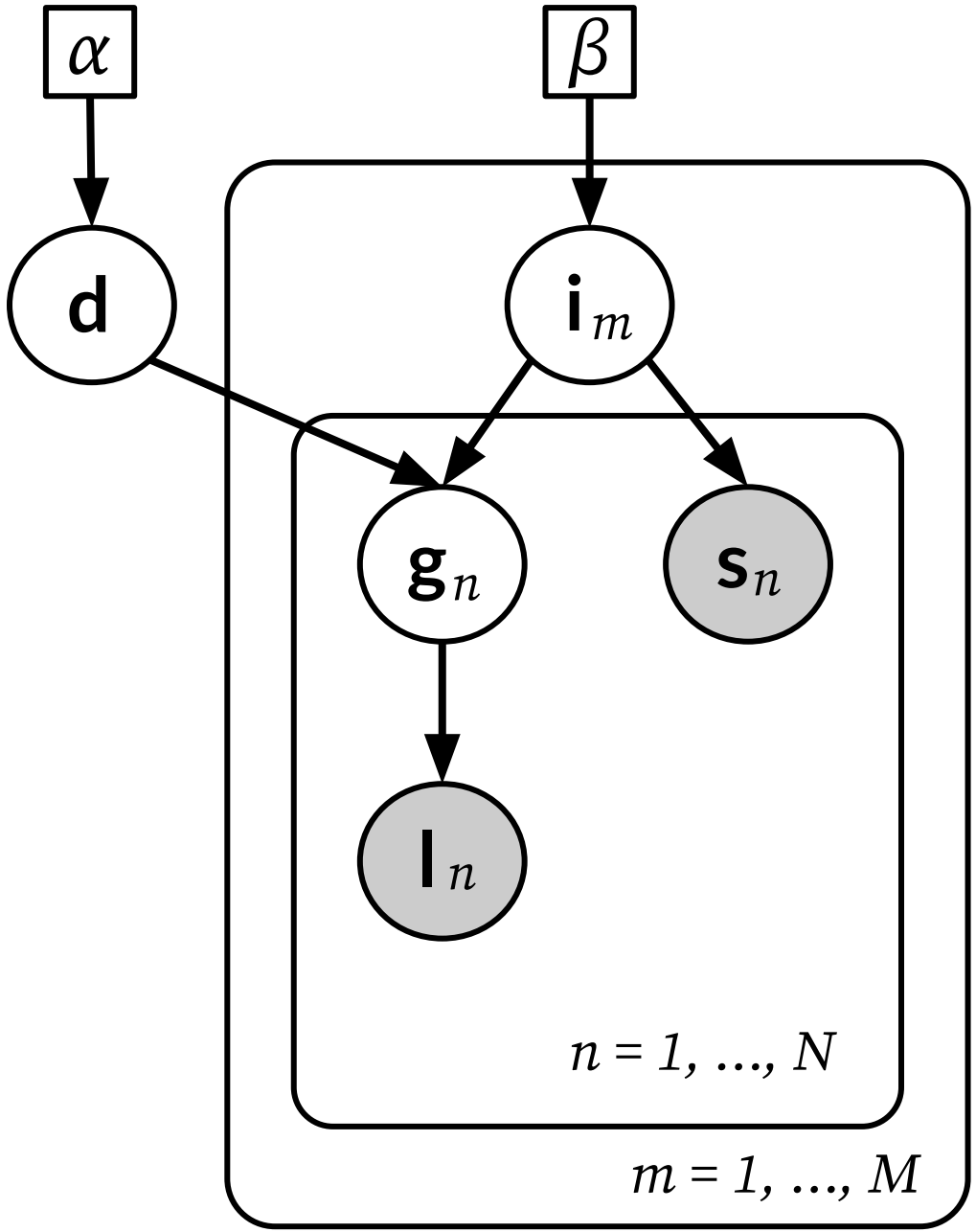
And if we move **d** outside the box?



# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

Sometimes, constant variables (*i.e.*, **not random** variables) are included.



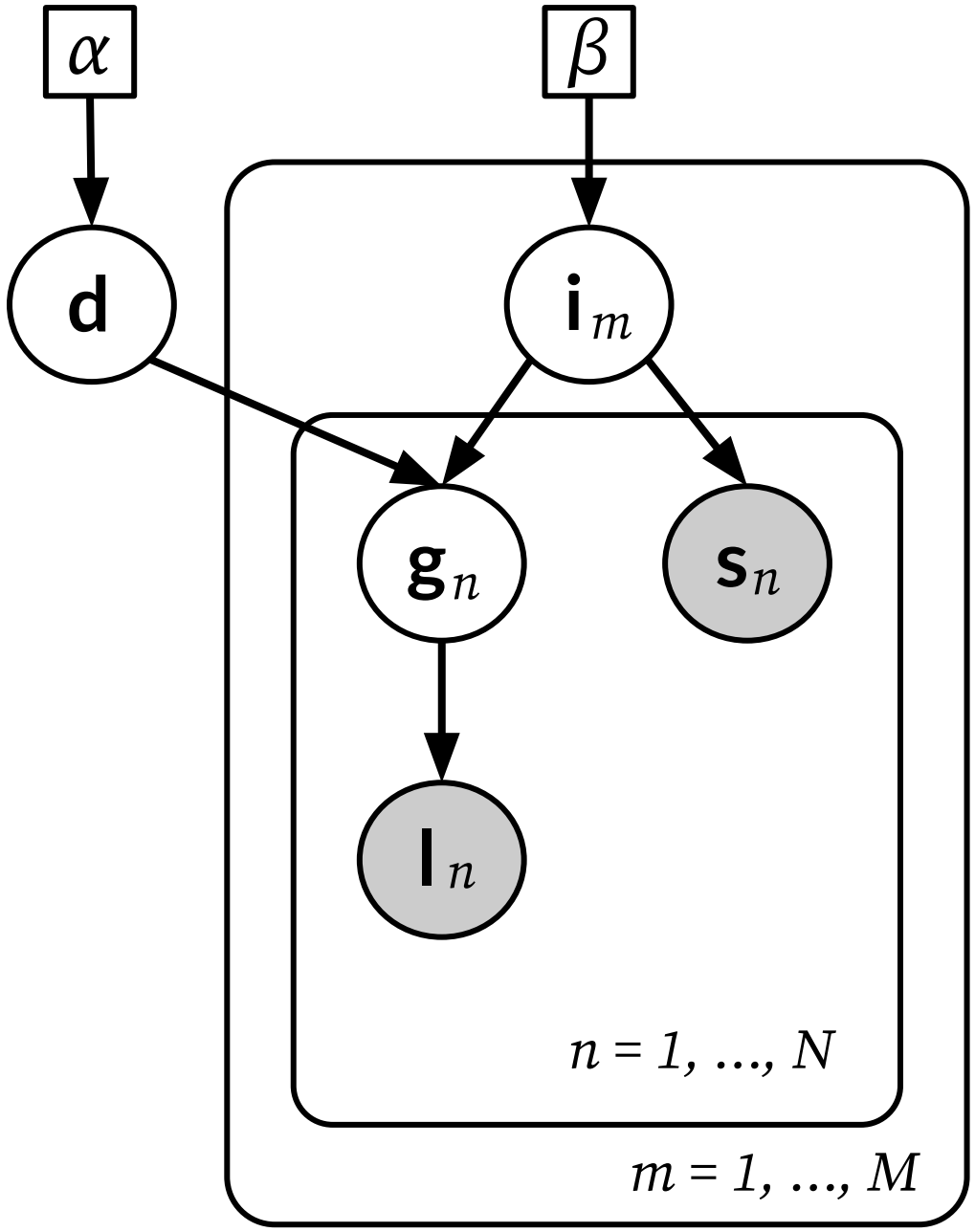
# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

Sometimes, constant variables (*i.e.*, **not random** variables) are included.

- For example, a parameter of a given distribution.
- (Often these are referred to as “hyperparameters”).
- Example:

$$d \sim \text{Gamma}(\alpha, 1)$$
$$i_m \sim \text{Bernoulli}(\beta), \text{ for } m = 1, \ldots, M$$



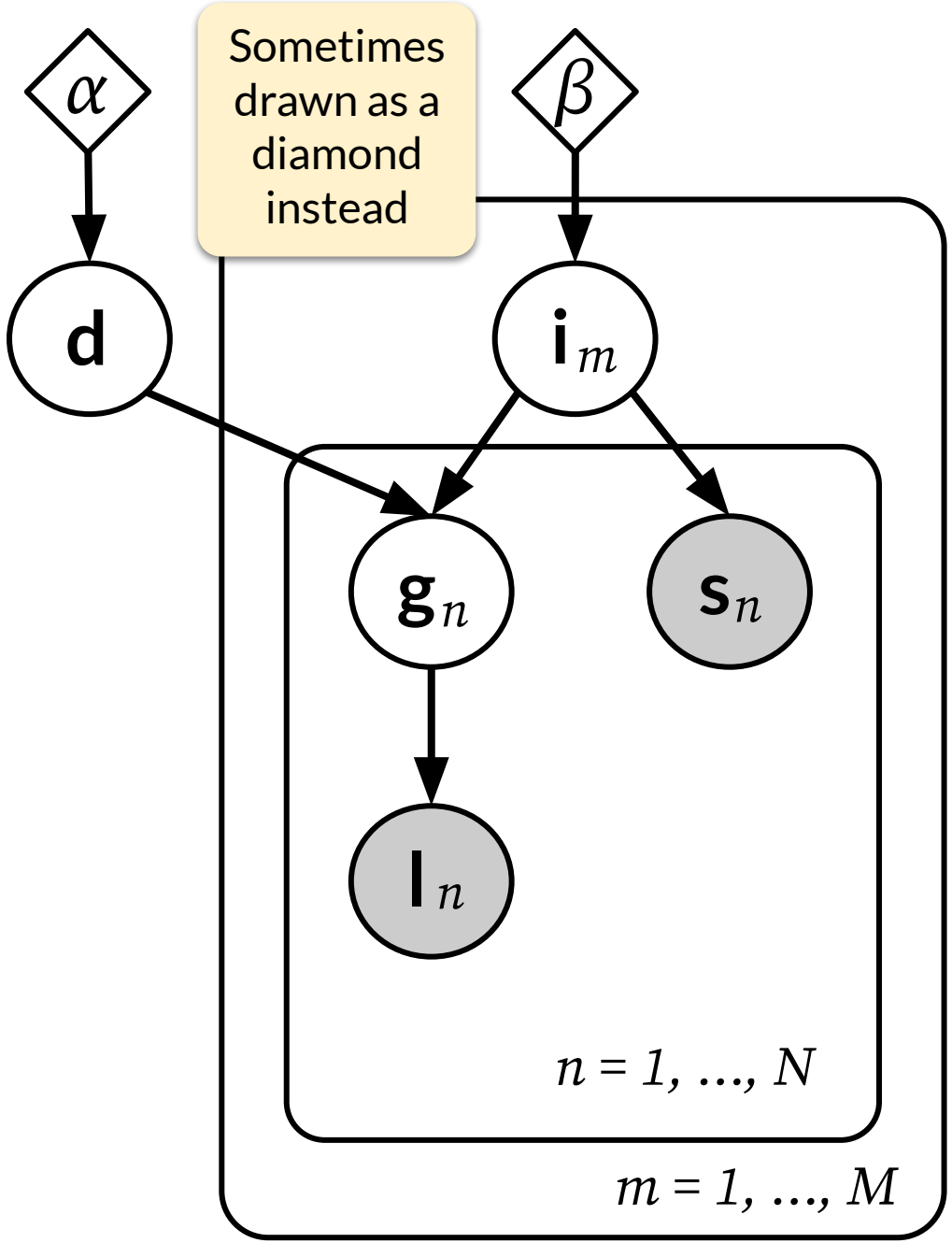
# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

Sometimes, constant variables (*i.e.*, **not random** variables) are included.

- For example, a parameter of a given distribution.
- (Often these are referred to as “hyperparameters”).
- Example:

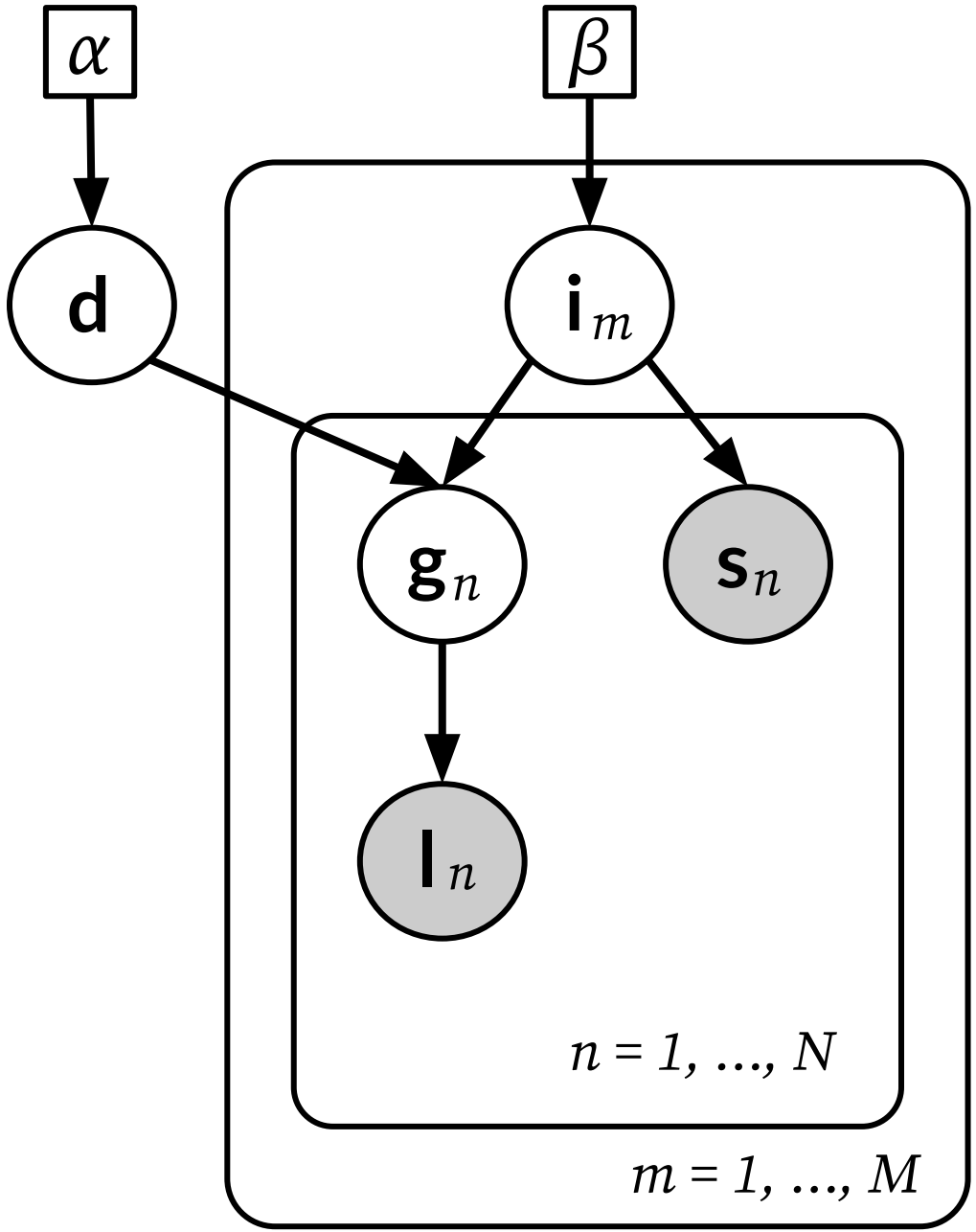
$$d \sim \text{Gamma}(\alpha, 1)$$
$$i_m \sim \text{Bernoulli}(\beta), \text{ for } m = 1, \ldots, M$$



# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

**To summarize, main rules of plate notation:**

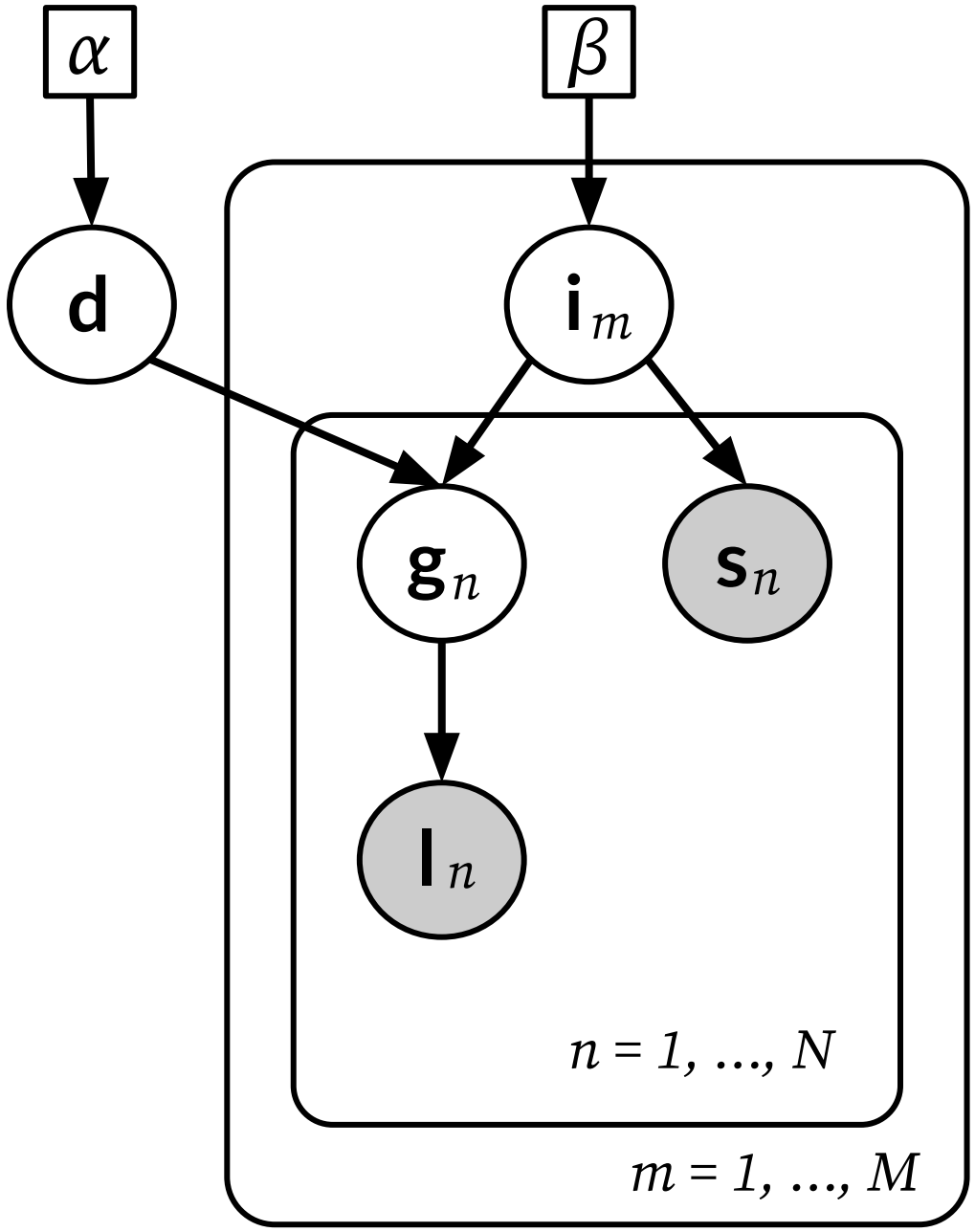


# **Bayesian Networks** — Plate Notation

Plate notation is a “visual language” for describing more-complex Bayesian networks.

**To summarize, main rules of plate notation:**

- Each random variable (or vector) has a node.
- Shaded=observed variable. Unshaded=latent variable.
- Constants (*eg*, hyperparameters) as squares/diamonds.
- Plates with indices denote repeated variables.
- Variables within a plate do not have directed edges between them.



# **Bayesian Networks** — Summary of Benefits

To summarize…. *What are the benefits of representing a joint PDF in this way (as a PGM)?*

# **Bayesian Networks** — Summary of Benefits

To summarize…. *What are the benefits of representing a joint PDF in this way (as a PGM)?*

Can define & identify conditional independencies (reductions in complexity) via the graph structure.

# **Bayesian Networks** — Summary of Benefits

To summarize…. *What are the benefits of representing a joint PDF in this way (as a PGM)?*

Can define & identify conditional independencies (reductions in complexity) via the graph structure.

Leads to algorithms for inference (computing conditionals/marginals) **based on the graph structure**.

# **Bayesian Networks** — Summary of Benefits

To summarize…. *What are the benefits of representing a joint PDF in this way (as a PGM)?*

Can define & identify conditional independencies (reductions in complexity) via the graph structure.

Leads to algorithms for inference (computing conditionals/marginals) **based on the graph structure**.

Sometimes this graph/plate data structure is used to specify a probabilistic model for deployment of automatic inference in probabilistic programming languages.

**NumPyro**

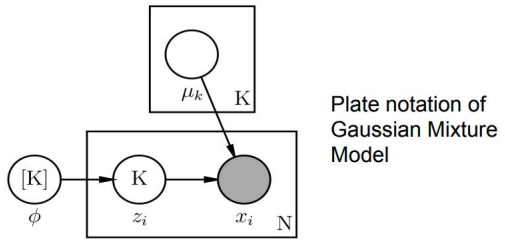



Plate notation of Gaussian Mixture Model

```
def gmm(data, K):
    phi = sample("phi", Dirichlet(np.ones(K)))
    with plate("K", K, dim=-1):
        mu = sample("mu", Normal(np.arange(K), 1))
    with plate("N", len(data), dim=-1):
        z = sample("z", Categorical(phi))
        sample("obs", Normal(mu[z], 1), obs=data)
```

NumPyro: Using Composable Effects for Flexible and Accelerated Probabilistic Programming, Phan et al.

# **Preview** — Next Class

In next class, we will go into the details of inference algorithms (for computing the conditionals/marginals) based on the graph structure!

But first in this class, want to show some example (famous) Bayesian networks, and give some examples of where they've been used…

# Famous Bayesian Networks!

Going through examples of famous Bayesian networks (and what they model).

- Markov models (Markov chains).
- Hidden Markov models (HMMs).
- Gaussian mixture models (GMMs).
- Latent Dirichlet allocation (LDA).
- Variational Autoencoder (VAE).

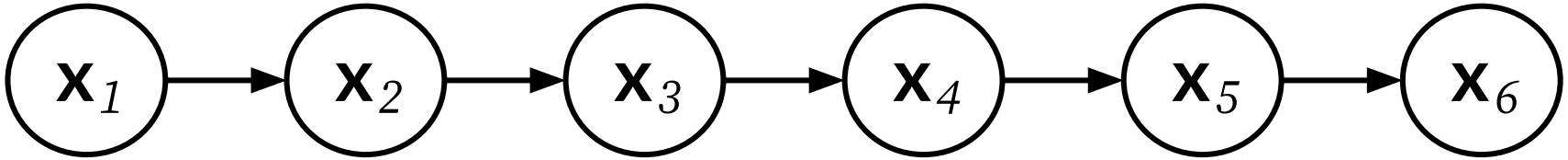
# **Famous Bayesian Networks** — Markov Models

# **Famous Bayesian Networks** — Markov Models

Markov Models … also known as **Markov chains**.

# **Famous Bayesian Networks** — Markov Models

Markov Models … also known as **Markov chains**.

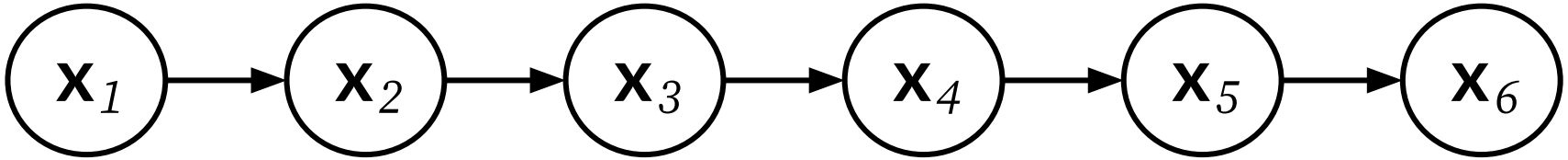



Generative process:

$$x_1 \sim p(\cdot)$$
$$x_2 \sim p(x_1)$$
$$\dots$$
$$x_n \sim p(x_{n-1})$$

# **Famous Bayesian Networks** — Markov Models

Markov Models … also known as **Markov chains**.



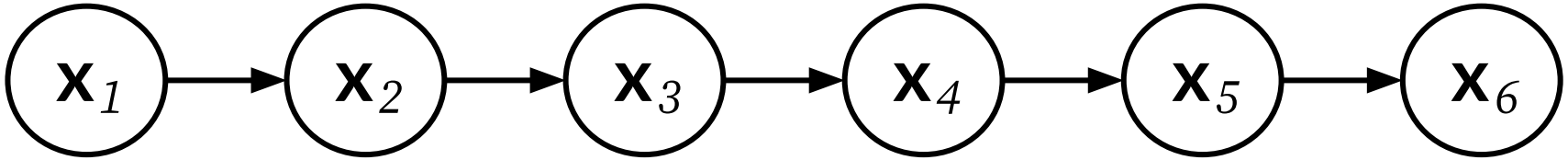

Generative process:

$$x_1 \sim \mathcal{N}(0, 1)$$
$$x_2 \sim \mathcal{N}(x_1, 1)$$
$$\dots$$
$$x_n \sim \mathcal{N}(x_{n-1}, 1)$$

An example (with normal distributions)

# **Famous Bayesian Networks** — Markov Models

Markov Models … also known as **Markov chains**.



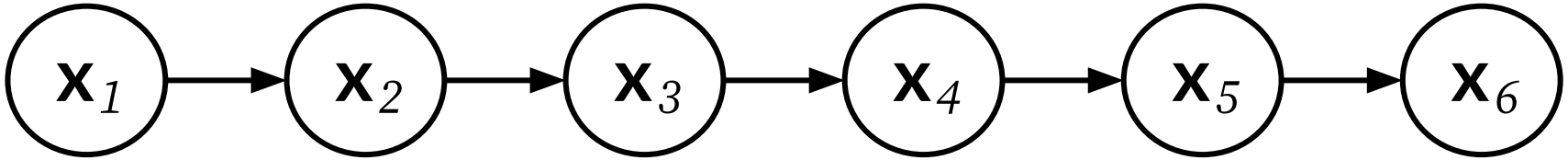

Generative process:

$$x_1 \sim p(x_1)$$
$$x_2 \sim p(x_2 \mid x_1)$$
$$\dots$$
$$x_n \sim p(x_n \mid x_{n-1})$$

An alternative way of writing the generative process

# **Famous Bayesian Networks** — Markov Models

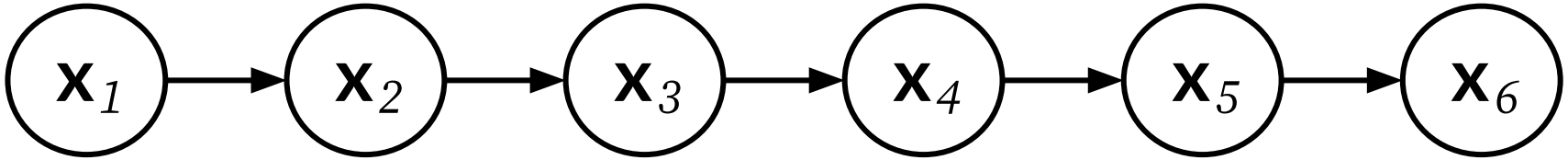
Markov Models … also known as **Markov chains**.




Applications:

# **Famous Bayesian Networks** — Markov Models

Markov Models … also known as **Markov chains**.




Applications:

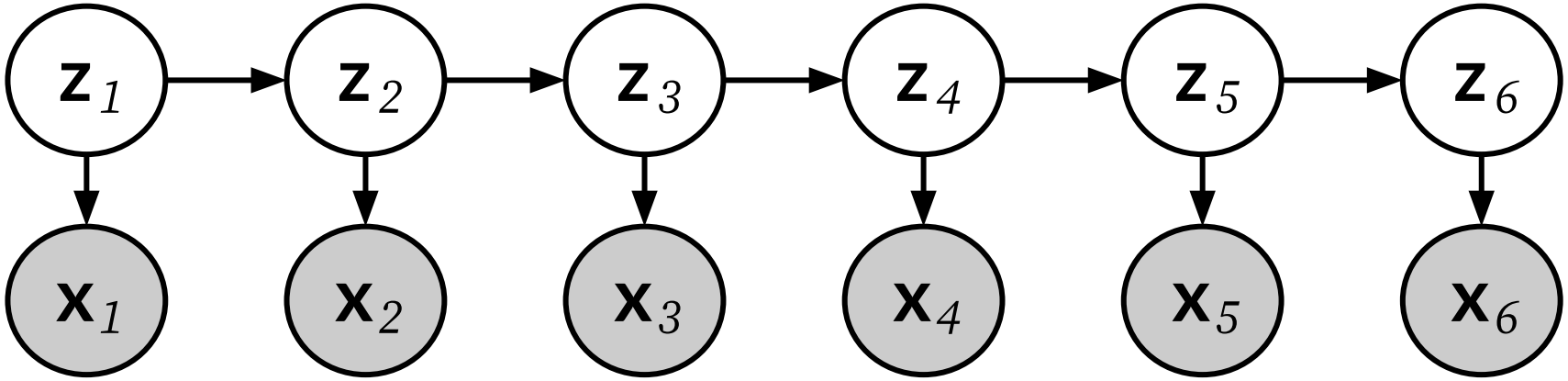
- Markov chain Monte Carlo — an approximate Bayesian inference method.
  - For sampling from a conditional (posterior) distribution of a joint model.
- Business: consumer behavior, brand loyalty, brand switching.
- Operations Research: *e.g.*, models in queuing theory.

**Often**: sampling, rather than inference of conditional distribution given observations.

# **Famous Bayesian Networks** — Hidden Markov Models (HMMs)

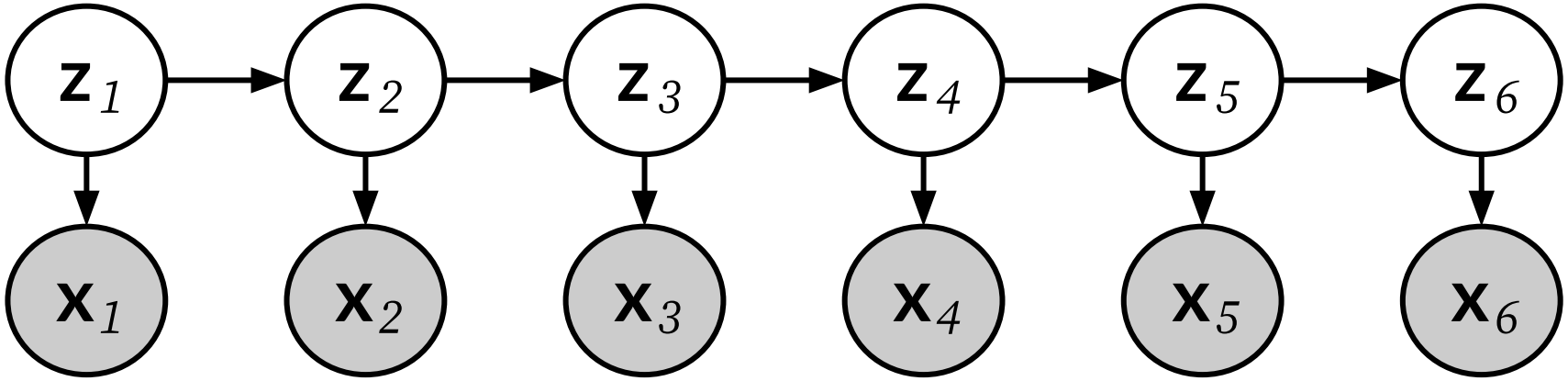
# **Famous Bayesian Networks** — Hidden Markov Models (HMMs)

Hidden Markov Models (HMMs) — an extension to partially observed Markov chains.



# **Famous Bayesian Networks** — Hidden Markov Models (HMMs)

Hidden Markov Models (HMMs) — an extension to partially observed Markov chains.




Backbone of latent variables (in a Markov chain).

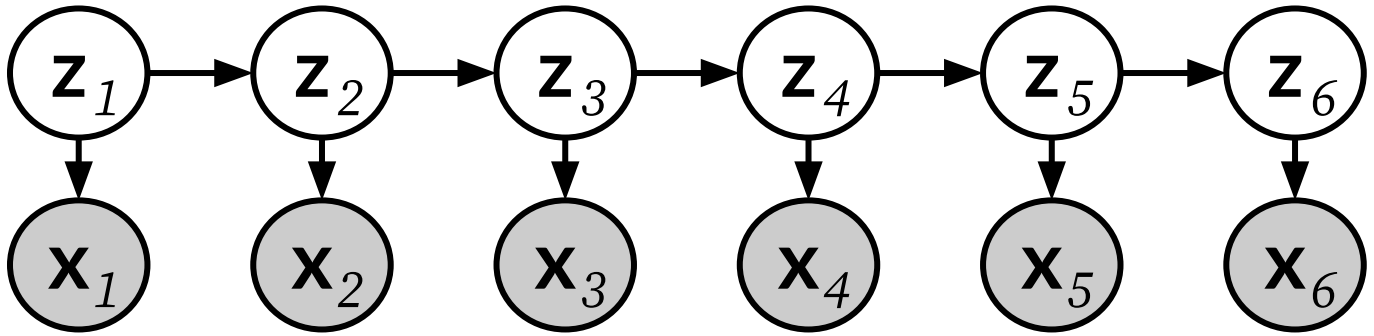
One observed variable (“emission variable”) for each latent variable.

# **Famous Bayesian Networks** — Hidden Markov Models (HMMs)

Hidden Markov Models (HMMs) — an extension to partially observed Markov chains.

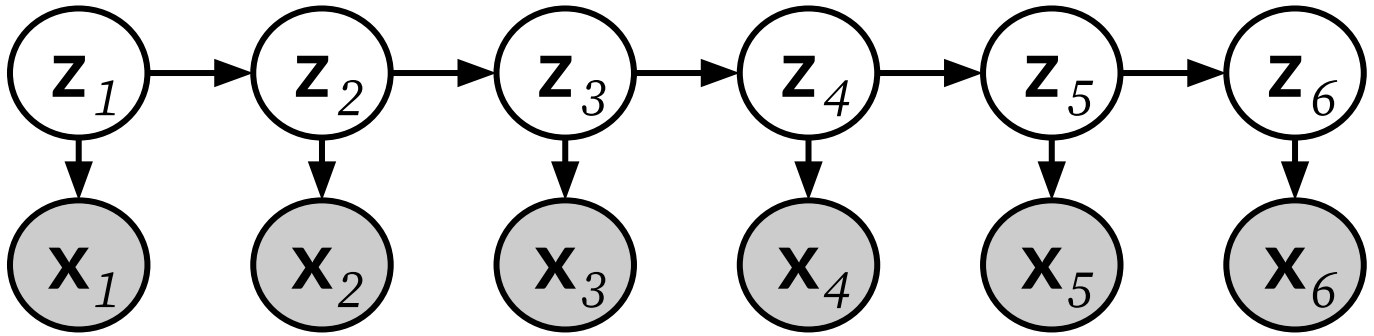
Generative process:

$$
\begin{aligned}
z_1 &\sim p(z_1) \\
x_1 &\sim p(x_1 \mid z_1) \\
z_2 &\sim p(z_2 \mid z_1) \\
x_2 &\sim p(x_2 \mid z_2) \\
&\dots \\
z_n &\sim p(z_n \mid z_{n-1}) \\
x_n &\sim p(x_n \mid z_n)
\end{aligned}
$$



# **Famous Bayesian Networks** — Hidden Markov Models (HMMs)

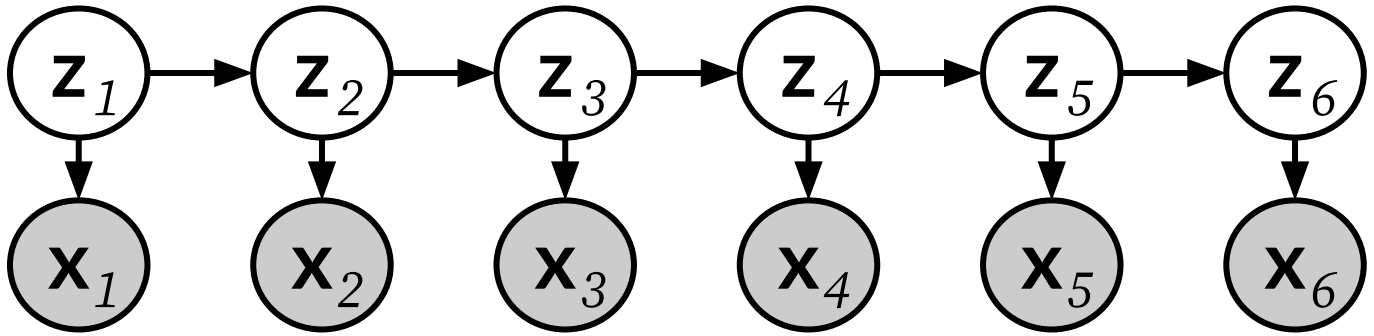
Hidden Markov Models (HMMs) — an extension to partially observed Markov chains.




Applications

# **Famous Bayesian Networks** — Hidden Markov Models (HMMs)

Hidden Markov Models (HMMs) — an extension to partially observed Markov chains.




Applications

- Time series modeling in speech, CV, computational biology (whenever there is a noisy signal).
- Viterbi algorithm (applies to this model): speech recognition, computational linguistics, bioinformatics, digital cellular/satellite/deep-space communications.

# **Famous Bayesian Networks** — Mixture Models (*e.g.*, GMMs)

Mixture Models, *e.g.*, Gaussian mixture models (GMMs)

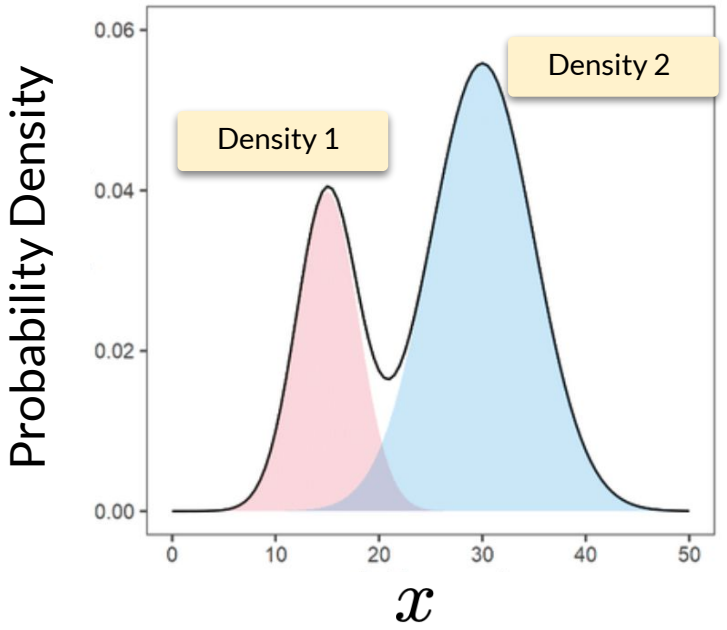
Source: “A primer to latent profile and latent class analysis”, Bauer et al.,

# **Famous Bayesian Networks** — Mixture Models (*e.g.*, GMMs)

Mixture Models, *e.g.*, Gaussian mixture models (GMMs)

Suppose we have samples from a *mixture of distributions* — i.e.

One dimensional data




Sometimes called:
"*Mixture densities*"
or
"*Component densities*"

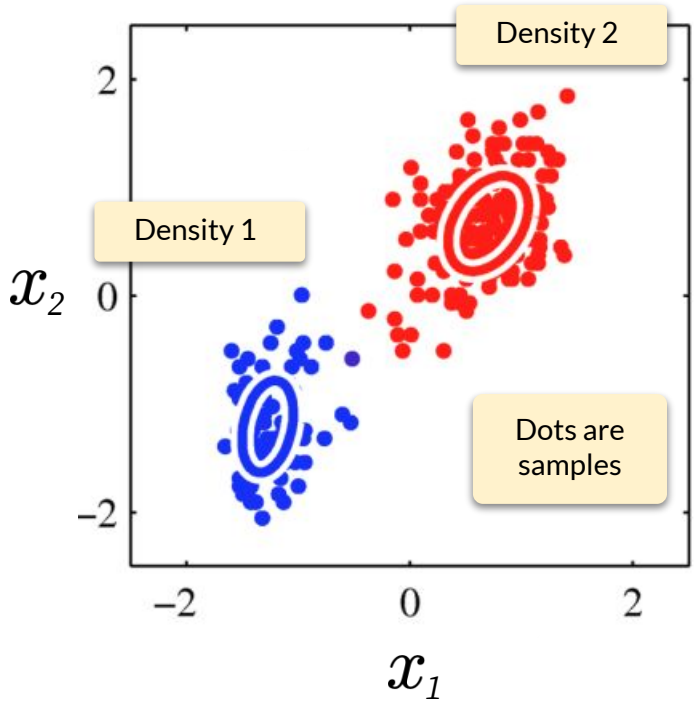
Source: "A primer to latent profile and latent class analysis", Bauer et al.,

# **Famous Bayesian Networks** — Mixture Models (*e.g.*, GMMs)

Mixture Models, *e.g.*, Gaussian mixture models (GMMs)

Suppose we have samples from a *mixture of distributions* — i.e.

Two dimensional data



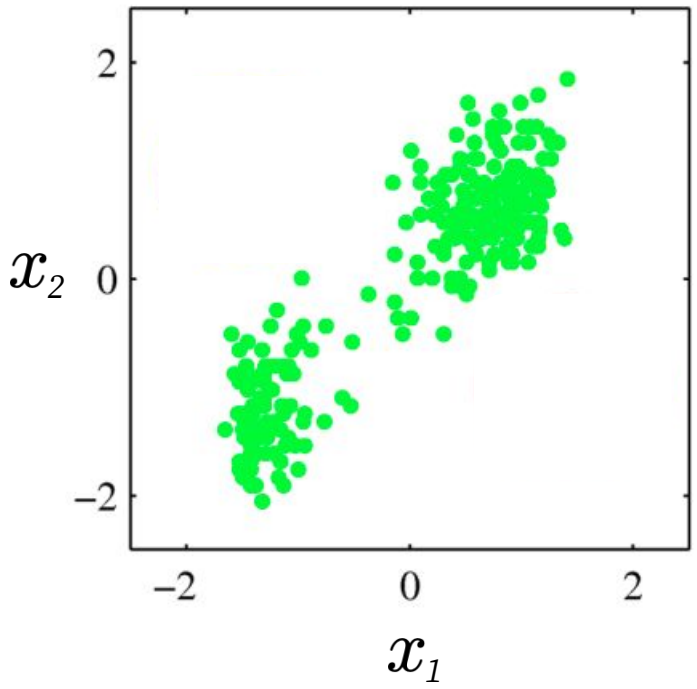

Source: Machine Learning for Computer Vision, Rudolph Triebel

# **Famous Bayesian Networks** — Mixture Models (*e.g.*, GMMs)

Mixture Models, *e.g.*, Gaussian mixture models (GMMs)

Suppose we have samples from a *mixture of distributions* — i.e.

Two dimensional data




What if we only observe the samples?

**Q:** How can we infer the parameters of the distributions that produced them?

Source: Machine Learning for Computer Vision, Rudolph Triebel

# **Famous Bayesian Networks** — Mixture Models (*e.g.*, GMMs)

Mixture Models, *e.g.*, Gaussian mixture models (GMMs)

**A:** We can use a PGM!

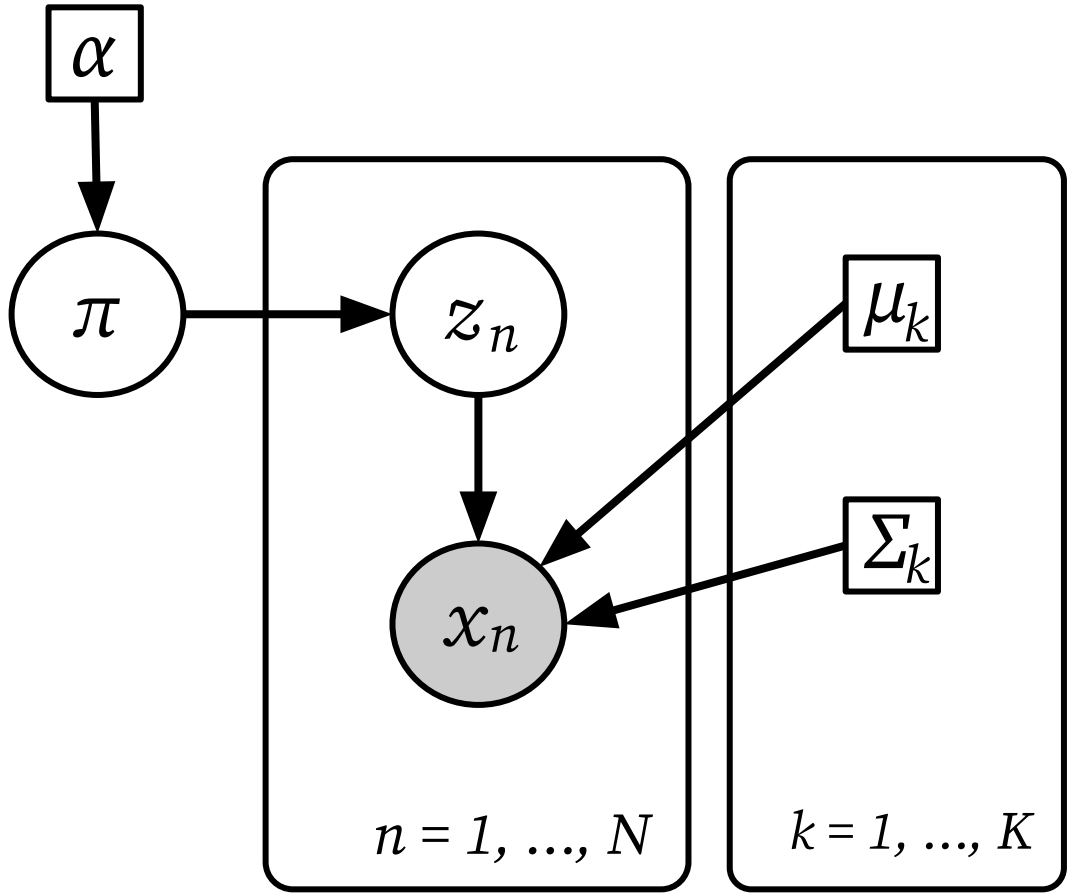
# **Famous Bayesian Networks** — Mixture Models (*e.g.*, GMMs)

Mixture Models, *e.g.*, Gaussian mixture models (GMMs)

**A:** We can use a PGM!

Called a **mixture model**.
(Or a Gaussian mixture model, GMM,
if component densities are normal)

Inference/estimation in this
model (given observations)
yields estimate of densities.

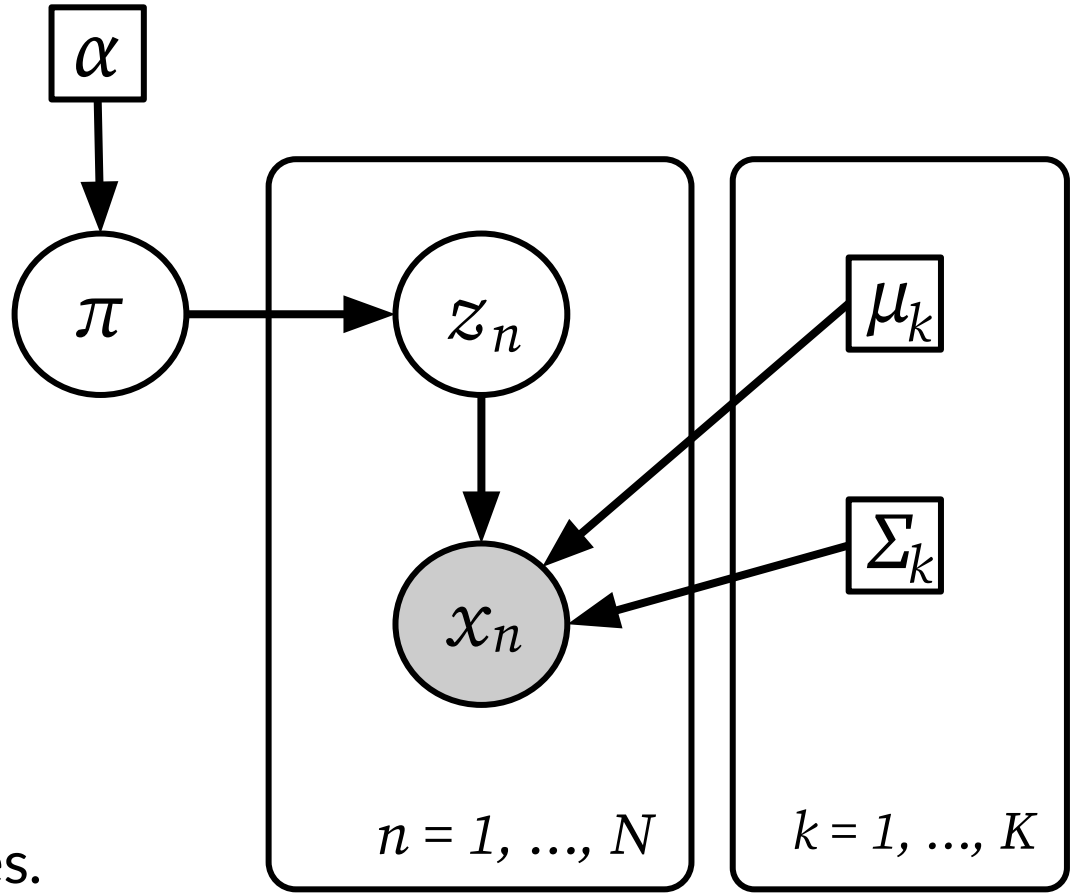


# **Famous Bayesian Networks** — Mixture Models (*e.g.*, GMMs)

Mixture Models, *e.g.*, Gaussian mixture models (GMMs)

Variables in PGM:

- $\pi$ - Mixture weights
- $z_n$ - Assignment variables (one per observation)
- $x_n$ - Observation variables (corresponds to dots in prev. image)
- $\mu_k, \Sigma_k$ - Parameters of mixture densities.

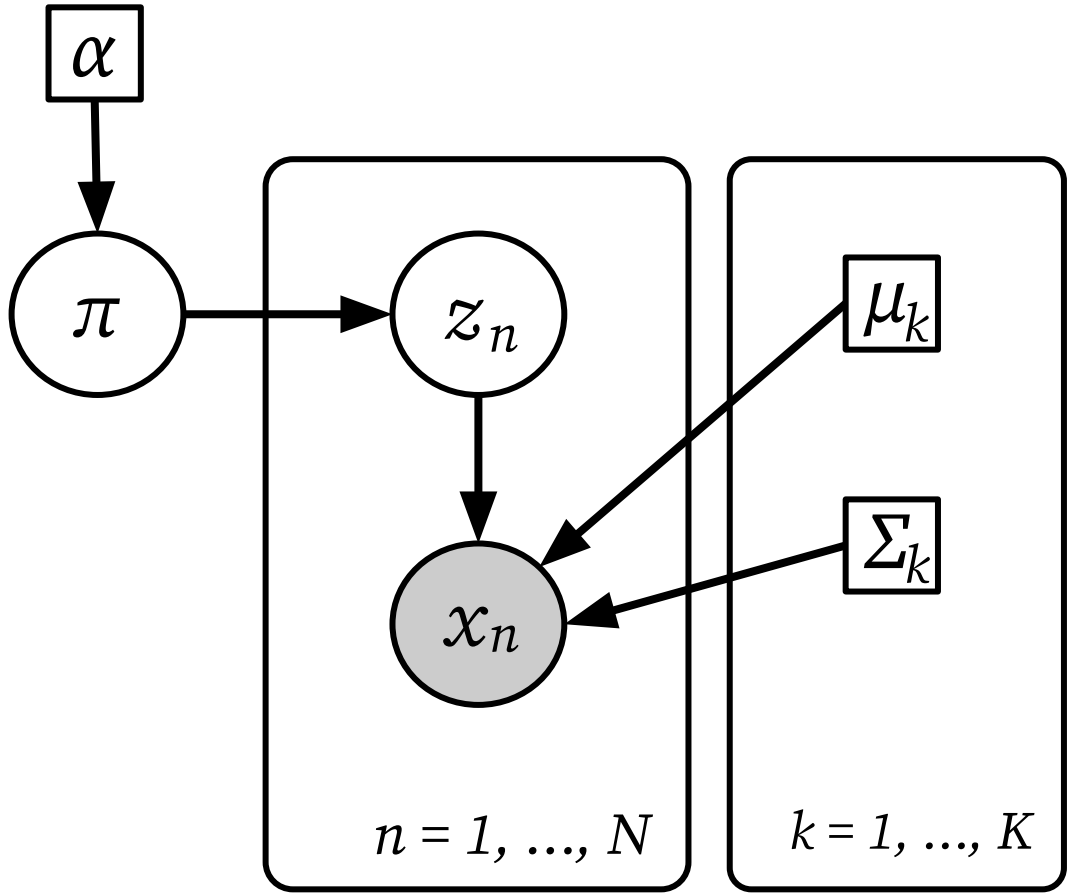


# **Famous Bayesian Networks** — Mixture Models (*e.g.*, GMMs)

Mixture Models, *e.g.*, Gaussian mixture models (GMMs)

Generative process:

$$\pi \sim \text{Dirichlet}(\alpha)$$
$$\text{for } n = 1, \ldots, N:$$
$$z_n \sim \text{Categorical}(\pi)$$
$$x_n \sim \mathcal{N}(\mu_{z_n}, \Sigma_{z_n})$$

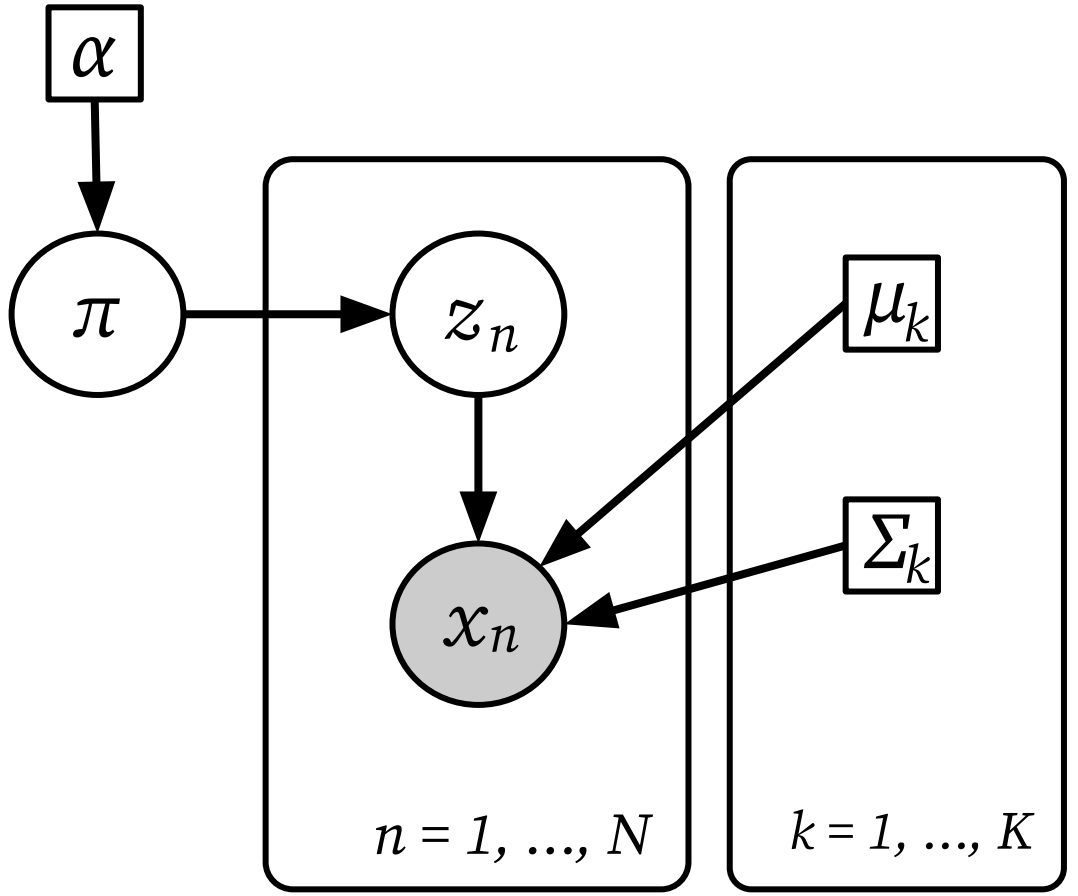


# **Famous Bayesian Networks** — Mixture Models (*e.g.*, GMMs)

Mixture Models, *e.g.*, Gaussian mixture models (GMMs)

Applications:

- Clustering of data.
- Density estimation.
- Anomaly detection.
- Image segmentation.
- Sometimes with far richer mixture densities (using neural networks).



# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Original Paper (2003)

→ 55,776 citations

Journal of Machine Learning Research 3 (2003) 993-1022 Submitted 2/02; Published 1/03

**Latent Dirichlet Allocation**

**David M. Blei** BLEI@CS.BERKELEY.EDU
*Computer Science Division*
*University of California*
*Berkeley, CA 94720, USA*

**Andrew Y. Ng** ANG@CS.STANFORD.EDU
*Computer Science Department*
*Stanford University*
*Stanford, CA 94305, USA*

**Michael I. Jordan** JORDAN@CS.BERKELEY.EDU
*Computer Science Division and Department of Statistics*
*University of California*
*Berkeley, CA 94720, USA*

**Editor:** John Lafferty

**Abstract**

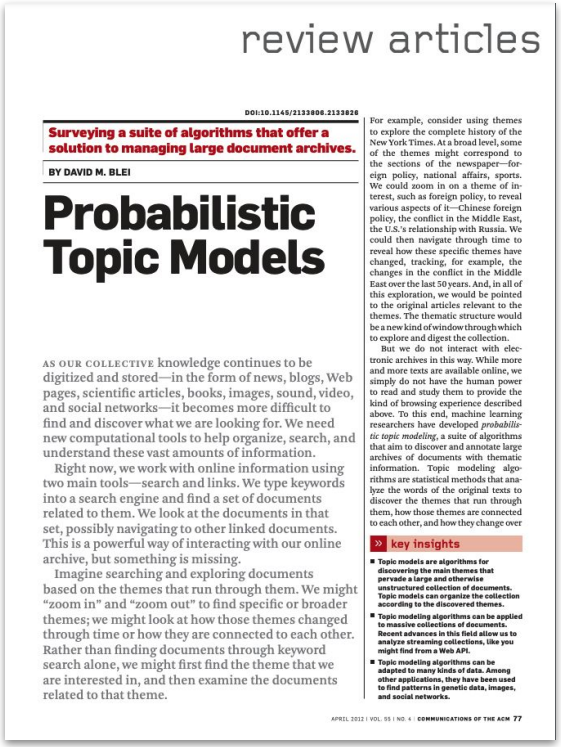
We describe *latent Dirichlet allocation* (LDA), a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document. We present efficient approximate inference techniques based on variational methods and an EM algorithm for empirical Bayes parameter estimation. We report results in document modeling, text classification, and collaborative filtering, comparing to a mixture of unigrams model and the probabilistic LSI model.

**1. Introduction**

In this paper we consider the problem of modeling text corpora and other collections of discrete data. The goal is to find short descriptions of the members of a collection that enable efficient processing of large collections while preserving the essential statistical relationships that are useful for basic tasks such as classification, novelty detection, summarization, and similarity and relevance judgments.

Significant progress has been made on this problem by researchers in the field of information retrieval (IR) (Baeza-Yates and Ribeiro-Neto, 1999). The basic methodology proposed by IR researchers for text corpora—a methodology successfully deployed in modern Internet search engines—reduces each document in the corpus to a vector of real numbers, each of which represents ratios of counts. In the popular *tf-idf* scheme (Salton and McGill, 1983), a basic vocabulary of "words" or "terms" is chosen, and, for each document in the corpus, a count is formed of the number of occurrences of each word. After suitable normalization, this term frequency count is compared to an inverse document frequency count, which measures the number of occurrences of a







DOI:10.1145/2133806.2133826

**Surveying a suite of algorithms that offer a solution to managing large document archives.**

BY DAVID M. BLEI

# Probabilistic Topic Models

AS OUR COLLECTIVE knowledge continues to be digitized and stored—in the form of news, blogs, Web pages, scientific articles, books, images, sound, video, and social networks—it becomes more difficult to find and discover what we are looking for. We need new computational tools to help organize, search, and understand these vast amounts of information.

Right now, we work with online information using two main tools—search and links. We type keywords into a search engine and find a set of documents related to them. We look at the documents in that set, possibly navigating to other linked documents. This is a powerful way of interacting with our online archive, but something is missing.

Imagine searching and exploring documents based on the themes that run through them. We might "zoom in" and "zoom out" to find specific or broader themes; we might look at how those themes changed through time or how they are connected to each other. Rather than finding documents through keyword search alone, we might first find the theme that we are interested in, and then examine the documents related to that theme.

For example, consider using themes to explore the complete history of the New York Times. At a broad level, some of the themes might correspond to the sections of the newspaper—foreign policy, national affairs, sports. We could zoom in on a theme of interest, such as foreign policy, to reveal various aspects of it—Chinese foreign policy, the conflict in the Middle East, the U.S.'s relationship with Russia. We could then navigate through time to reveal how these specific themes have changed, tracking, for example, the changes in the conflict in the Middle East over the last 50 years. And, in all of this exploration, we would be pointed to the original articles relevant to the themes. The thematic structure would be a new kind of window through which to explore and digest the collection.

But we do not interact with electronic archives in this way. While more and more texts are available online, we simply do not have the human power to read and study them to provide the kind of browsing experience described above. To this end, machine learning researchers have developed *probabilistic topic modeling*, a suite of algorithms that aim to discover and annotate large archives of documents with thematic information. Topic modeling algorithms are statistical methods that analyze the words of the original texts to discover the themes that run through them, how those themes are connected to each other, and how they change over

**» key insights**

- **Topic models are algorithms for discovering the main themes that pervade a large and otherwise unstructured collection of documents. Topic models can organize the collection according to the discovered themes.**
- **Topic modeling algorithms can be applied to massive collections of documents. Recent advances in this field allow us to analyze streaming collections, like you might find from a Web API.**
- **Topic modeling algorithms can be adapted to many kinds of data. Among other applications, they have been used to find patterns in genetic data, images, and social networks.**



Newer Paper (2012)

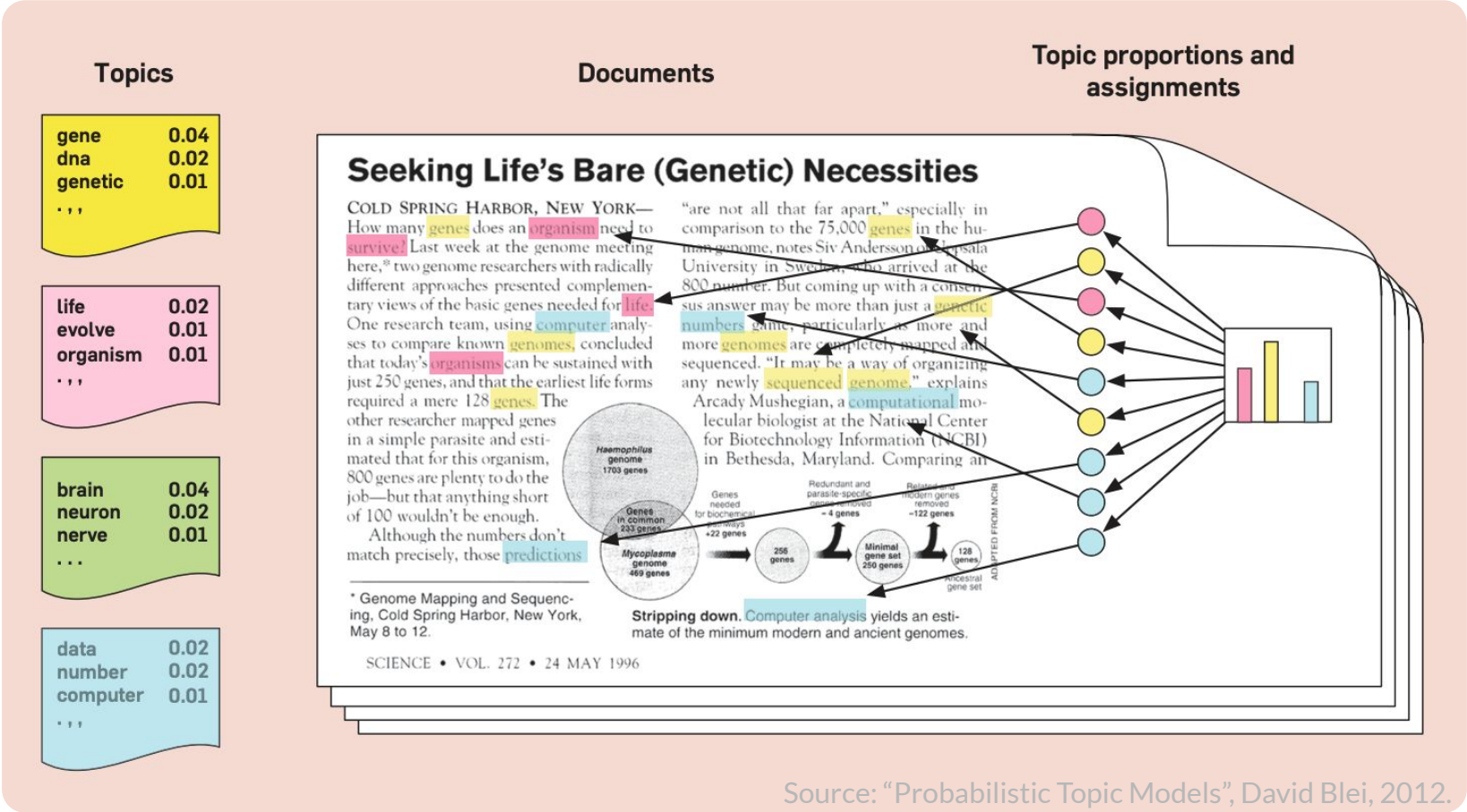
("Probabilistic Topic Models")

# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model* over documents.

# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

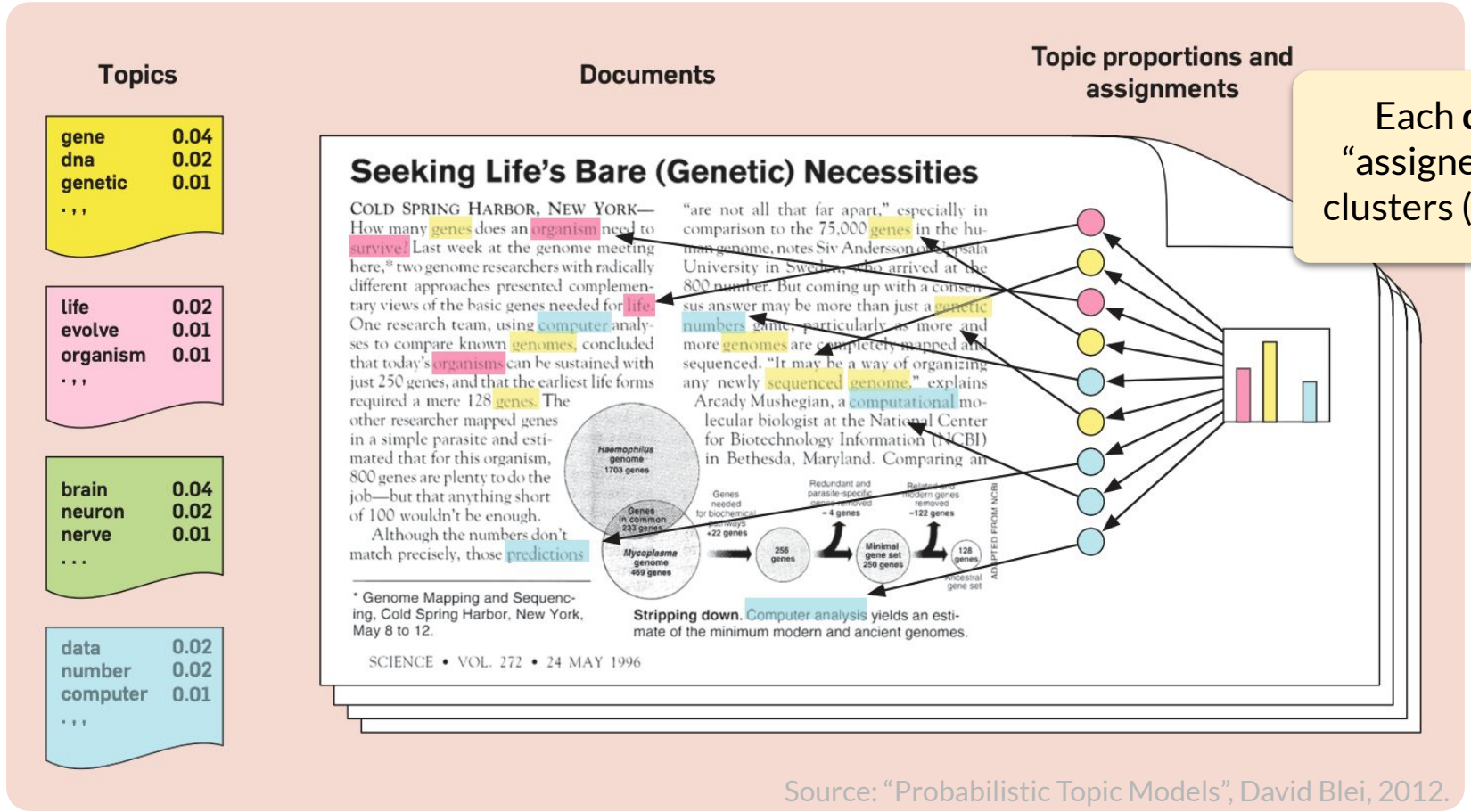
Latent Dirichlet Allocation (LDA) — an *admixture model* over documents.




Source: "Probabilistic Topic Models", David Blei, 2012.

# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model* over documents.

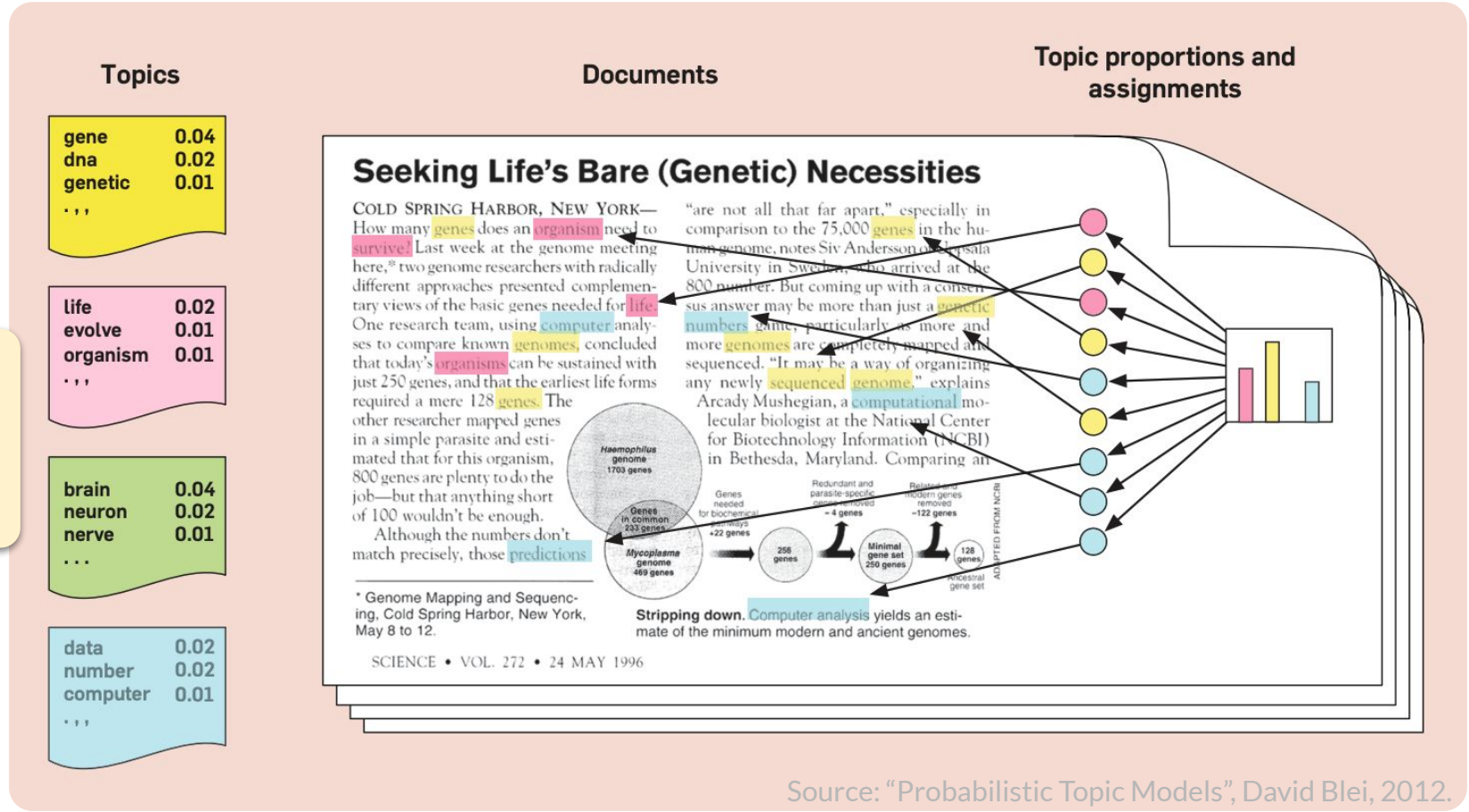



Source: "Probabilistic Topic Models", David Blei, 2012.

# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model* over documents.

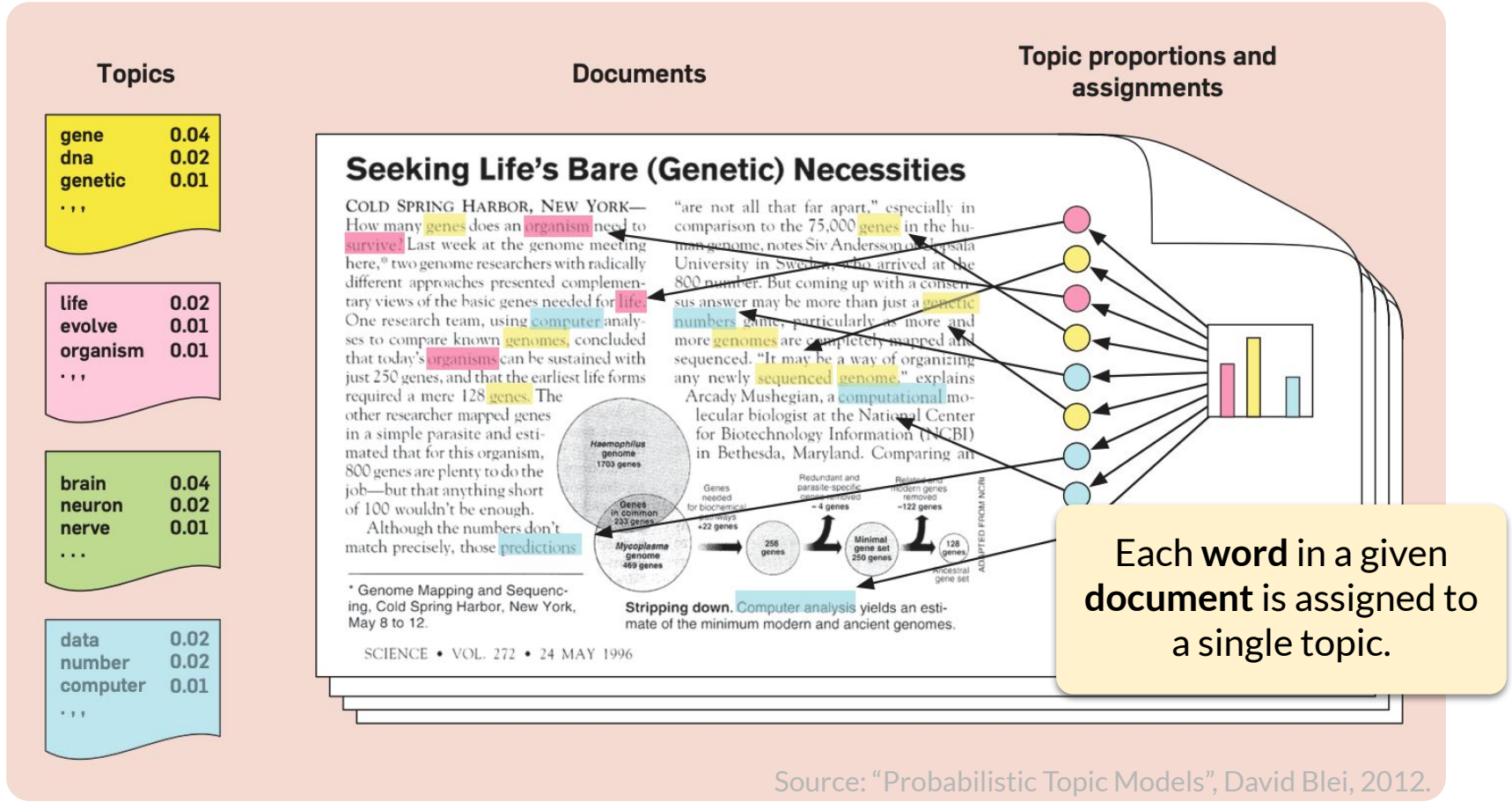
Each **topic** is a distribution over **words** (over the vocabulary)




Source: "Probabilistic Topic Models", David Blei, 2012.

# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model* over documents.

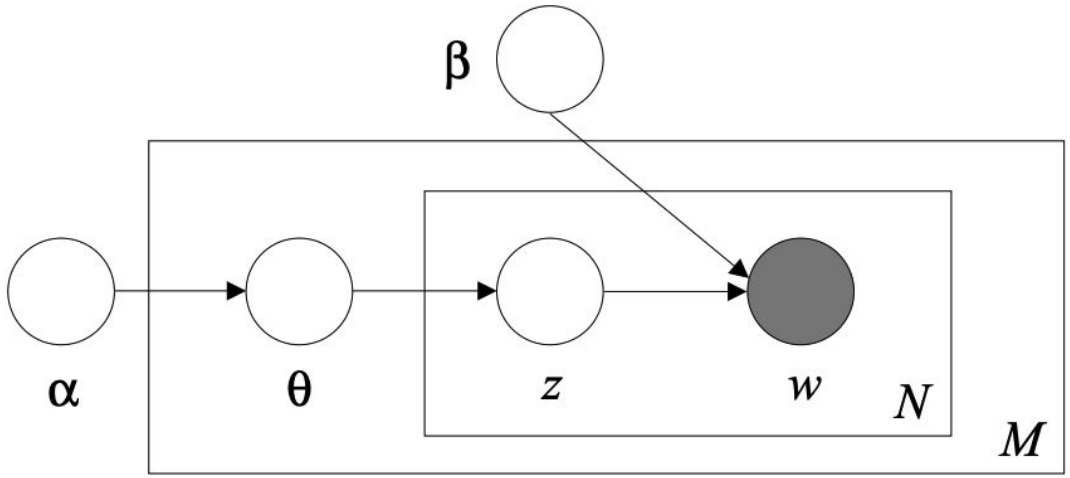



Source: "Probabilistic Topic Models", David Blei, 2012.

# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model*.

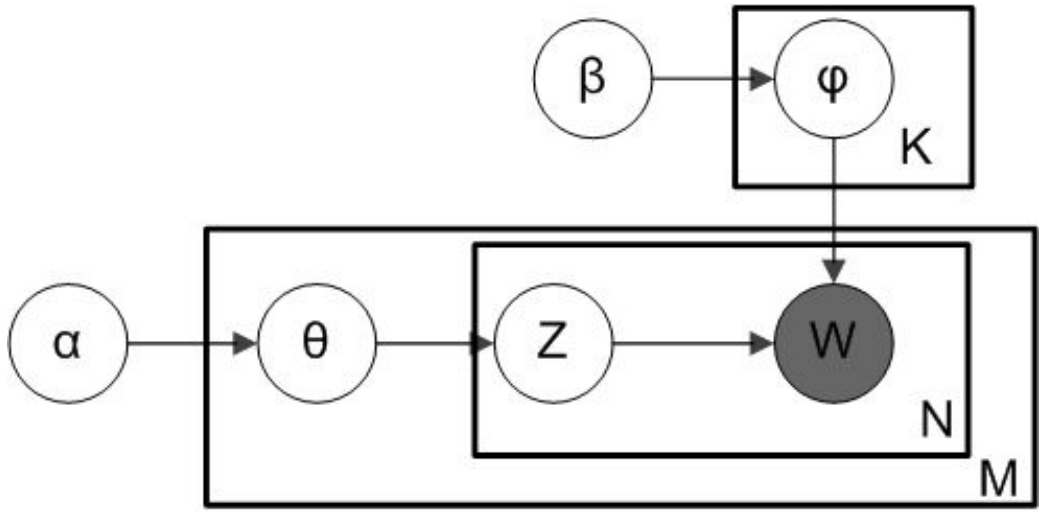
PGM from the original paper (2003).



# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model*.

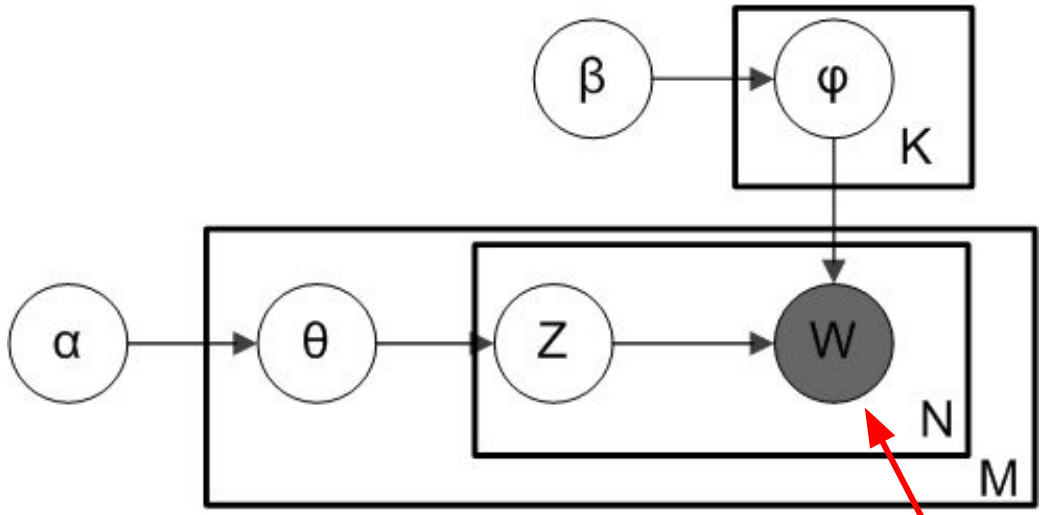
Updated slightly
(from wikipedia)



# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model*.

Updated slightly
(from wikipedia)

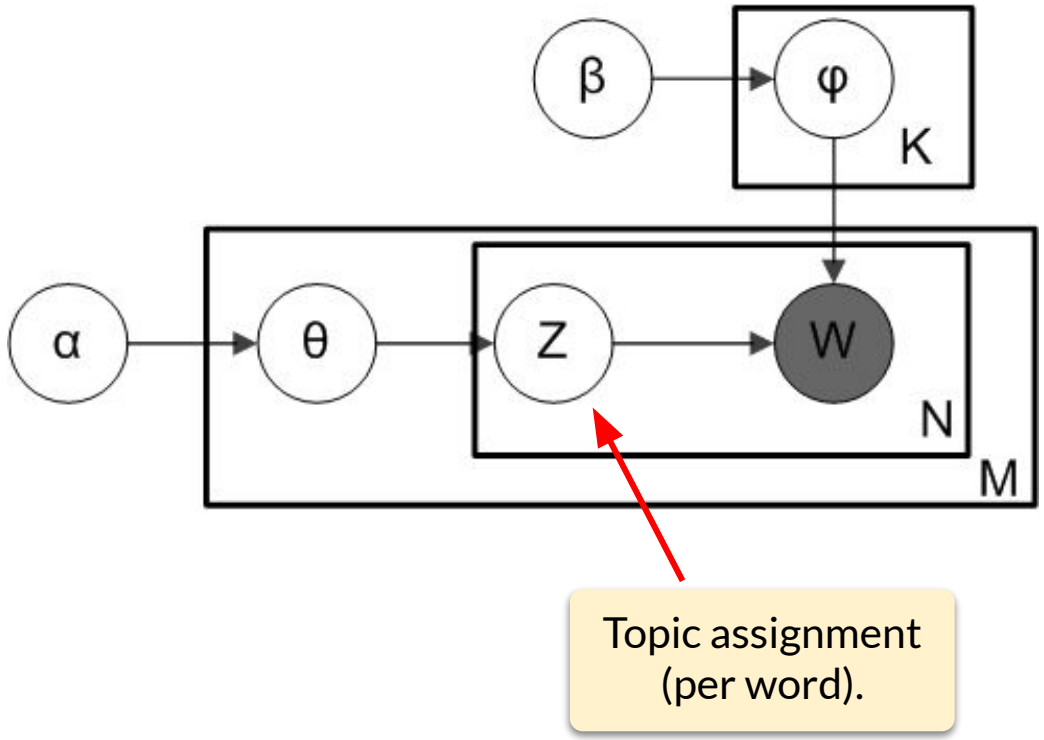



Words ($N$ words per document)

# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model*.

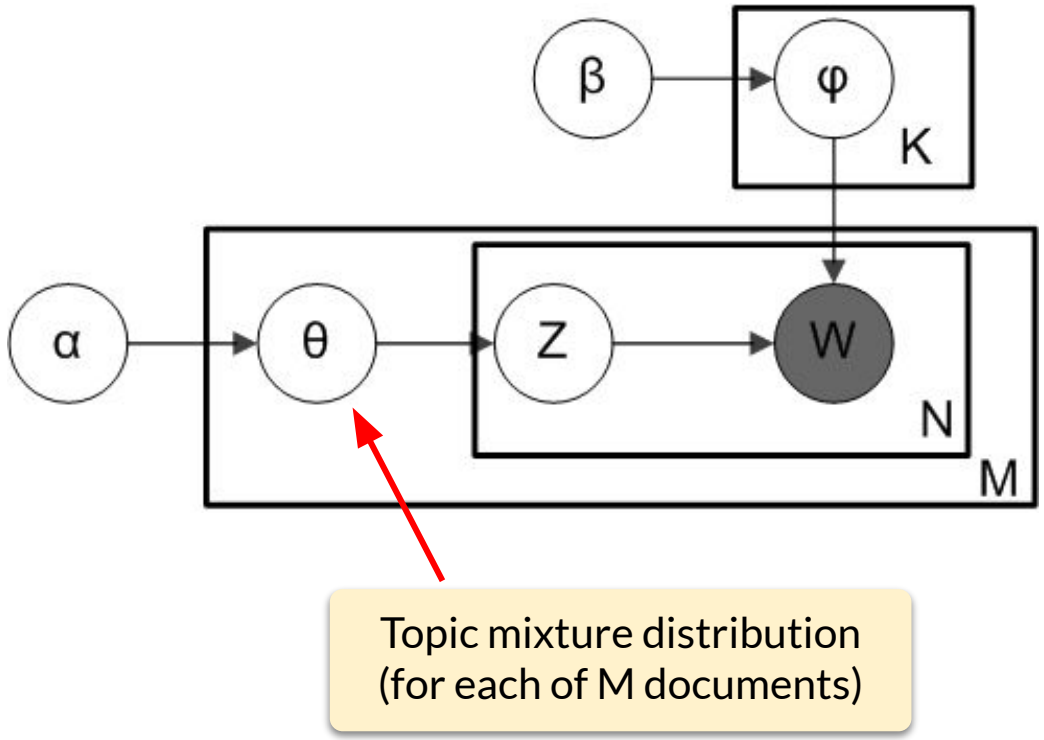
Updated slightly
(from wikipedia)



# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model*.

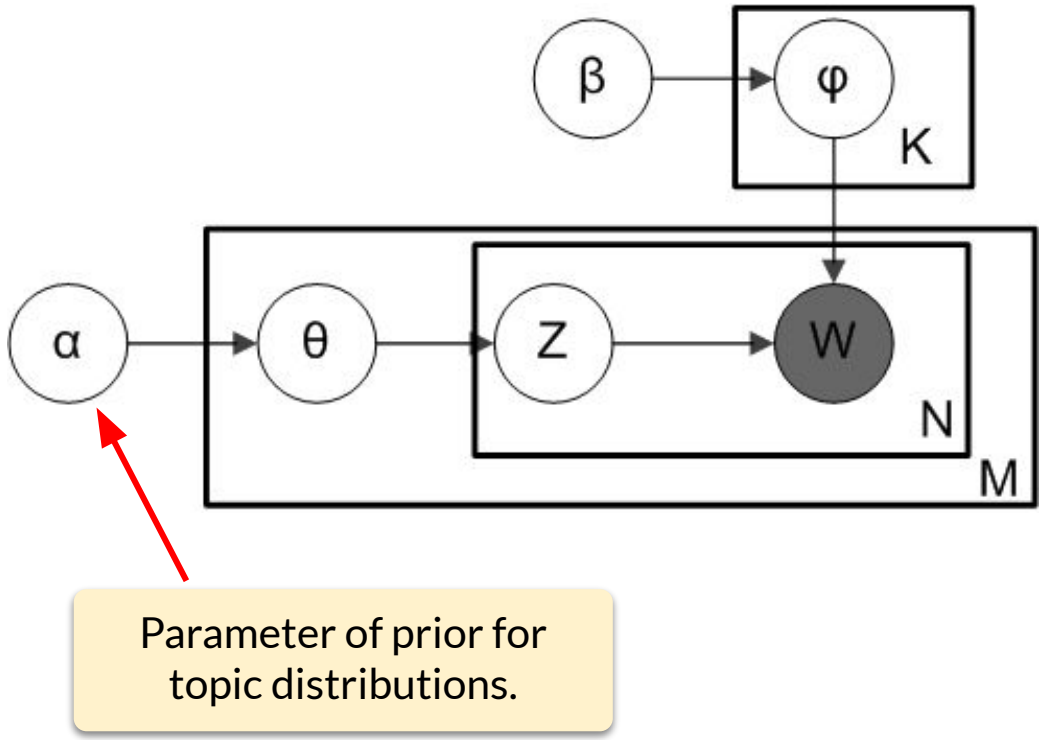
Updated slightly
(from wikipedia)



# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model*.

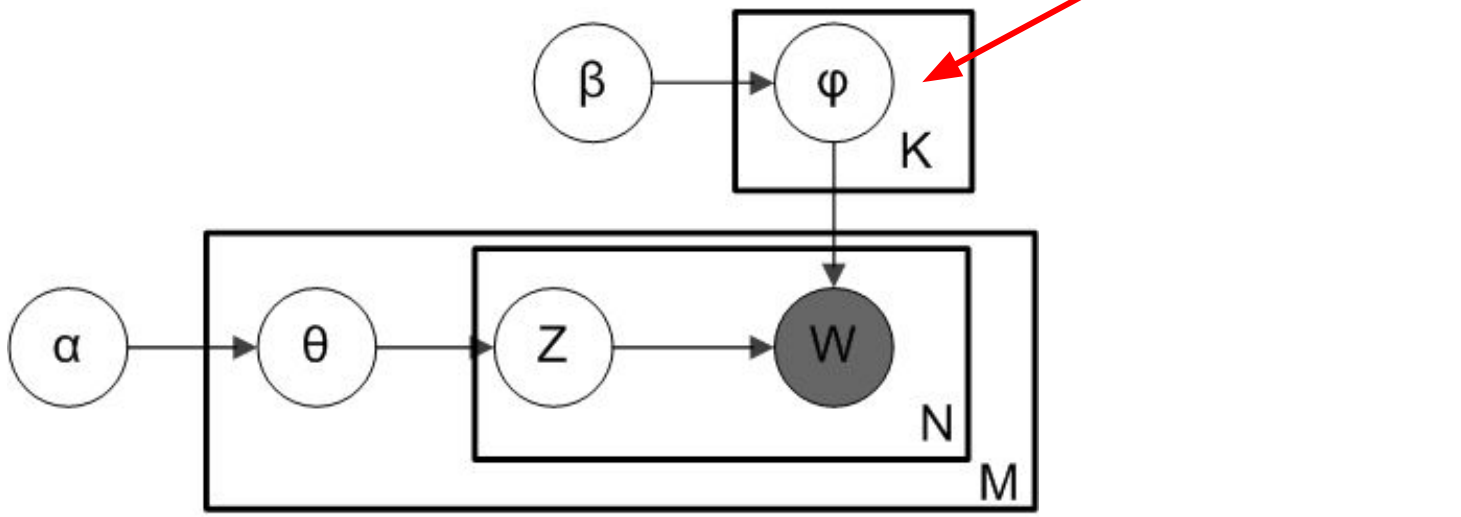
Updated slightly
(from wikipedia)



# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model*.

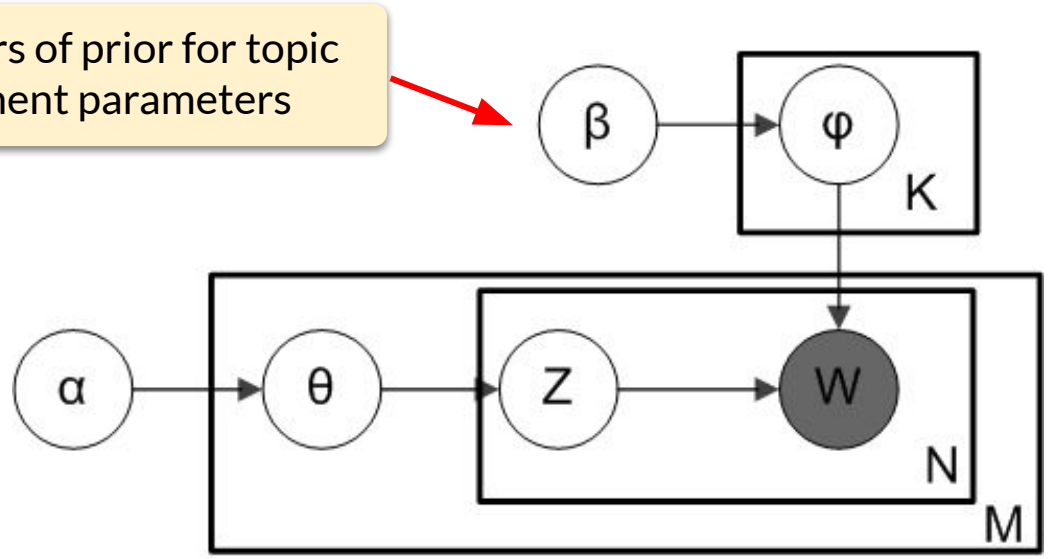
Updated slightly
(from wikipedia)



# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model*.

Updated slightly
(from wikipedia)

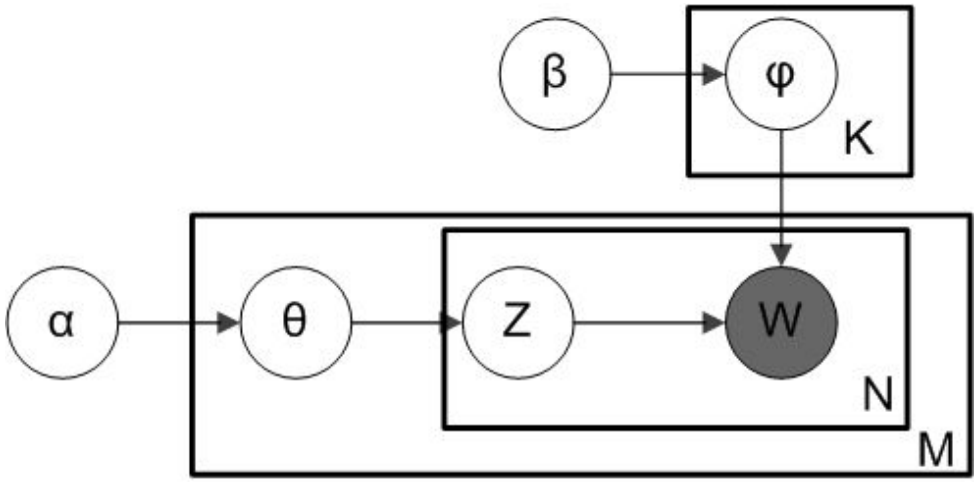


# **Famous Bayesian Networks** — Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) — an *admixture model*.

Applications:

- Information retrieval.
- Content recommendation.
- Recommender system, collaborative filtering (items>users in place of documents>words)
- Document analysis & feature extraction.



# **Famous Bayesian Networks** — Neural Transformations in PGMs

# **Famous Bayesian Networks** — Neural Transformations in PGMs

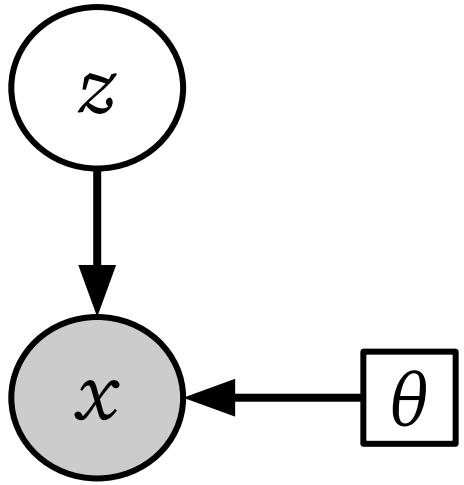
Can incorporate more-complex function transformations (or neural networks) into Bayesian networks!

# **Famous Bayesian Networks** — Neural Transformations in PGMs

Can incorporate more-complex function transformations (or neural networks) into Bayesian networks!

For example, a simple PGM might be:

$$z \sim p(z) = \mathcal{N}(0, 1)$$
$$x \sim p(x \mid z) = \mathcal{N}(z, 1)$$

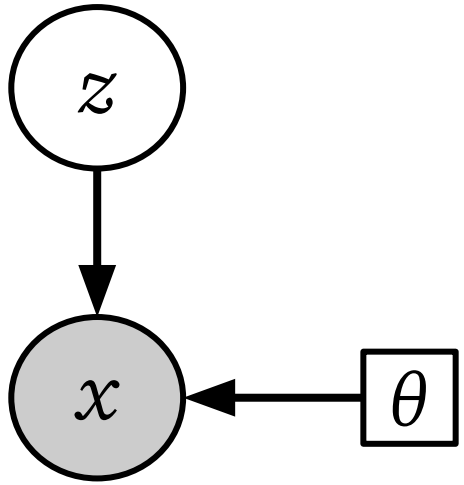


# **Famous Bayesian Networks** — Neural Transformations in PGMs

Can incorporate more-complex function transformations (or neural networks) into Bayesian networks!

But instead we could do:

$$z \sim p(z) \quad = \mathcal{N}(0, 1)$$
$$x \sim p_\theta(x \mid z) \quad = \mathcal{N}(g_\theta(z), 1)$$



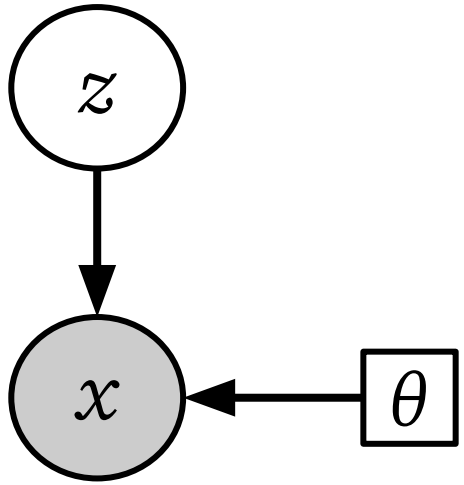
# **Famous Bayesian Networks** — Neural Transformations in PGMs

Can incorporate more-complex function transformations (or neural networks) into Bayesian networks!

But instead we could do:

$$z \sim p(z) \quad = \mathcal{N}(0, 1)$$
$$x \sim p_\theta(x \mid z) \quad = \mathcal{N}(g_\theta(z), 1)$$

$g_\theta(z)$ might be a complex fixed/known, or have parameters that we learn (*e.g.*, if we use a neural network!).



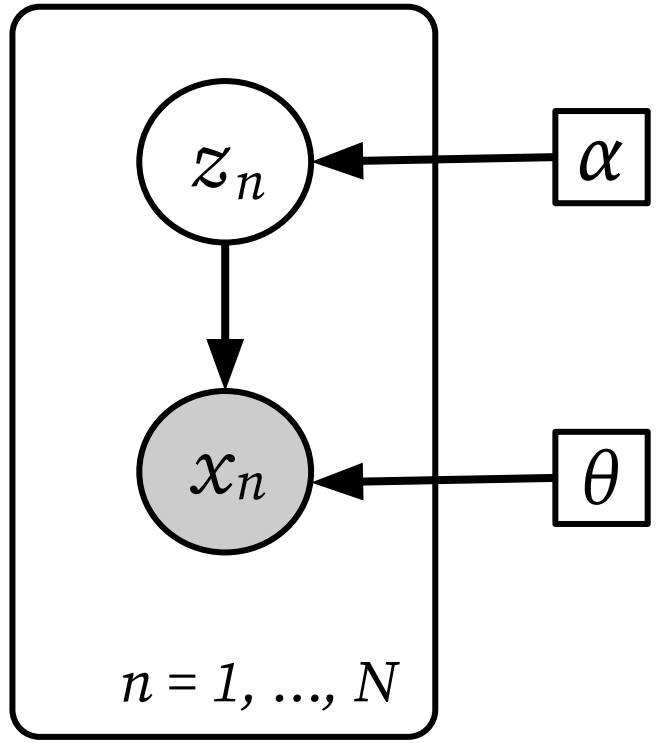
# **Famous Bayesian Networks** — Variational Autoencoder

Example: Variational Autoencoder (VAE), as a Bayesian network.

# **Famous Bayesian Networks** — Variational Autoencoder

Example: Variational Autoencoder (VAE), as a Bayesian network.

Super simple structure!



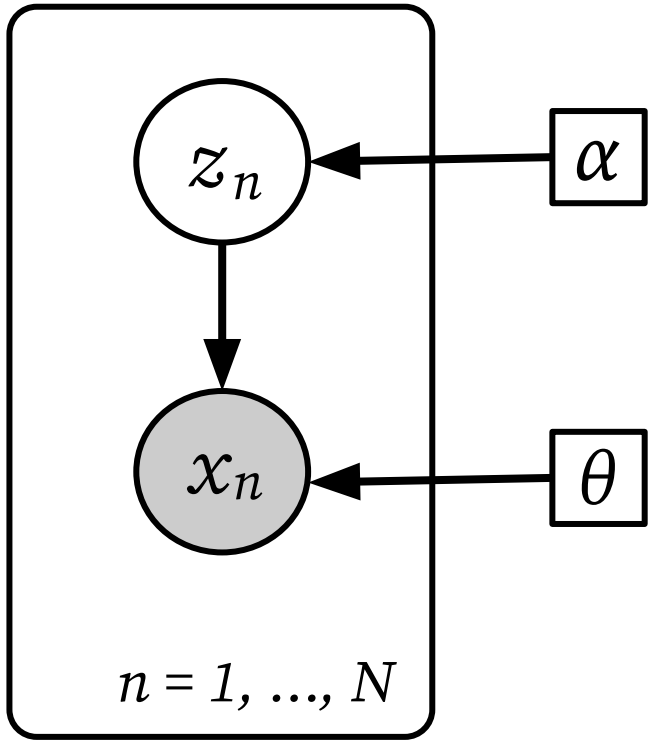
# **Famous Bayesian Networks** — Variational Autoencoder

Example: Variational Autoencoder (VAE), as a Bayesian network.

Super simple structure!

(Will cover VAEs in more detail later in the semester).

Generative process defined by a neural network — with parameters $\theta$.

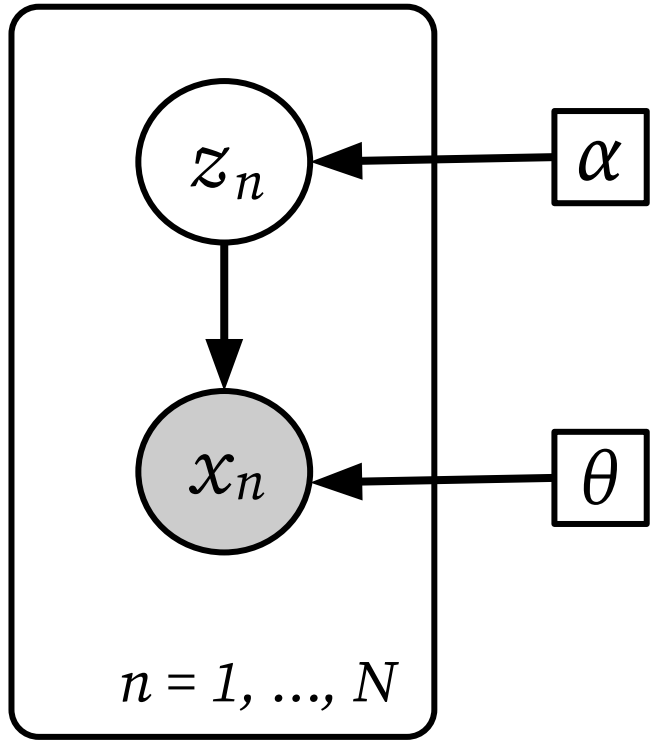


# **Famous Bayesian Networks** — Variational Autoencoder

Example: Variational Autoencoder (VAE), as a Bayesian network.

Generative process:

$$\text{for } n = 1, \ldots, N:$$
$$z_n \sim p(z \mid \alpha)$$
$$x_n \sim p_\theta(x \mid z_n)$$



# **Famous Bayesian Networks** — Variational Autoencoder

Example: Variational Autoencoder (VAE), as a Bayesian network.

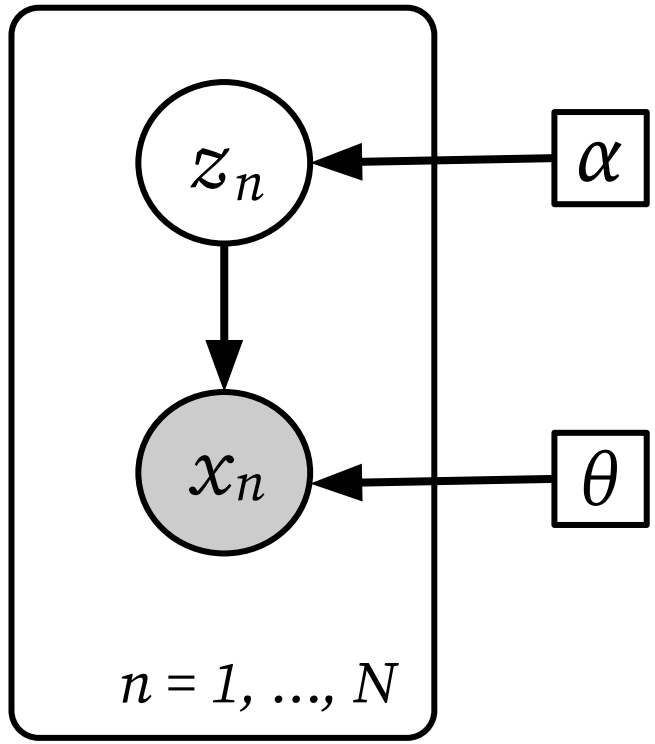
Generative process:

For each data point $n$

$$\text{for } n = 1, \ldots, N:$$

$$z_n \sim p(z \mid \alpha)$$

$$x_n \sim p_\theta(x \mid z_n)$$



# **Famous Bayesian Networks** — Variational Autoencoder

Example: Variational Autoencoder (VAE), as a Bayesian network.

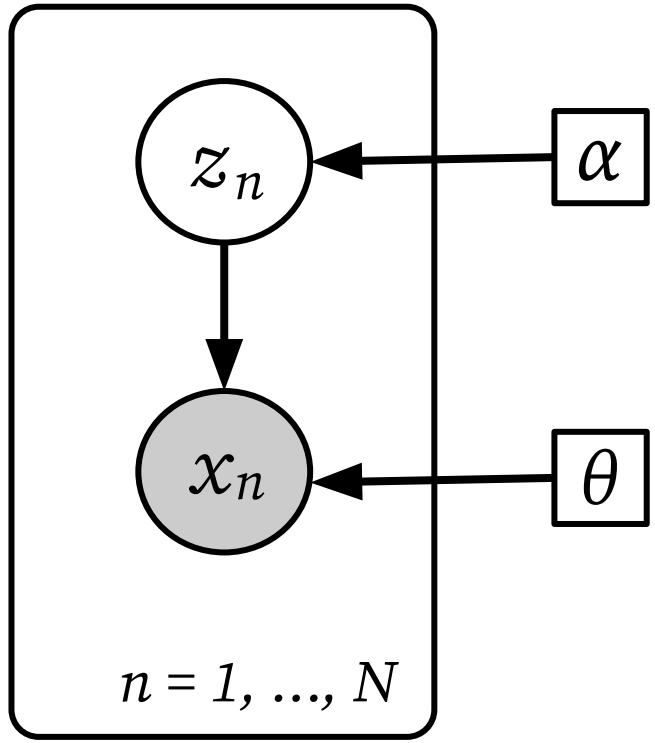
Generative process:

$$\text{for } n = 1, \ldots, N:$$

$$z_n \sim p(z \mid \alpha)$$

$$x_n \sim p_\theta(x \mid z_n)$$

Sample from prior



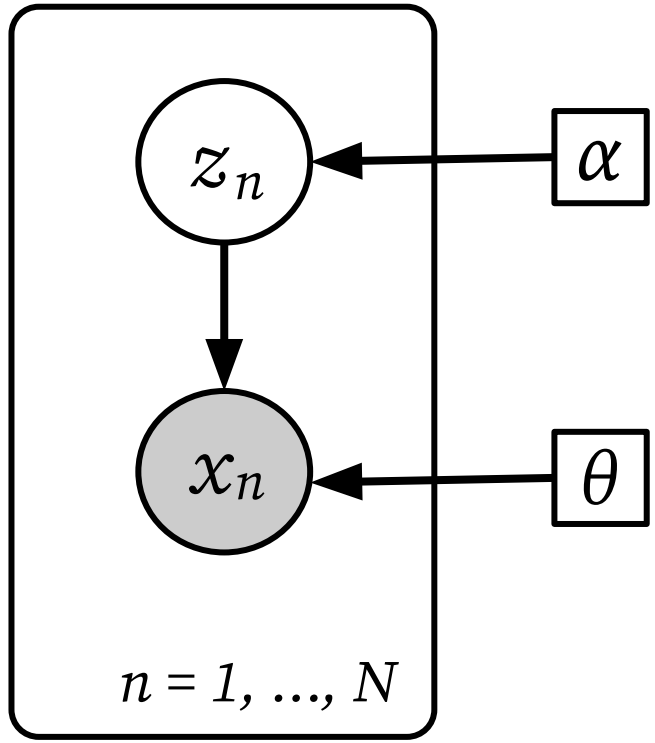
# **Famous Bayesian Networks** — Variational Autoencoder

Example: Variational Autoencoder (VAE), as a Bayesian network.

Generative process:

$$\text{for } n = 1, \ldots, N:$$
$$z_n \sim p(z \mid \alpha)$$
$$x_n \sim p_\theta(x \mid z_n)$$

Sample from distribution parameterized via $\theta$.



# **Famous Bayesian Networks** — Variational Autoencoder

Example: Variational Autoencoder (VAE), as a Bayesian network.

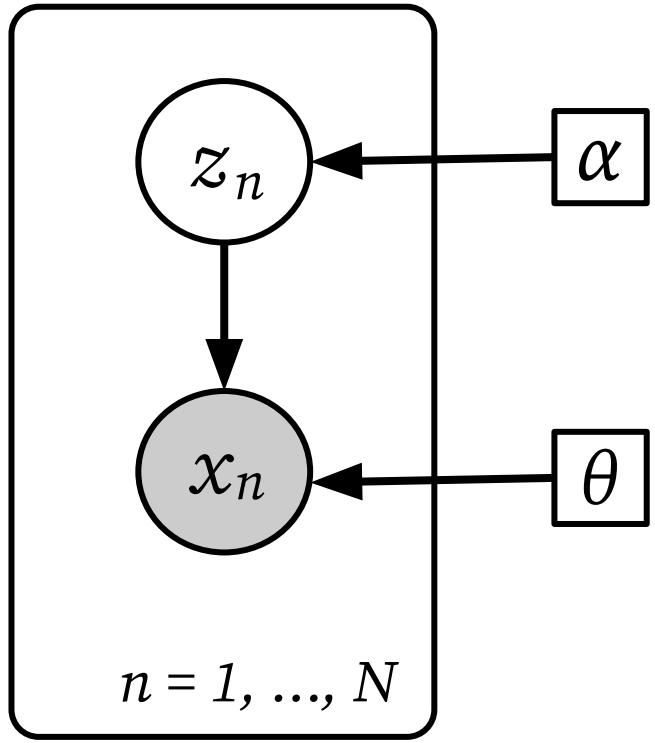
Generative process:

$$\text{for } n = 1, \ldots, N:$$

$$z_n \sim p(z \mid \alpha) \quad = \mathcal{N}(0, \alpha^2)$$

$$x_n \sim p_\theta(x \mid z_n) \quad = \mathcal{N}(\mu_\theta(z_n), \Sigma_\theta(z_n))$$

Example densities



# **Famous Bayesian Networks** — Variational Autoencoder

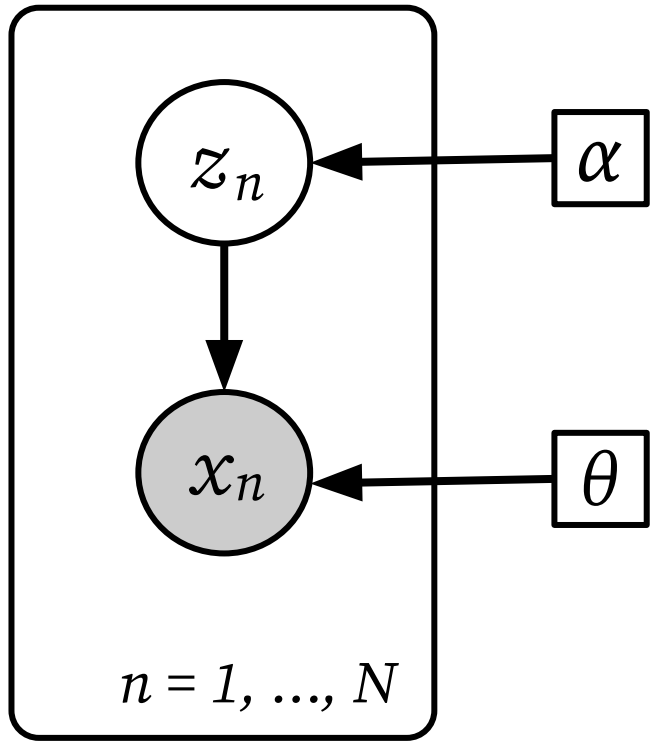
Example: Variational Autoencoder (VAE), as a Bayesian network.

Generative process:

$$\text{for } n = 1, \ldots, N:$$

$$z_n \sim p(z \mid \alpha) \quad = \mathcal{N}(0, \alpha^2)$$

$$x_n \sim p_\theta(x \mid z_n) \quad = \mathcal{N}(\mu_\theta(z_n), \Sigma_\theta(z_n))$$



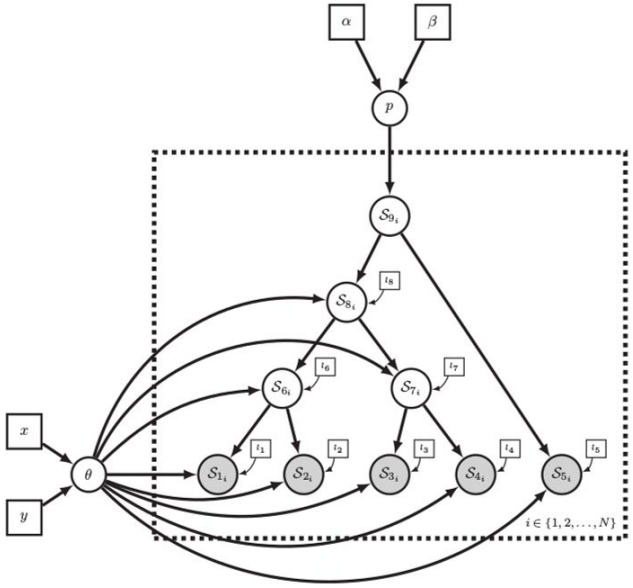

Sample from distribution parameterized via output of a **neural network.**

$\mu_\theta(z_n), \Sigma_\theta(z_n)$

# **Bayesian Networks** — Many Other Fun Examples

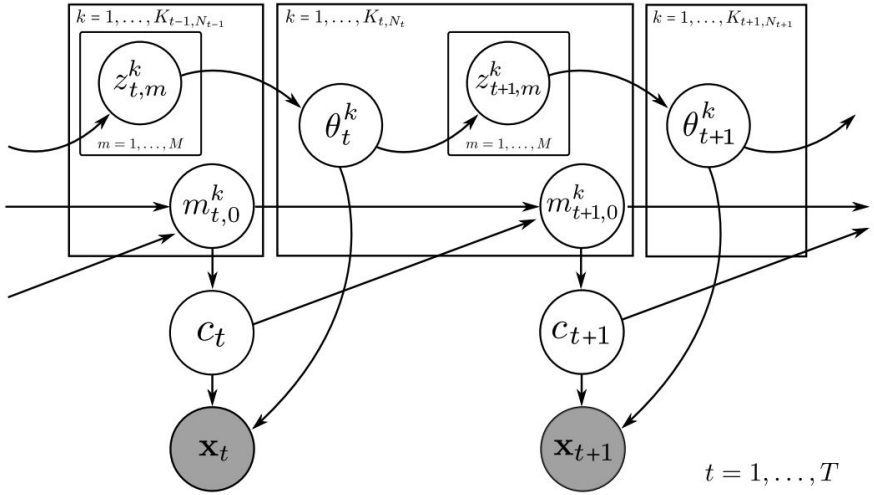
# **Bayesian Networks** — Many Other Fun Examples

## Phylogenetics



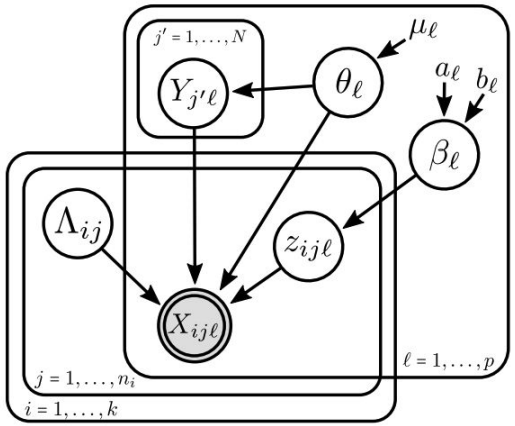
Source: "Probabilistic Graphical Model Representation in Phylogenetics", Höhna 2014

## Object Tracking (in videos)



Source: "The Dependent Dirichlet Process Mixture of Objects for Detection-free Tracking and Object Modeling", Neiswanger et al., 2014

## Record Linkage (in databases)



Source: "Performance Bounds for Graphical Record Linkage", Steorts et al., 2014

# **Bayesian Networks** — Summary

# **Bayesian Networks** — Summary

Bayesian network is define by a tuple $(G = \{V, E\}, P)$ where $P$ is specified as a set of local conditional probability distributions (CPDs) associated with nodes $V$

⇒ Efficient representation using a graph-based data structure.

Computing joint probability of any assignment obtained by multiplying CPDs.

Can sample from joint by sampling from CPDs according to DAG ordering.

Can identify some conditional independencies by looking at graph properties.

Nodes in graph can correspond to random variables or random vectors.

# Undirected PGMs:  Markov Random Fields

# **Undirected PGMs** — Introduction

# **Undirected PGMs** — Introduction

Bayesian network can represent a rich set of probability distributions (and conditional independence assumptions).

However, some distributions have independence assumptions that cannot be captured exactly.

# **Undirected PGMs** — Introduction

Bayesian network can represent a rich set of probability distributions (and conditional independence assumptions).

However, some distributions have independence assumptions that cannot be captured exactly.

**Example**. Consider the generative process:

$$X, Y \sim \text{Bernoulli}(0.5)$$
$$Z = X \text{ xor } Y$$

# **Undirected PGMs** — Introduction

Bayesian network can represent a rich set of probability distributions (and conditional independence assumptions).

However, some distributions have independence assumptions that cannot be captured exactly.

**Example**. Consider the generative process:

$$X, Y \sim \text{Bernoulli}(0.5)$$
$$Z = X \text{ xor } Y$$

1 if X and Y are different.
0 otherwise.

# **Undirected PGMs** — Introduction

Bayesian network can represent a rich set of probability distributions (and conditional independence assumptions).

However, some distributions have independence assumptions that cannot be captured exactly.

**Example**. Consider the generative process:

$$X, Y \sim \text{Bernoulli}(0.5)$$
$$Z = X \text{ xor } Y$$
$$\Rightarrow \{X \perp Y, \ X \perp Z, \ Y \perp Z\}$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Undirected PGMs** — Introduction

Bayesian network can represent a rich set of probability distributions (and conditional independence assumptions).

However, some distributions have independence assumptions that cannot be captured exactly.

**Example**. Consider the generative process:

$$X, Y \sim \text{Bernoulli}(0.5)$$
$$Z = X \text{ xor } Y$$

$\Rightarrow \{X \perp Y, \ X \perp Z, \ Y \perp Z\}$

But $Z$ is not independent of $\{X, Y\}$.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Undirected PGMs** — Introduction

Bayesian network can represent a rich set of probability distributions (and conditional independence assumptions).

However, some distributions have independence assumptions that cannot be captured exactly.

**Example**. Consider the generative process:

$$X, Y \sim \text{Bernoulli}(0.5)$$
$$Z = X \text{ xor } Y$$

$\Rightarrow \{X \perp Y, \ X \perp Z, \ Y \perp Z\}$

But $Z$ is not independent of $\{X, Y\}$.

Called “noisy xor” example.

# **Undirected PGMs** — Introduction

Bayesian network can represent a rich set of probability distributions (and conditional independence assumptions).

However, some distributions have independence assumptions that cannot be captured exactly.

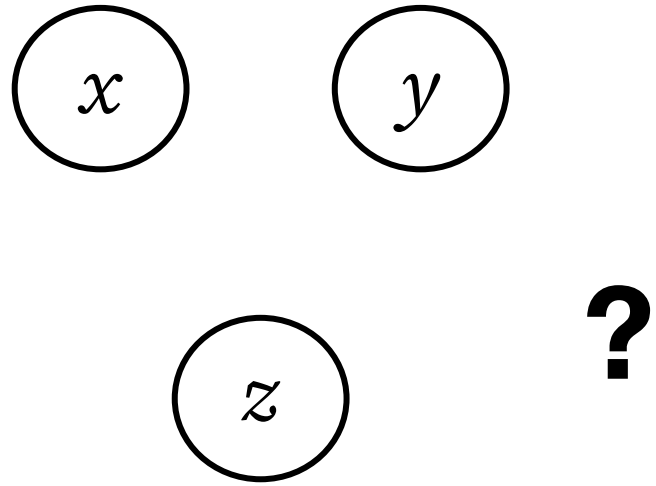
**Example**. Consider the generative process:

$$X, Y \sim \text{Bernoulli}(0.5)$$
$$Z = X \text{ xor } Y$$

$\Rightarrow \{X \perp Y, \ X \perp Z, \ Y \perp Z\}$

But $Z$ is not independent of $\{X, Y\}$.

**What graph structure to use?**




Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Undirected PGMs** — Introduction

Bayesian network can represent a rich set of probability distributions (and conditional independence assumptions).

However, some distributions have independence assumptions that cannot be captured exactly.

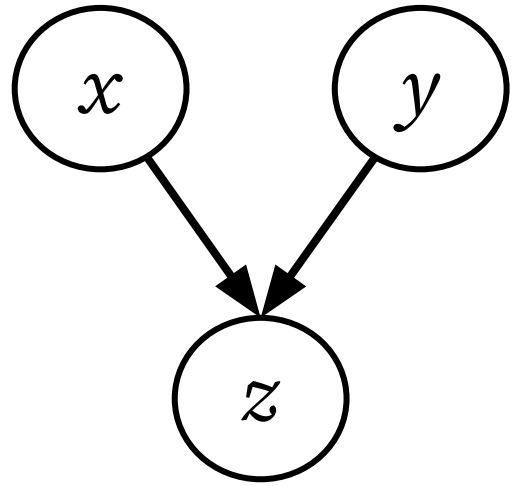
**Example**. Consider the generative process:

$$X, Y \sim \text{Bernoulli}(0.5)$$
$$Z = X \text{ xor } Y$$

$\Rightarrow \{X \perp Y, \ X \perp Z, \ Y \perp Z\}$

But $Z$ is not independent of $\{X, Y\}$.

**What graph structure to use?**




?

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Undirected PGMs** — Introduction

Bayesian network can represent a rich set of probability distributions (and conditional independence assumptions).

However, some distributions have independence assumptions that cannot be captured exactly.

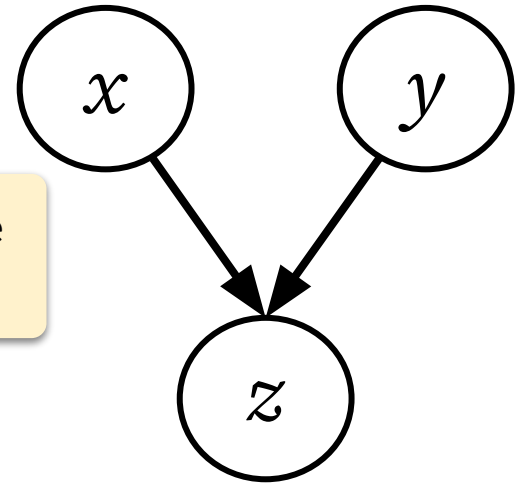
**Example**. Consider the generative process:

$$X, Y \sim \text{Bernoulli}(0.5)$$
$$Z = X \text{ xor } Y$$

$$\Rightarrow \{X \perp Y, \ X \perp Z, \ Y \perp Z\}$$

But $Z$ is **not** independent of $\{X, Y\}$.

**What graph structure to use?**




PGM does **not** capture all independencies!

?

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Undirected PGMs** — Introduction

Bayesian network can represent a rich set of probability distributions (and conditional independence assumptions).

However, some distributions have independence assumptions that cannot be captured exactly.

Furthermore, some probability distributions (*i.e.*, models of the real world) are more easy to define using alternative abstractions…

# **Undirected PGMs** — Introduction

There exists an alternative **“visual language”** for compactly representing and visualizing a probability distribution that is based on **undirected graphs**.

# **Undirected PGMs** — Introduction

There exists an alternative **"visual language"** for compactly representing and visualizing a probability distribution that is based on **undirected graphs**.

These are known as Markov Random Fields (MRFs).

In some cases, they can capture conditional independence assumptions that Bayesian networks cannot represent.

(But come with their own set of drawbacks as well)

# **Markov Random Fields** — Motivating Example

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Motivating Example

As a motivating example…

Suppose that we are modeling voting preferences among people: $A$ , $B$ , $C$ , $D$

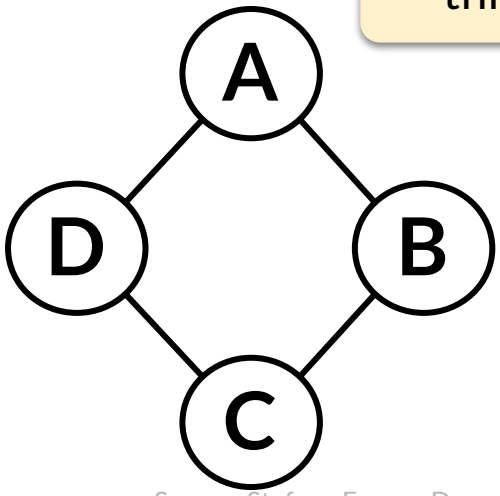
# **Markov Random Fields** — Motivating Example

As a motivating example…

Suppose that we are modeling voting preferences among people: $A$ , $B$ , $C$ , $D$

Let's say that $( A , B )$, $( B , C )$, $( C , D )$, and $( D , A )$ are friends.

Naturally represented by this graph.




Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Motivating Example

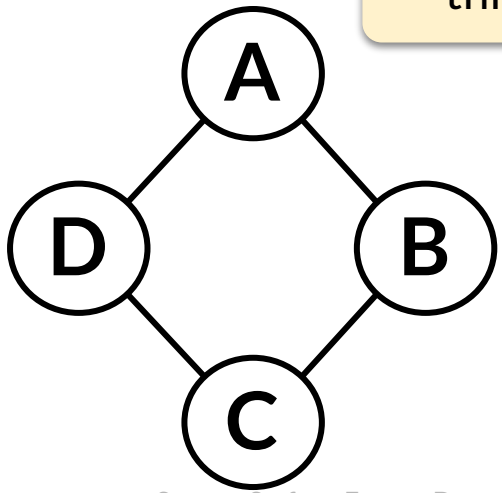
As a motivating example…

Suppose that we are modeling voting preferences among people: $A$ , $B$ , $C$ , $D$

Let's say that $(A, B)$, $(B, C)$, $(C, D)$, and $(D, A)$ are friends.

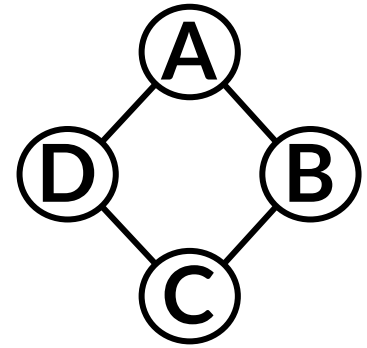
And friends tend to have **similar voting preferences**.

Naturally represented by this graph.



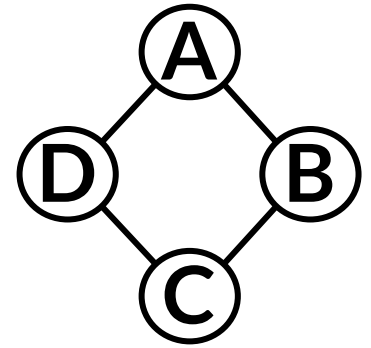

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Motivating Example




One way to define a probability over the joint voting decision of $A$ , $B$ , $C$ , $D$ is:

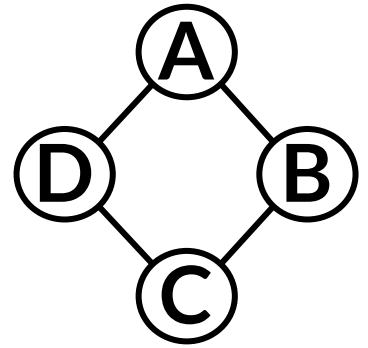
# **Markov Random Fields** — Motivating Example




One way to define a probability over the joint voting decision of $A$ , $B$ , $C$ , $D$ is:

- Assign scores to each assignment of these variables, then define probability as the normalized score.

# **Markov Random Fields** — Motivating Example




One way to define a probability over the joint voting decision of $A$ , $B$ , $C$ , $D$ is:

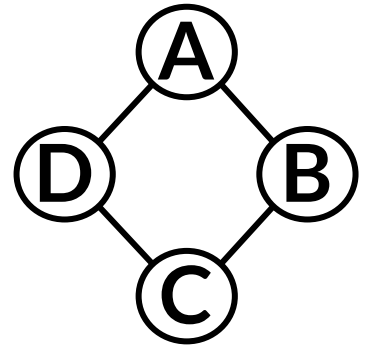
- Assign scores to each assignment of these variables, then define probability as the normalized score.
- A score can be any function, but we'll define it to be of the form:

$$\tilde{p}(A, B, C, D) = \phi(A, B)\ \phi(B, C)\ \phi(C, D)\ \phi(D, A)$$

- Where $\phi(X, Y)$ is a *factor* that assigns more weight to consistent vote among friends $X, Y$.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Motivating Example




One way to define a probability over the joint voting decision of $A$ , $B$ , $C$ , $D$ is:

- Assign scores to each assignment of these variables, then define probability as the normalized score.
- A score can be any function, but we'll define it to be of the form:

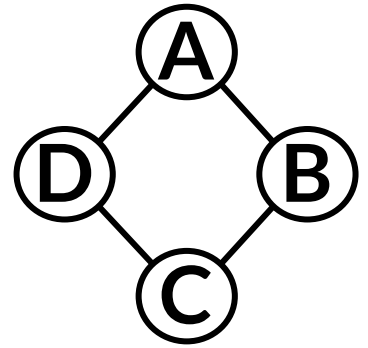
$$\tilde{p}(A, B, C, D) = \phi(A, B)\ \phi(B, C)\ \phi(C, D)\ \phi(D, A)$$

- Where $\phi(X, Y)$ is a *factor* that assigns more weight to consistent vote among friends $X, Y$.
- For example:

$$\phi(X, Y) = \begin{cases} 10 & \text{if } X = Y = 1 \\ 5 & \text{if } X = Y = 0 \\ 1 & \text{otherwise.} \end{cases}$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Motivating Example

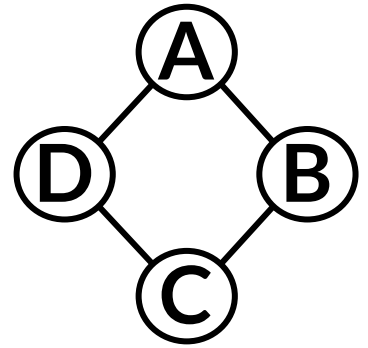



The final probability is then defined as

$$p(A, B, C, D) = \frac{1}{Z}\tilde{p}(A, B, C, D)$$

Where $Z = \sum_{A,B,C,D}\tilde{p}(A, B, C, D)$ is a *normalizing constant* that ensures the probability distribution sums to one.

# **Markov Random Fields** — Motivating Example

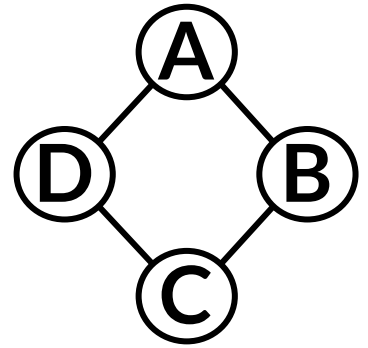



The final probability is then defined as

$$p(A,B,C,D) = \frac{1}{Z}\,\tilde{p}(A,B,C,D) = \frac{1}{Z}\,\phi(A,B)\;\phi(B,C)\;\phi(C,D)\;\phi(D,A)$$

Where $Z = \sum_{A,B,C,D} \tilde{p}(A,B,C,D)$ is a *normalizing constant* that ensures the probability distribution sums to one.

# **Markov Random Fields** — Motivating Example




The final probability is then defined as

$$p(A, B, C, D) = \frac{1}{Z}\,\tilde{p}(A, B, C, D) = \frac{1}{Z}\,\phi(A, B)\;\phi(B, C)\;\phi(C, D)\;\phi(D, A)$$

Where $Z = \sum_{A,B,C,D} \tilde{p}(A, B, C, D)$ is a *normalizing constant* that ensures the probability distribution sums to one.

When normalized, we can view

- Factor $\phi(A, B)$ as an *interaction* pushing $B$'s vote closer to that of $A$.
- And can view $\phi(B, C)$ as an *interaction* pushing $B$'s vote closer to $C$, etc.
- And the most-likely vote will require reconciling these conflicting influences.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Differences with Bayesian Networks

Note a few differences between Markov Random Fields and Bayesian Networks:

# **Markov Random Fields** — Differences with Bayesian Networks

Note a few differences between Markov Random Fields and Bayesian Networks:

- In MRFs, we are not saying anything about how one variable is generated from another.
- Simply indicate a level of coupling between dependent variables in a graph.
- Don’t have to specify a full generative story ⇒ could say it requires less prior knowledge.
- In turn this defines an *energy landscape* (unnormalized PDF), which is then converted into a probability (joint) distribution via the normalization constant.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Formal Definition

# **Markov Random Fields** — Formal Definition

Definition. A Markov Random Field (MRF) is a probability distribution $p$ over variables $x_1, \ldots, x_n$ defined by an undirected graph $G$, with the form

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c)$$

where $C$ denotes the set of cliques (*i.e.*, fully connected subgraphs) of $G$, and each factor $\phi_c$ is a non-negative function over the variables in a clique.

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Formal Definition

Definition. A Markov Random Field (MRF) is a probability distribution $p$ over variables $x_1, \ldots, x_n$ defined by an undirected graph $G$, with the form

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c)$$

where $C$ denotes the set of cliques (*i.e.*, fully connected subgraphs) of $G$, and each factor $\phi_c$ is a non-negative function over the variables in a clique.

And the *partition function* $Z$ ensures that the distribution sums to one:

$$Z = \sum_{x_1, \ldots, x_n} \prod_{c \in C} \phi_c(x_c)$$

Source: Stefano Ermon, Deep Generative Models (CS236) Class

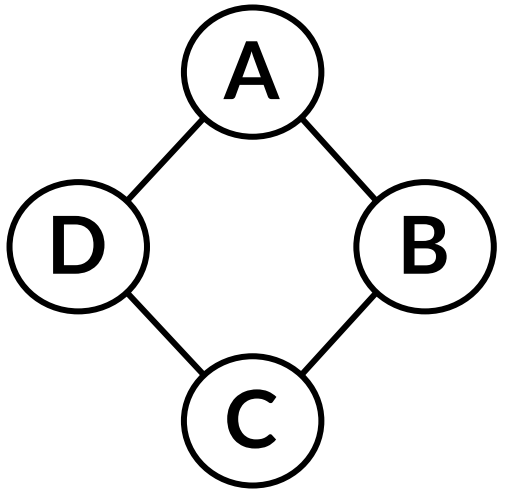
# **Markov Random Fields** — A few notes on cliques

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — A few notes on cliques

Given a graph $G$, our probability distribution may contain factors whose scope is **any clique** in $G$.

Including: a single node, a pair of connected nodes, a triangle, etc.



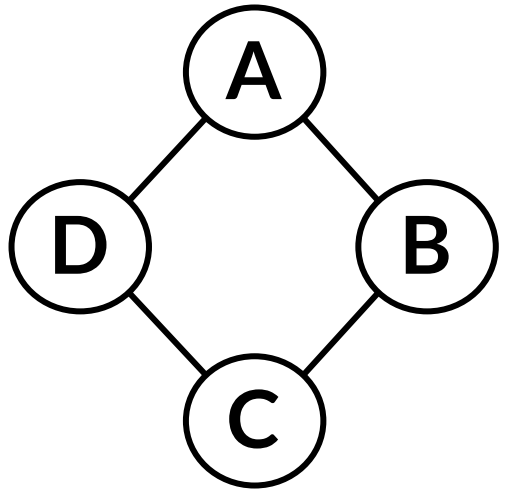

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — A few notes on cliques

Given a graph $G$, our probability distribution may contain factors whose scope is **any clique** in $G$.

Including: a single node, a pair of connected nodes, a triangle, etc.

*E.g.*, in friends example, could have specified factors defined on individual nodes (*unary factors*), but chose not to.



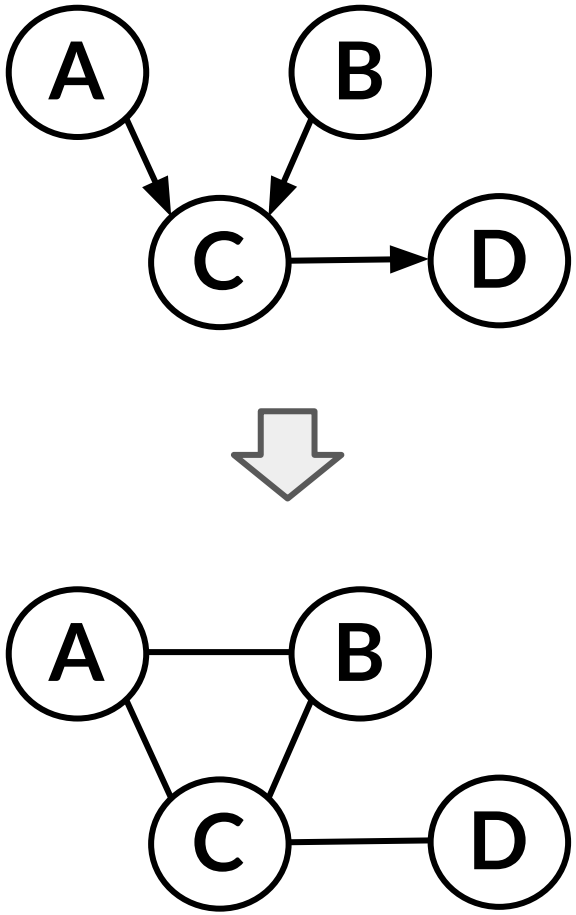

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Conversion Between BN and MRF

# **Markov Random Fields** — Conversion Between BN and MRF

Bayesian networks are a special case of MRFs with a very specific type of clique factor.

⇒ Can convert from BN to MRF:



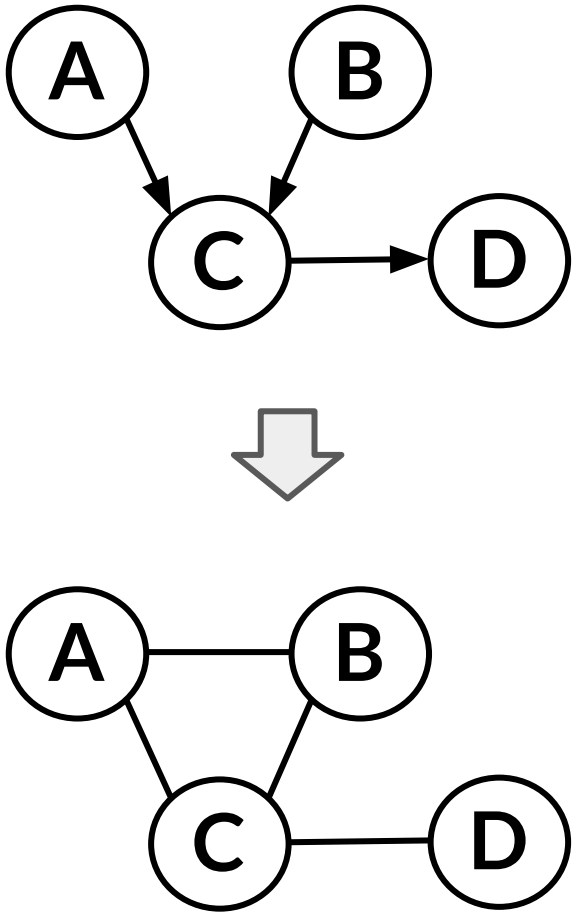

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Conversion Between BN and MRF

Bayesian networks are a special case of MRFs with a very specific type of clique factor.

⇒ Can convert from BN to MRF:

- If we take a directed graph $G$ and add "side edges" to all parents of a given node (and removing their directionality)
- This is called "*moralization*".



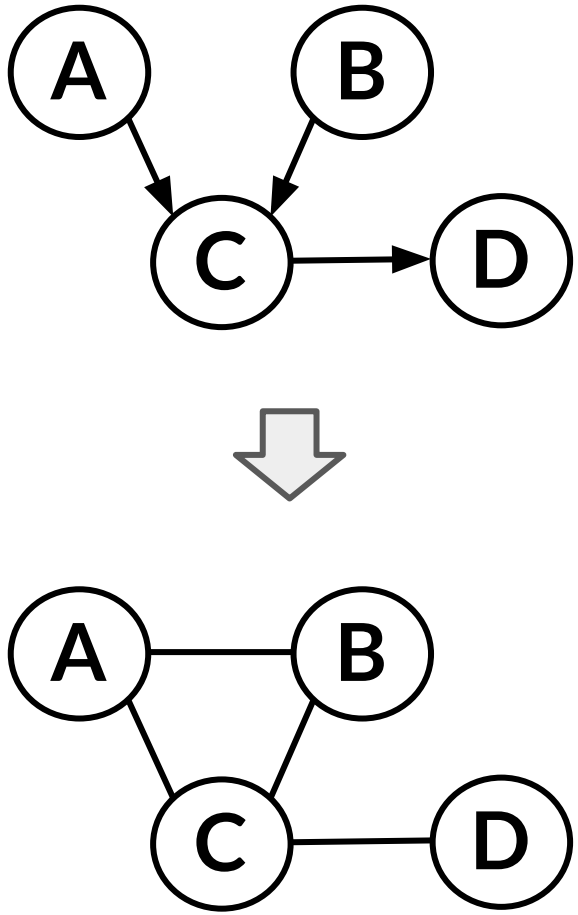

Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Conversion Between BN and MRF

Bayesian networks are a special case of MRFs with a very specific type of clique factor.

⇒ Can convert from BN to MRF:

- If we take a directed graph $G$ and add "side edges" to all parents of a given node (and removing their directionality)
- This is called "*moralization*".
- Note: There is also a process for converting from MRF to BN, but you may lose some (conditional) independencies!




Source: Stefano Ermon, Deep Generative Models (CS236) Class

# **Markov Random Fields** — Drawbacks

However, MRFs also have potential drawbacks:

# **Markov Random Fields** — Drawbacks

However, MRFs also have potential drawbacks:

- Computing the normalization constant $Z$ requires summing over a potentially exponential number of assignments.
- ⇒ In the general case, this is NP-hard; thus many undirected models will be intractable and will require approximation techniques.
- Undirected models may be difficult to interpret.
- It is much easier to generate data from a Bayesian network, which is important in some applications.

# Some Famous Markov Random Fields

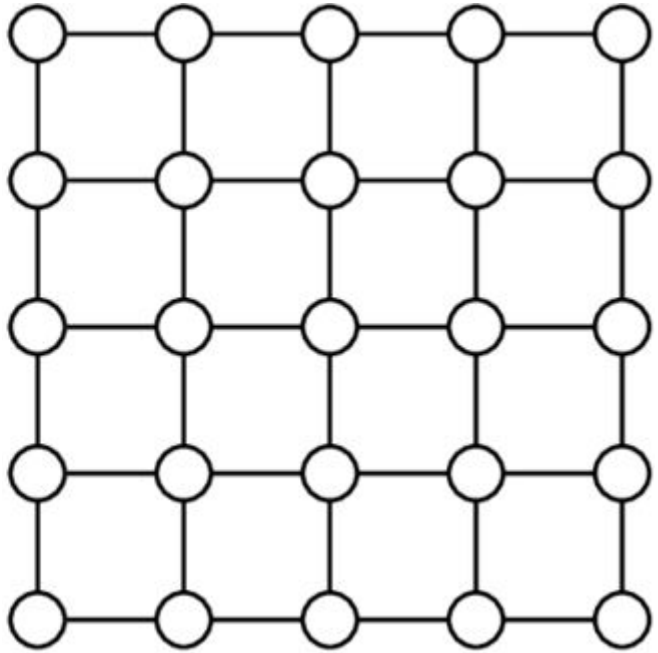
We'll go through a couple of famous MRFs

# **Some Famous Markov Random Fields** — Ising Model

# **Some Famous Markov Random Fields** — Ising Model

Originally a model of ferromagnetism in statistical mechanics!

**Markov Random Field**



# **Some Famous Markov Random Fields** — Ising Model

Originally a model of ferromagnetism in statistical mechanics!

MRF is very simple:

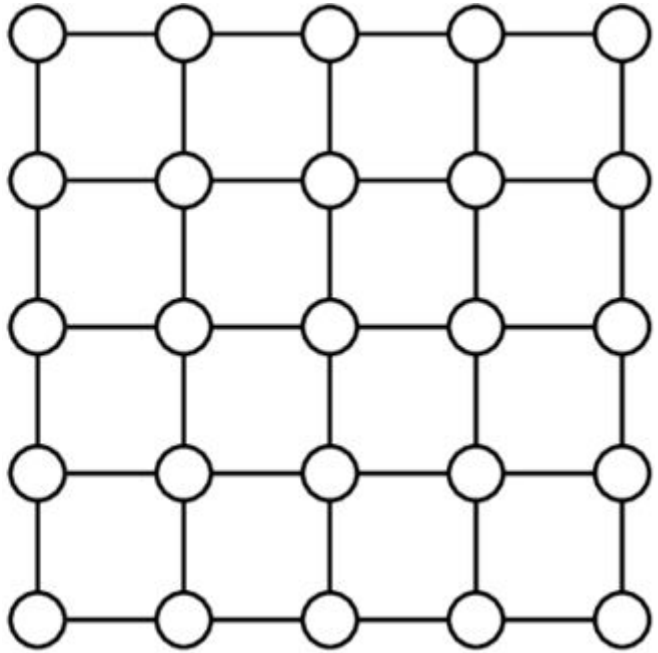
A lattice of nodes (*"grid graph"*).

Each typically a binary random variable.

- Represent "spins" that can be in one of two states.

Neighboring spins that agree have a lower energy than those that disagree.

**Markov Random Field**

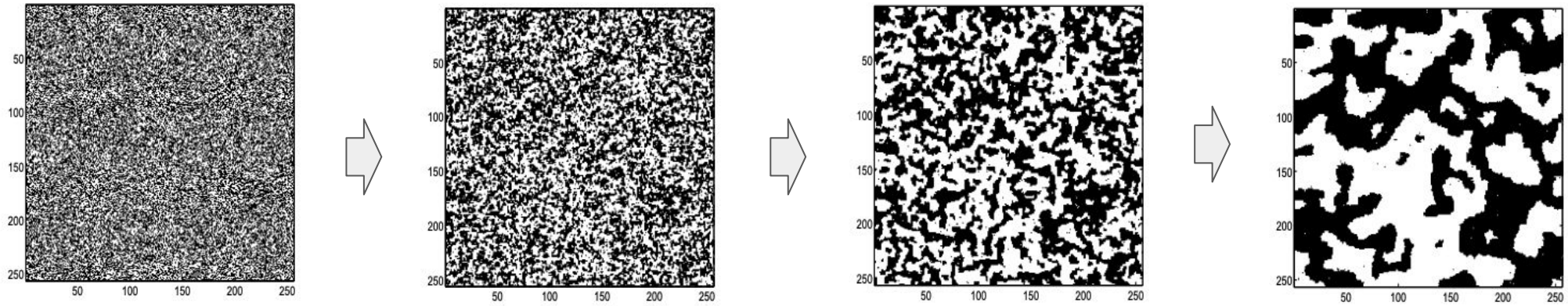


# **Some Famous Markov Random Fields** — Ising Model

Simulating samples from the joint PDF.

# **Some Famous Markov Random Fields** — Ising Model

Simulating samples from the joint PDF.




Source: Brani Vidakovic, “Markov Random Fields”

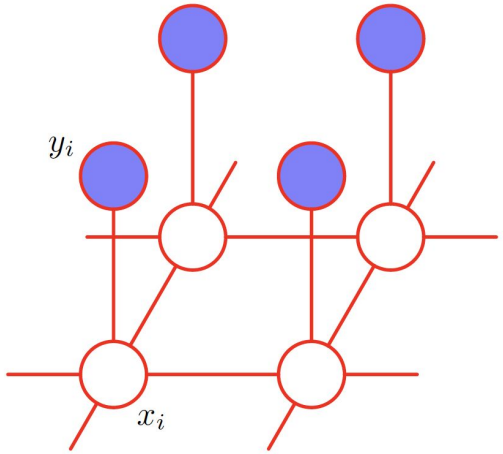
⇒ Neighbors tend to share values!

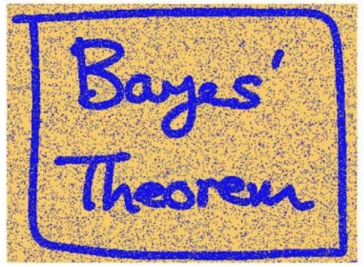
# **Some Famous Markov Random Fields** — Ising Model

Applications:

# **Some Famous Markov Random Fields** — Ising Model

Applications: image denoising.



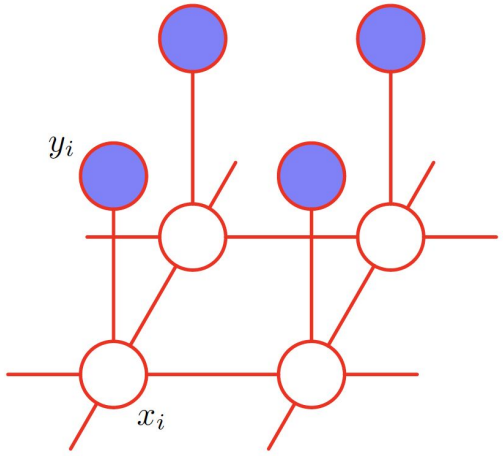



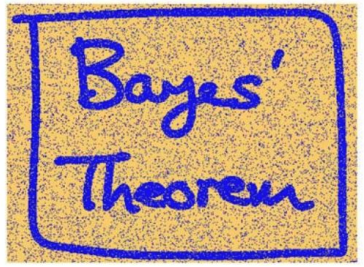


Source: Sargur Srihari, "Markov Random Fields"

# **Some Famous Markov Random Fields** — Ising Model

Applications: image denoising.





Source: Sargur Srihari, "Markov Random Fields"

Along with models of: phase separation, liquid-gas transitions, binary alloys, spin glasses, protein folding, opinion dynamics, social network analysis, etc.

# **Some Famous Markov Random Fields** — Conditional Random Field

# **Some Famous Markov Random Fields** — Conditional Random Field

Conditional random field — similar to an undirected version of a HMM.

- Predict latent variables based on observed variables (on sequences).

Original Paper (2001)

→ 19,154 citations

**Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data**


**John Lafferty**[†*] LAFFERTY@CS.CMU.EDU
**Andrew McCallum**[*†] MCCALLUM@WHIZBANG.COM
**Fernando Pereira**[*‡] FPEREIRA@WHIZBANG.COM

[*]WhizBang! Labs–Research, 4616 Henry Street, Pittsburgh, PA 15213 USA
[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[‡]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA



**Abstract**

We present *conditional random fields*, a framework for building probabilistic models to segment and label sequence data. Conditional random fields offer several advantages over hidden Markov models and stochastic grammars for such tasks, including the ability to relax strong independence assumptions made in those models. Conditional random fields also avoid a fundamental limitation of maximum entropy Markov models (MEMMs) and other discriminative Markov models based on directed graphical models, which can be biased towards states with few successor states. We present iterative parameter estimation algorithms for conditional random fields and compare the performance of the resulting models to HMMs and MEMMs on synthetic and natural-language data.


**1. Introduction**

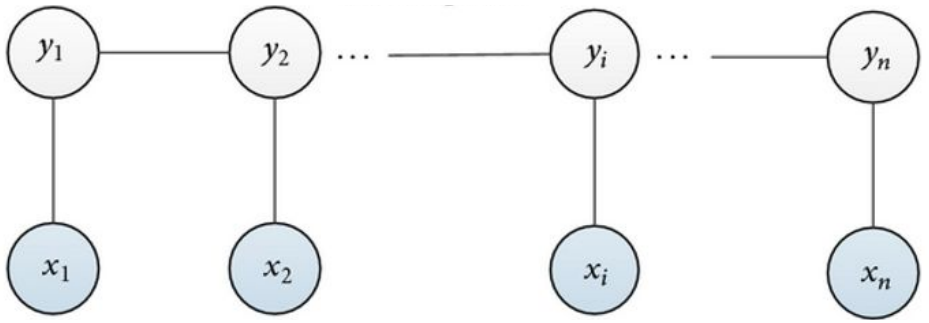
The need to segment and label sequences arises in many different problems in several scientific fields. Hidden Markov models (HMMs) and stochastic grammars are well understood and widely used probabilistic models for such problems. In computational biology, HMMs and stochastic grammars have been successfully used to align biological sequences, find sequences homologous to a known evolutionary family, and analyze RNA secondary structure (Durbin et al., 1998). In computational linguistics and computer science, HMMs and stochastic grammars have been applied to a wide variety of problems in text and speech processing, including topic segmentation, part-of-speech (POS) tagging, information extraction, and syntactic disambiguation (Manning & Schütze, 1999).

HMMs and stochastic grammars are generative models, assigning a joint probability to paired observation and label sequences; the parameters are typically trained to maximize the joint likelihood of training examples. To define a joint probability over observation and label sequences, a generative model needs to enumerate all possible observation sequences, typically requiring a representation in which observations are task-appropriate atomic entities, such as words or nucleotides. In particular, it is not practical to represent multiple interacting features or long-range dependencies of the observations, since the inference problem for such models is intractable.

This difficulty is one of the main motivations for looking at conditional models as an alternative. A conditional model specifies the probabilities of possible label sequences given an observation sequence. Therefore, it does not expend modeling effort on the observations, which at test time are fixed anyway. Furthermore, the conditional probability of the label sequence can depend on arbitrary, non-independent features of the observation sequence without forcing the model to account for the distribution of those dependencies. The chosen features may represent attributes at different levels of granularity of the same observations (for example, words and characters in English text), or aggregate properties of the observation sequence (for instance, text layout). The probability of a transition between labels may depend not only on the current observation, but also on past and future observations, if available. In contrast, generative models must make very strict independence assumptions on the observations, for instance conditional independence given the labels, to achieve tractability.

Maximum entropy Markov models (MEMMs) are conditional probabilistic sequence models that attain all of the above advantages (McCallum et al., 2000). In MEMMs, each source state[1] has a exponential model that takes the observation features as input, and outputs a distribution over possible next states. These exponential models are trained by an appropriate iterative scaling method in the

[1]Output labels are associated with states; it is possible for several states to have the same label, but for simplicity in the rest of this paper we assume a one-to-one correspondence.




Source: “Conditional Random Fields”, codersarts.com

# **Some Famous Markov Random Fields** — Conditional Random Field

Conditional random field — similar to an undirected version of a HMM.

- Predict latent variables based on observed variables (on sequences).

Original Paper (2001)

→ 19,154 citations

**Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data**


**John Lafferty**[†*] LAFFERTY@CS.CMU.EDU
**Andrew McCallum**[*†] MCCALLUM@WHIZBANG.COM
**Fernando Pereira**[*‡] FPEREIRA@WHIZBANG.COM

[*]WhizBang! Labs–Research, 4616 Henry Street, Pittsburgh, PA 15213 USA
[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[‡]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA



**Abstract**

We present *conditional random fields*, a framework for building probabilistic models to segment and label sequence data. Conditional random fields offer several advantages over hidden Markov models and stochastic grammars for such tasks, including the ability to relax strong independence assumptions made in those models. Conditional random fields also avoid a fundamental limitation of maximum entropy Markov models (MEMMs) and other discriminative Markov models based on directed graphical models, which can be biased towards states with few successor states. We present iterative parameter estimation algorithms for conditional random fields and compare the performance of the resulting models to HMMs and MEMMs on synthetic and natural-language data.


**1. Introduction**

The need to segment and label sequences arises in many different problems in several scientific fields. Hidden Markov models (HMMs) and stochastic grammars are well understood and widely used probabilistic models for such problems. In computational biology, HMMs and stochastic grammars have been successfully used to align biological sequences, find sequences homologous to a known evolutionary family, and analyze RNA secondary structure (Durbin et al., 1998). In computational linguistics and computer science, HMMs and stochastic grammars have been applied to a wide variety of problems in text and speech processing, including topic segmentation, part-of-speech (POS) tagging, information extraction, and syntactic disambiguation (Manning & Schütze, 1999).

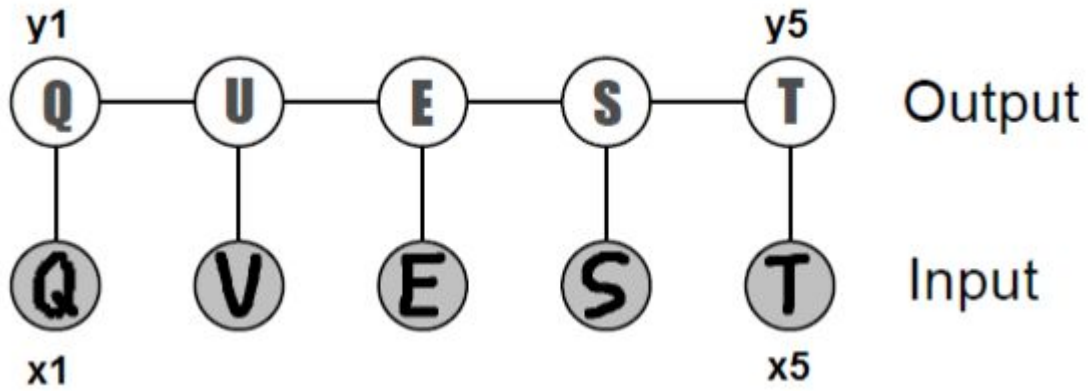
HMMs and stochastic grammars are generative models, assigning a joint probability to paired observation and label sequences; the parameters are typically trained to maximize the joint likelihood of training examples. To define a joint probability over observation and label sequences, a generative model needs to enumerate all possible observation sequences, typically requiring a representation in which observations are task-appropriate atomic entities, such as words or nucleotides. In particular, it is not practical to represent multiple interacting features or long-range dependencies of the observations, since the inference problem for such models is intractable.

This difficulty is one of the main motivations for looking at conditional models as an alternative. A conditional model specifies the probabilities of possible label sequences given an observation sequence. Therefore, it does not expend modeling effort on the observations, which at test time are fixed anyway. Furthermore, the conditional probability of the label sequence can depend on arbitrary, non-independent features of the observation sequence without forcing the model to account for the distribution of those dependencies. The chosen features may represent attributes at different levels of granularity of the same observations (for example, words and characters in English text), or aggregate properties of the observation sequence (for instance, text layout). The probability of a transition between labels may depend not only on the current observation, but also on past and future observations, if available. In contrast, generative models must make very strict independence assumptions on the observations, for instance conditional independence given the labels, to achieve tractability.

Maximum entropy Markov models (MEMMs) are conditional probabilistic sequence models that attain all of the above advantages (McCallum et al., 2000). In MEMMs, each source state[1] has a exponential model that takes the observation features as input, and outputs a distribution over possible next states. These exponential models are trained by an appropriate iterative scaling method in the

[1]Output labels are associated with states; it is possible for several states to have the same label, but for simplicity in the rest of this paper we assume a one-to-one correspondence.

*E.g.*, used to segment or label sequence data.




Source: Stefano Ermon, Deep Generative Models (CS236) Class

Applications in NLP, computer vision, bioinformatics.

# Conclusion

# Conclusion

**What are PGMS?** A rich visual language for defining conditional independence assumptions (and other structure) in joint probability distributions.

# Conclusion

**What are PGMS?** A rich visual language for defining conditional independence assumptions (and other structure) in joint probability distributions.

**Today:**

- Directed vs undirected graphical models: Bayesian networks and Markov random fields.
- Plate notation, generative process notation.
- Observed variables vs. latent variables vs. (hyper)parameters.
- Famous/classic Bayes Nets (HMMs, GMMs, LDA, VAE, etc.)
- Pros and Cons of Bayes Nets vs MRFs.
- Famous/classic MRFs (Ising Model, CRF).

# **Assignments and Grading** — Grading Breakdown

| Assignment | % of Grade |
|---|---|
| 1. Paper Presentation | 20% |
| 2. In-class Participation and Discussion | |
| 2a. Role 1 — Discussion Lead 1 | 8% |
| 2a. Role 2 — Discussion Lead 2 | 8% |
| 2a. Role 3 — Scribe | 9% |
| 3. Course Project | |
| 3a. Project Pitch | 8% |
| 3b. Midway Report | 10% |
| 3c. Final Presentation | 12% |
| 3d. Final Report | 25% |

# **Course Project** — Group Project

This will be a **group project** — groups of 3-4 students.

- Aiming for ~10 groups total (due to timing constraints)
- We will help facilitate this during class.
- *E.g.*, everyone will introduce themselves and describe research interests, which we will write/share, to help in matching.
- Need to aim to form teams and select project idea by roughly end of this month.

# **Course Project** — Guidance & Expectations

What does this project entail?

# **Course Project** — Guidance & Expectations

What does this project entail?

I want people to use a probabilistic or generative model in some way!

- Application of prob/gen models from this class on a novel task or dataset.
- Algorithmic improvements in learning, inference, or evaluation of prob/gen models.
- Theoretical analysis of any aspect of existing prob/gen models.

# **Course Project** — Guidance & Expectations

What does this project entail?

I want people to use a probabilistic or generative model in some way!

- Application of prob/gen models from this class on a novel task or dataset.
- Algorithmic improvements in learning, inference, or evaluation of prob/gen models.
- Theoretical analysis of any aspect of existing prob/gen models.

Goal is to complete a small-scale implementation or pilot study during the class.

I'm more focused on interesting conceptual ideas, rather than on performance/results.

Aim to connect it to the research you are focusing on outside of this class!

# **Course Project** — Assignments

The project will be worth a substantial portion of the grade, and consist of four main assignments:

# **Course Project** — Assignments

The project will be worth a substantial portion of the grade, and consist of four main assignments:

- **Project pitch:** Each group will come up with a project idea, make a few slides, and share their idea with the class for feedback (**10 minutes long**, on **Feb 7 & Feb 14**)

# **Course Project** — Assignments

The project will be worth a substantial portion of the grade, and consist of four main assignments:

- **Project pitch:** Each group will come up with a project idea, make a few slides, and share their idea with the class for feedback (**10 minutes long**, on **Feb 7 & Feb 14**)

- **Midway report:** Each group will write a short (4 page) midway report for their project, focusing on a literature review, implementation plan, and any initial experiments.
  - Latex template will be provided.

# **Course Project** — Assignments

The project will be worth a substantial portion of the grade, and consist of four main assignments:

- **Project pitch:** Each group will come up with a project idea, make a few slides, and share their idea with the class for feedback (**10 minutes long**, on **Feb 7 & Feb 14**)
- **Midway report:** Each group will write a short (4 page) midway report for their project, focusing on a literature review, implementation plan, and any initial experiments.
  - Latex template will be provided.
- **Final presentation:** Each group will  give a presentation to the class on their final project (**30 minutes long**, on **Apr 25 & May 2**)

# **Course Project** — Assignments

The project will be worth a substantial portion of the grade, and consist of four main assignments:

- **Project pitch:** Each group will come up with a project idea, make a few slides, and share their idea with the class for feedback (**10 minutes long**, on **Feb 7 & Feb 14**)
- **Midway report:** Each group will write a short (4 page) midway report for their project, focusing on a literature review, implementation plan, and any initial experiments.
  - Latex template will be provided.
- **Final presentation:** Each group will  give a presentation to the class on their final project (**30 minutes long**, on **Apr 25 & May 2**)
- **Final report:** Each group will submit a final report for their project, describing all details, background, prior work, and results (**8-10 pages long**, **due May 9**).
  - Latex template will be provided.

# Student Introductions

Link to spreadsheet:

https://docs.google.com/spreadsheets/d/1h5VtS7vFTn8mgsYYBQNDIpBxFI6OUKbrUKNrtXF6iDg/edit?usp=sharing