CSCI 699 - ProbGen

Probabilistic and Generative Models

Willie Neiswanger

Lecture 3 - Classic PGM Algorithms

Plan for Today

Plan for Today

Lecture: Classic algorithms in probabilistic graphical models (PGMs) for exact and approximate inference & learning.

Plan for Today

Lecture: Classic algorithms in probabilistic graphical models (PGMs) for exact and approximate inference & learning.

- Exact inference algorithms in PGMs.
- Variable Elimination algorithm.
- Belief Propagation algorithm (sum/max-product message passing).
- Famous algorithms: Forward-Backward & Viterbi.
- Expectation-Maximization (EM) algorithm.

Source: USC Viterbi Magazine, “The Viterbi Algorithm at 50”

Plan for Today

Lecture: Classic algorithms in probabilistic graphical models (PGMs) for exact and approximate inference & learning.

- Exact inference algorithms in PGMs.
- Variable Elimination algorithm.
- Belief Propagation algorithm (sum/max-product message passing).
- Famous algorithms: Forward-Backward & Viterbi.
- Expectation-Maximization (EM) algorithm.

Source: USC Viterbi Magazine, “The Viterbi Algorithm at 50”

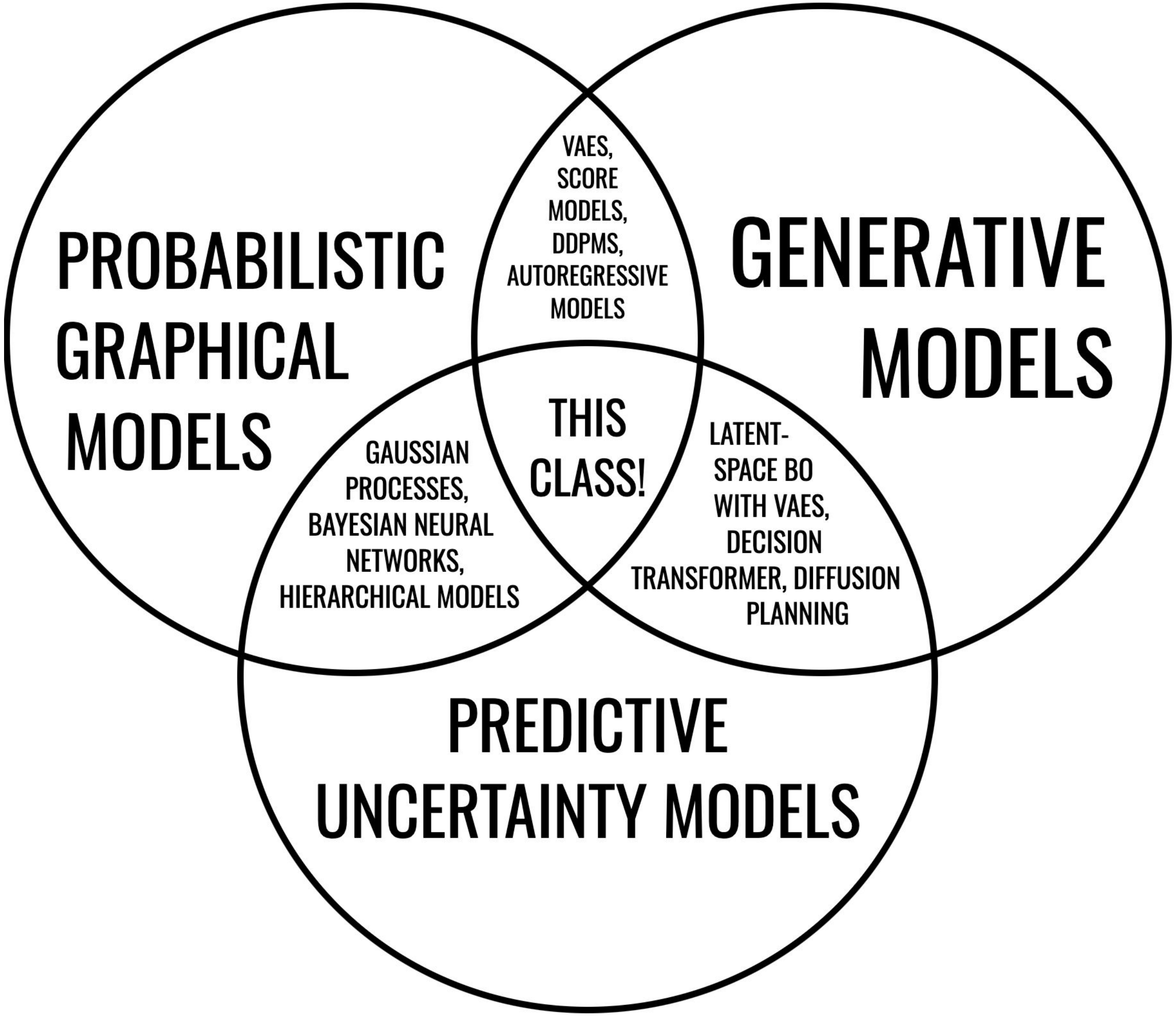
After:

- Check-in on group formation and project ideation.
- Tutorial on CARC access and cluster/slurm usage.

Putting this In Context

Putting this In Context

This course focuses on probabilistic models and their central role within modern machine learning and generative modeling.



Last Class

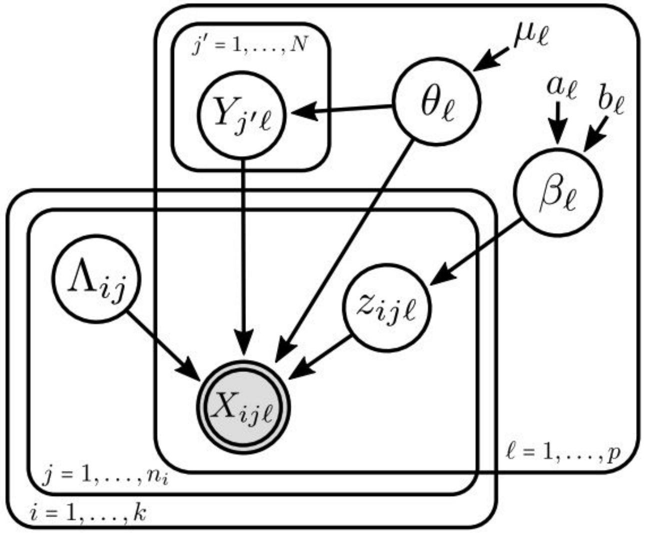
Our last class was on probabilistic graphical models (PGMs).

Last Class

Our last class was on probabilistic graphical models (PGMs).

A useful tool for describing **probabilistic models**, *e.g.*, $p_\theta(x_1, \ldots, x_n)$.

⇒ A joint probability distribution over multiple variables, which models some real world events or observations.



This Class (and future classes)

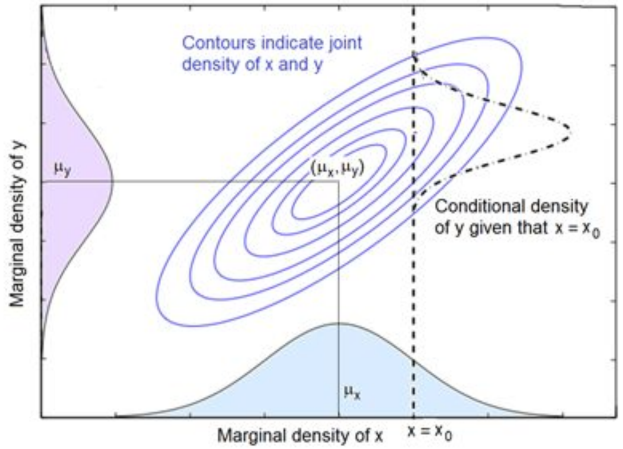
This class — and many of our upcoming classes — are all about **inference and learning in probabilistic models**.

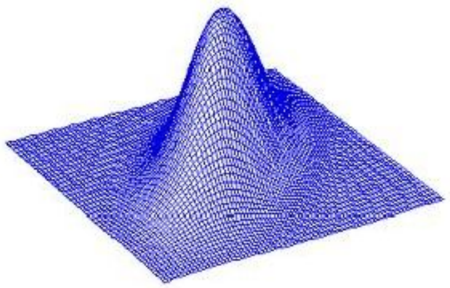
This Class (and future classes)

This class — and many of our upcoming classes — are all about **inference and learning in probabilistic models**.

I.e., computing:

- Marginal distributions
- Conditional distributions
- Their maximizers (*e.g.*, MAP/*maximum a posteriori* inferences)
- Other point estimates of parameters (MLE/*maximum likelihood* estimates)





Source: Pietro Vischia - Statistics for HEP

Inference and Learning in Probabilistic Models

Why is this important?

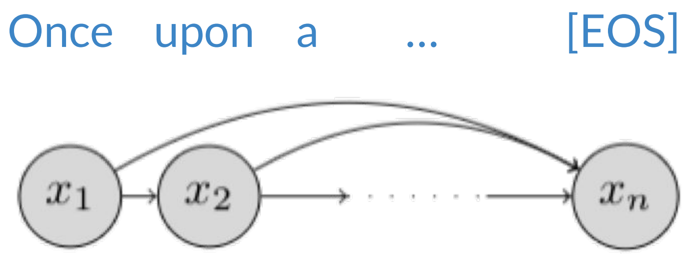
Inference and Learning in Probabilistic Models

Why is this important?

Many of the core procedures for deep generative models (and probabilistic models in general) are "just" inference and learning in probabilistic model.

And then using the inferred distributions downstream (*e.g.* in sampling procedures.)

Source: An Introduction to Flow Matching. By Fjelde, Mathieu, and Dutordoir.



Inference and Learning in Probabilistic Models

A few examples of this:

Inference and Learning in Probabilistic Models

A few examples of this:

Posterior Inference **(⇒ inferring a conditional dist.)**

Inference and Learning in Probabilistic Models

A few examples of this:

Posterior Inference **(⇒ inferring a conditional dist.)**

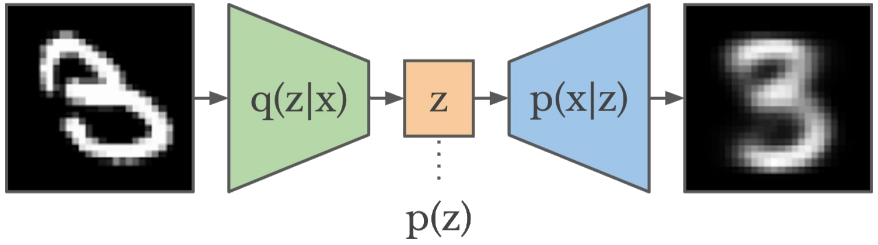
- *E.g., Variational Inference Algorithms*
- ⇒ approximate posterior inference (conditional distribution of latent variables given observed variables) via an optimization procedure.

Inference and Learning in Probabilistic Models

A few examples of this:

Posterior Inference **(⇒ inferring a conditional dist.)**

- *E.g., Variational Inference Algorithms*
- ⇒ approximate posterior inference (conditional distribution of latent variables given observed variables) via an optimization procedure.



- VAE (Variational Autoencoder)
- DDPM (Denoising Diffusion Probabilistic Model)
- DDIM (Denoising Diffusion Implicit Model)
- Variational Inference in (Deep) Gaussian Processes

Inference and Learning in Probabilistic Models

A few examples of this:

Posterior Inference (⇒ inferring a conditional dist.)

- *E.g., Markov chain Monte Carlo (MCMC) Algorithms*

Inference and Learning in Probabilistic Models

A few examples of this:

Posterior Inference **(⇒ inferring a conditional dist.)**

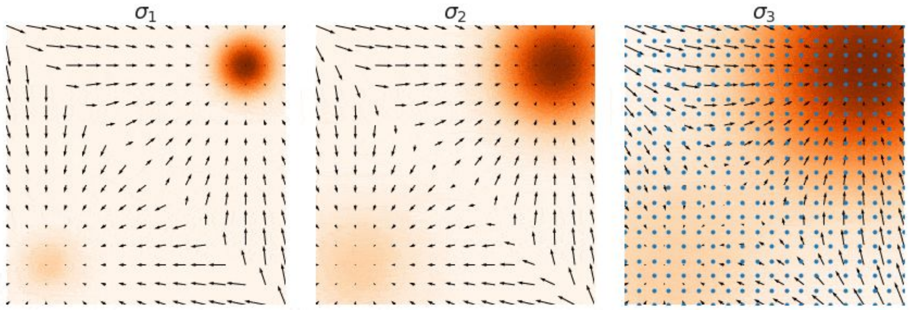
- *E.g., Markov chain Monte Carlo (MCMC) Algorithms*
- ⇒ approximate posterior inference (conditional distribution of latent variables given observed variables)... *via sampling from conditional dist*.

Inference and Learning in Probabilistic Models

A few examples of this:

Posterior Inference (⇒ inferring a conditional dist.)

- *E.g., Markov chain Monte Carlo (MCMC) Algorithms*
- ⇒ approximate posterior inference (conditional distribution of latent variables given observed variables)... *via sampling from conditional dist.*



- Score-based generative models.
- Bayesian neural networks.
- Energy-based deep generative models.
- Boltzmann Machines (RBMs, DBMs, and DBNs)

Inference and Learning in Probabilistic Models

A few examples of this:

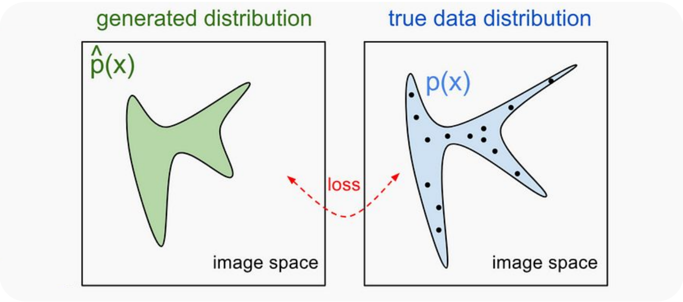
Learning in Probabilistic Models

Inference and Learning in Probabilistic Models

A few examples of this:

Learning in Probabilistic Models

- *E.g., MAP or MLE point estimates*
- ⇒ estimating parameters of a probabilistic model via maximizing likelihood or posterior (*e.g.*, parameter could be neural network weights).



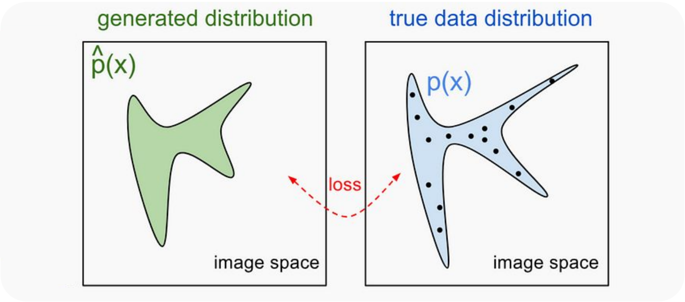
Source: OpenAI

Inference and Learning in Probabilistic Models

A few examples of this:

Learning in Probabilistic Models

- *E.g., MAP or MLE point estimates*
- ⇒ estimating parameters of a probabilistic model via maximizing likelihood or posterior (*e.g.*, parameter could be neural network weights).



Source: OpenAI

- Autoregressive models (PixelCNN, PixelRNN, Transformers).
- Flow-based generative models (normalizing flow, continuous normalizing flow).
- Hidden Markov Models (HMMs), *e.g.*, via Viterbi algorithm.

Inference and Learning in Probabilistic Models

A few examples of this:

Sampling from Probabilistic Models

Inference and Learning in Probabilistic Models

A few examples of this:

Sampling from Probabilistic Models

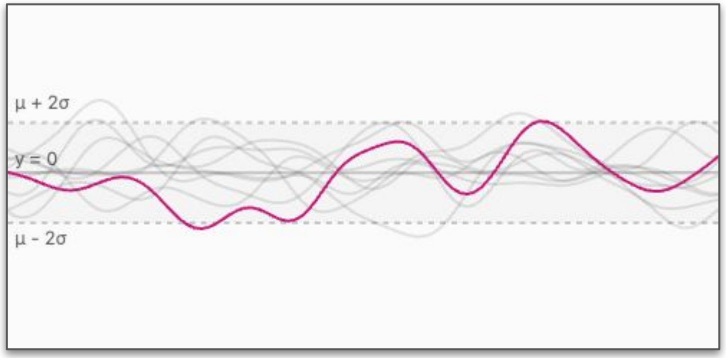
- *E.g., generation from learned model or posterior*
- ⇒ generating samples from a probabilistic model after training or inference — *e.g.*, from a learned model or posterior distribution.

Inference and Learning in Probabilistic Models

A few examples of this:

Sampling from Probabilistic Models

- *E.g., generation from learned model or posterior*
- ⇒ generating samples from a probabilistic model after training or inference — *e.g.*, from a learned model or posterior distribution.



- VAE, GAN
- Autoregressive Models
- Flow-based Models
- Diffusion Models
- BayesOpt methods (Entropy search, Thompson sampling)

Today's Lecture

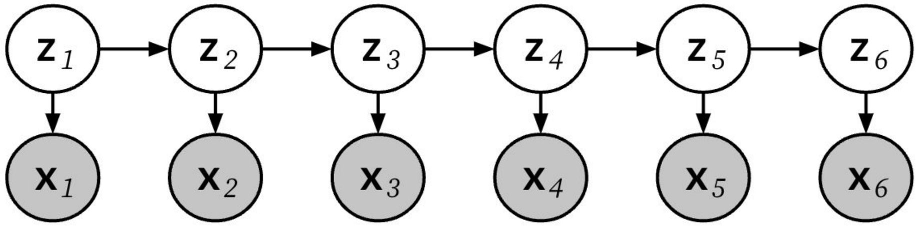
Exact algorithms for **inference and learning**, in probabilistic graphical models.

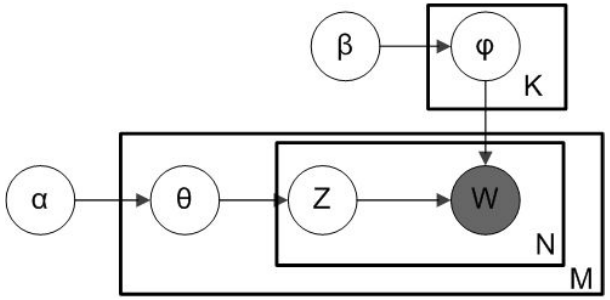
Today's Lecture

Exact algorithms for **inference and learning**, in probabilistic graphical models.

E.g., PGMs with discrete random variables.

In these models, we can do **exact inference.**

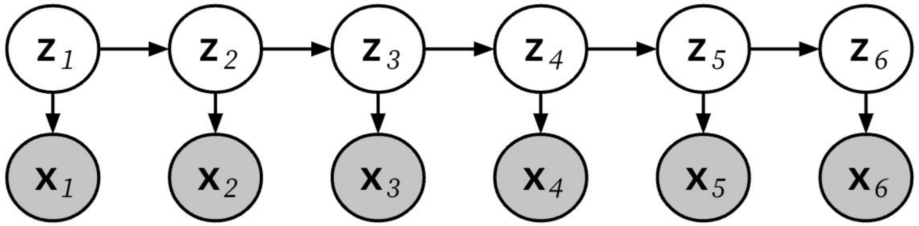




Today's Lecture

Exact algorithms for **inference and learning**, in probabilistic graphical models.

E.g., PGMs with discrete random variables.

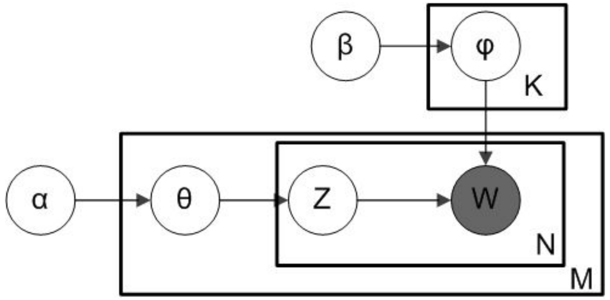


In these models, we can do **exact inference.**

⇒ We have algorithms that compute marginal/conditional distributions of a probability model exactly, without any approximation.

⇒ Sometimes computing in an efficient manner (polynomial time), depending on the structure of the PGM.

⇒ Will lead to some famous algorithms (Forward-Backward, Viterbi, Baum-Welch).



Following Lectures

In subsequent lectures we will move onto **inference and learning** in more complex probabilistic models (including deep generative models).

Following Lectures

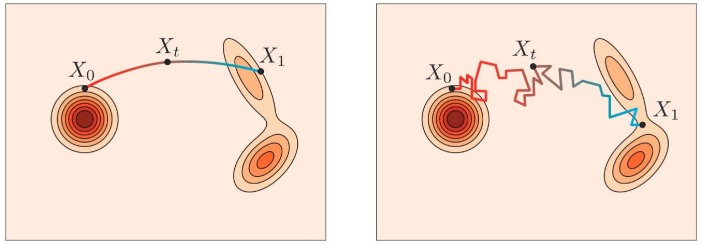
In subsequent lectures we will move onto **inference and learning** in more complex probabilistic models (including deep generative models).

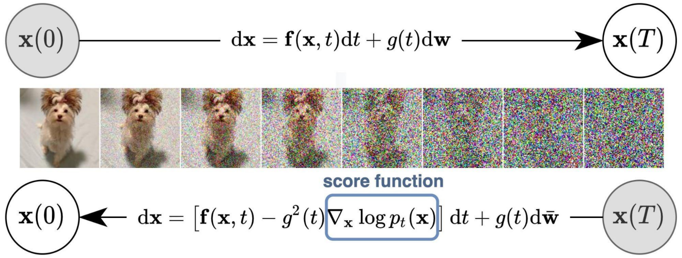
In these models, we will often have to resort to **approximate inference.**

⇒ Approximations to marginal/conditional distributions.

⇒ Sometimes involving different representations of distributions, such as sampling-based, using a simplified parametric form, or defined implicitly via a neural network.

⇒ Which may only hold, *e.g.*, in the limit as time $\to$ **∞**, or if you can solve a non-convex optimization problem.





Review: Probabilistic Graphical Models (PGMs)

Review — Last Class on 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)

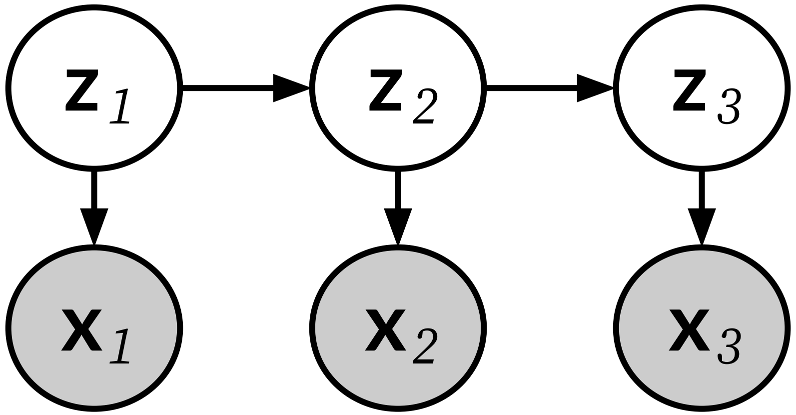
E.g., Conditional Random Field (CRF)

Review — Last Class on 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)

Review — Last Class on Probabilistic Graphical Models (PGMs)

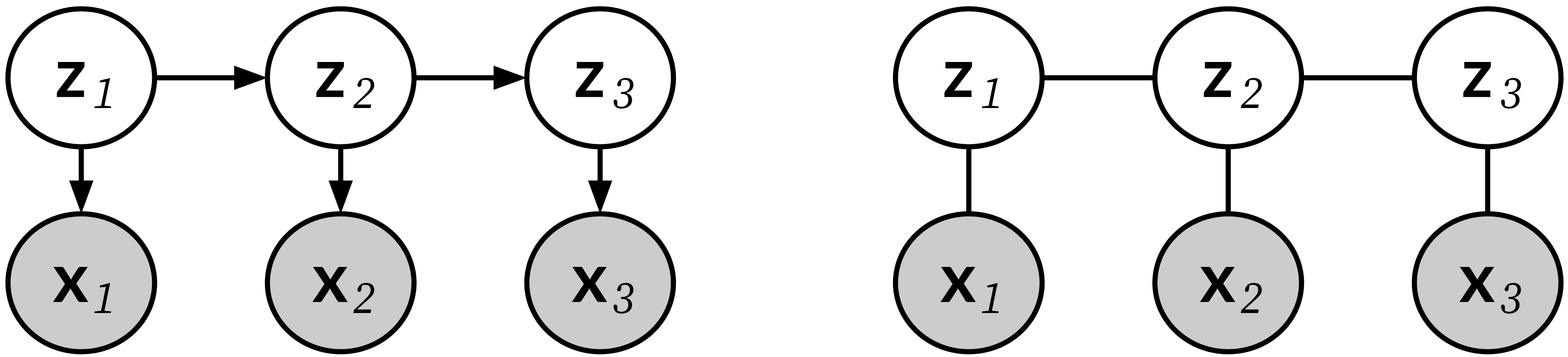
At a high level: two main "types" of PGMs.

Bayesian Networks (directed)

E.g., Hidden Markov Model (HMM)

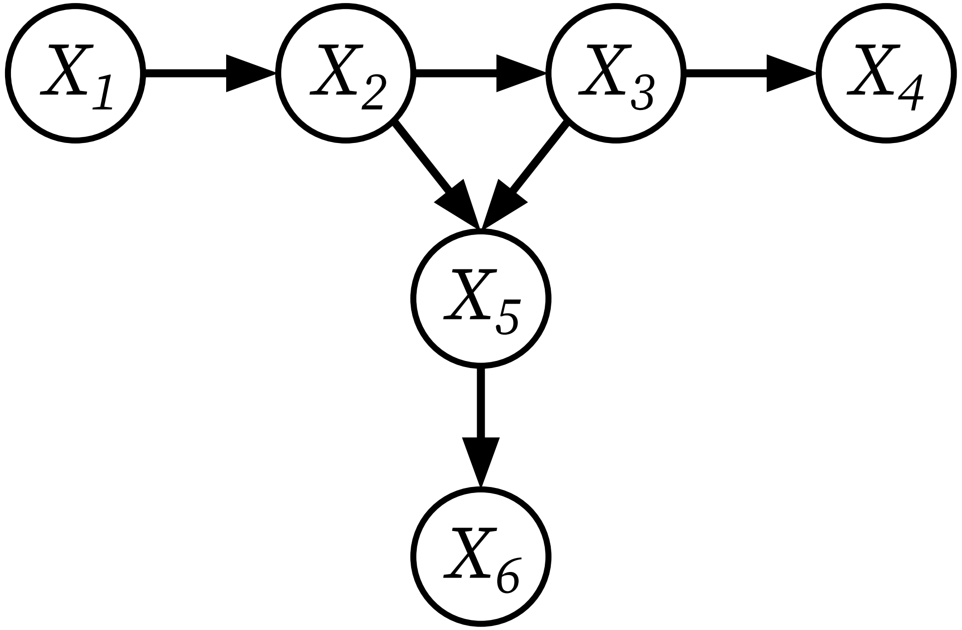
Markov Random Fields (undirected)

E.g., Conditional Random Field (CRF)



Review — Last Class on Probabilistic Graphical Models (PGMs)

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

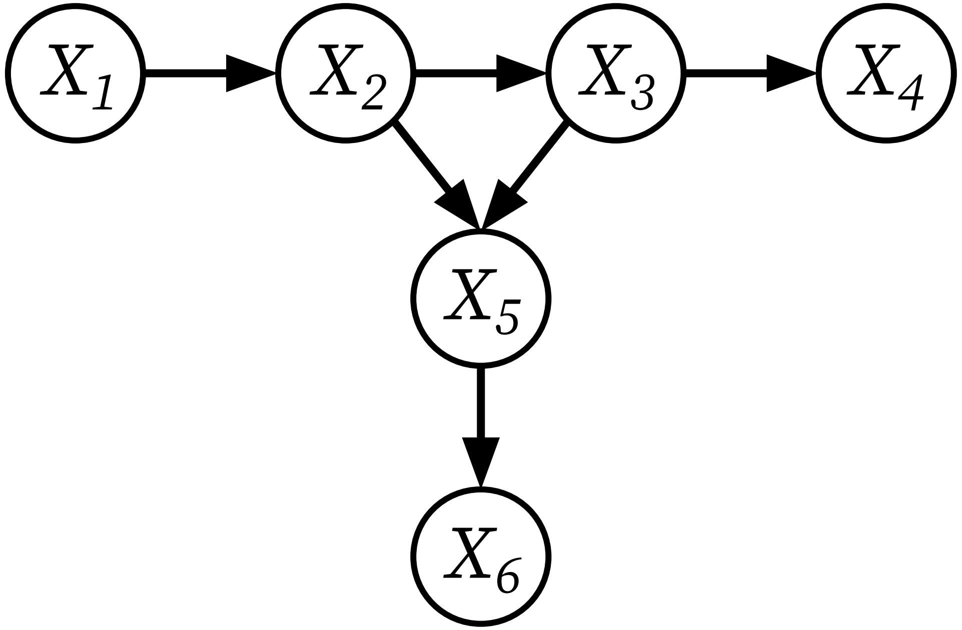


Review — Last Class on Probabilistic Graphical Models (PGMs)

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.



Review — Last Class on Probabilistic Graphical Models (PGMs)

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

Review — Last Class on Probabilistic Graphical Models (PGMs)

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 ϕ_c is a non-negative function over the variables in a clique.

Review — Last Class on Probabilistic Graphical Models (PGMs)

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

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

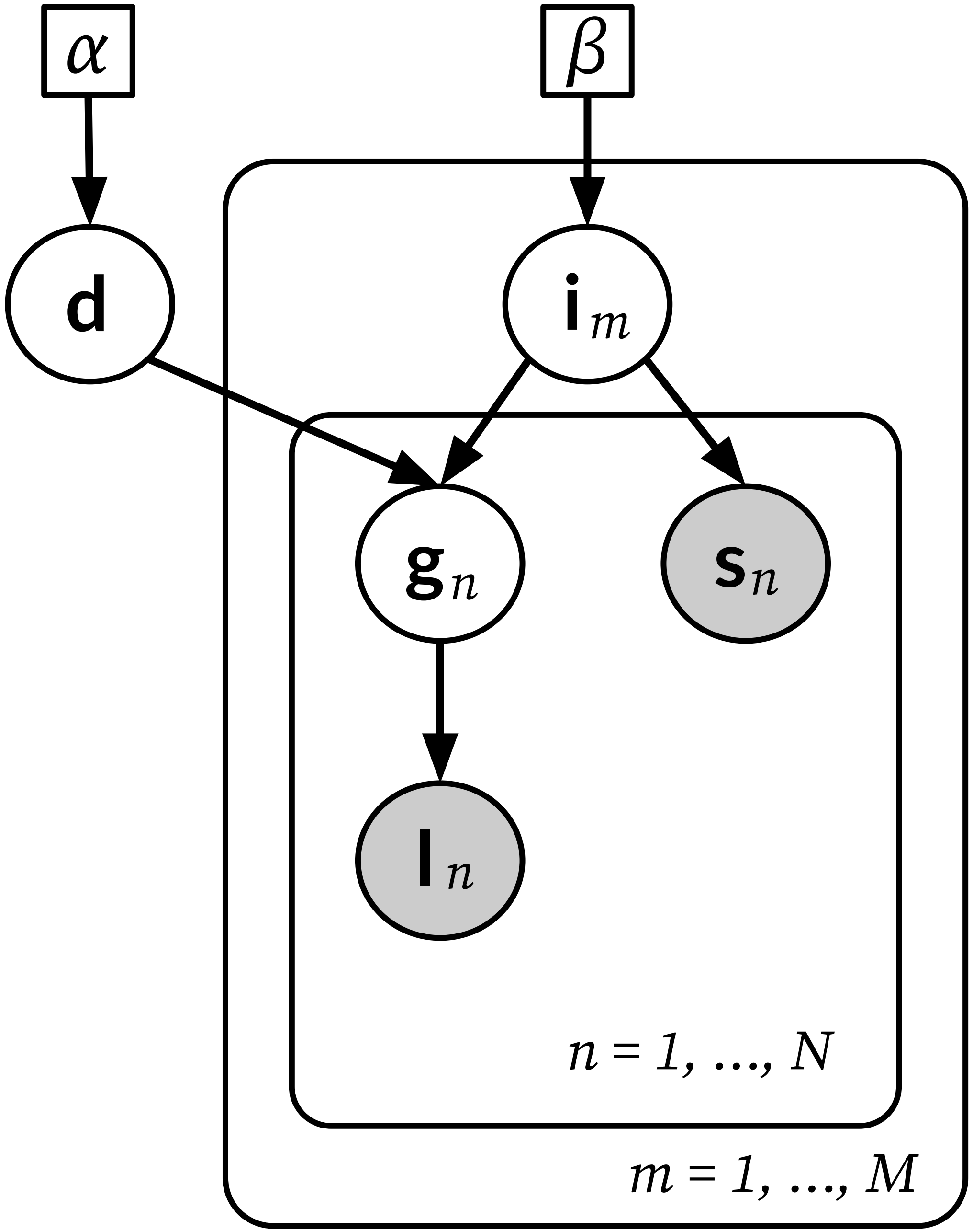
where *C* denotes the set of cliques (*i.e.*, fully connected subgraphs) of *G*, and each factor ϕ_c is a non-negative function over the variables in a clique.

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

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

Review — Last Class on PGMs

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

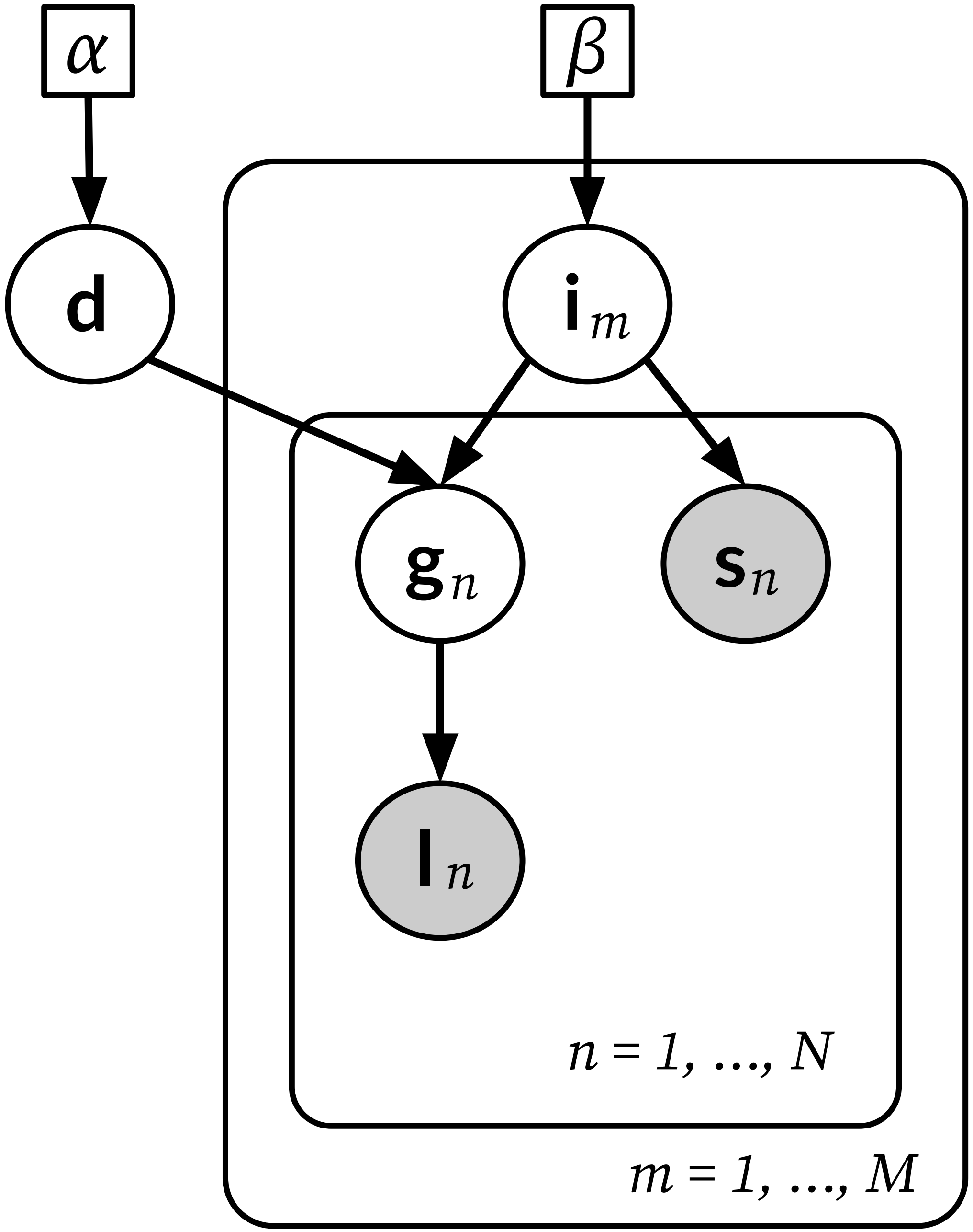


Review — Last Class on PGMs

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

We covered:

- Observed vs. latent variables.
- Plates (with indices).
- Multiple / nested plates.
- Constants (not random variables).



Review — Last Class on PGMs

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

Also, covered generative process notation:

$d \sim p(d)$

For m *= 1, …, M:*

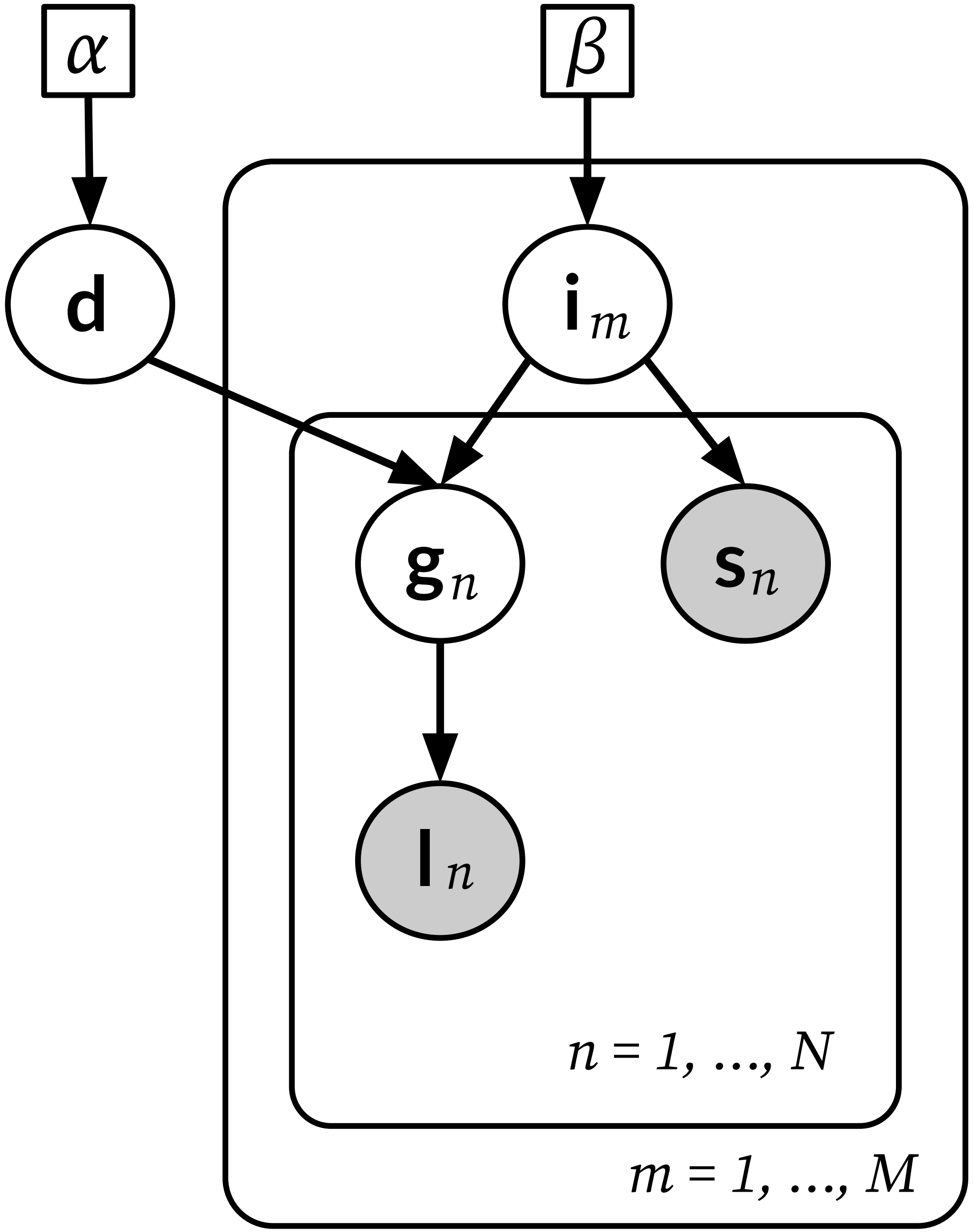
$\quad i \sim p(i)$

$\quad$ *For* n *= 1, …, N:*

$\quad\quad g \sim p(g \mid i, d)$

$\quad\quad s \sim p(s \mid i)$

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



Review — Last Class on PGMs

Famous PGMs

Review — Last Class on PGMs

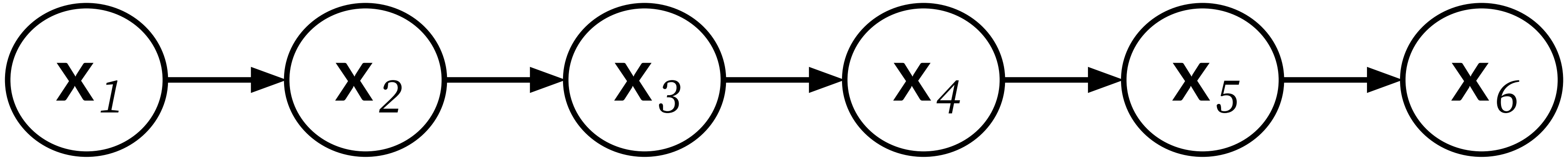
Famous PGMs

Bayesian Networks:

Review — Last Class on PGMs

Famous PGMs

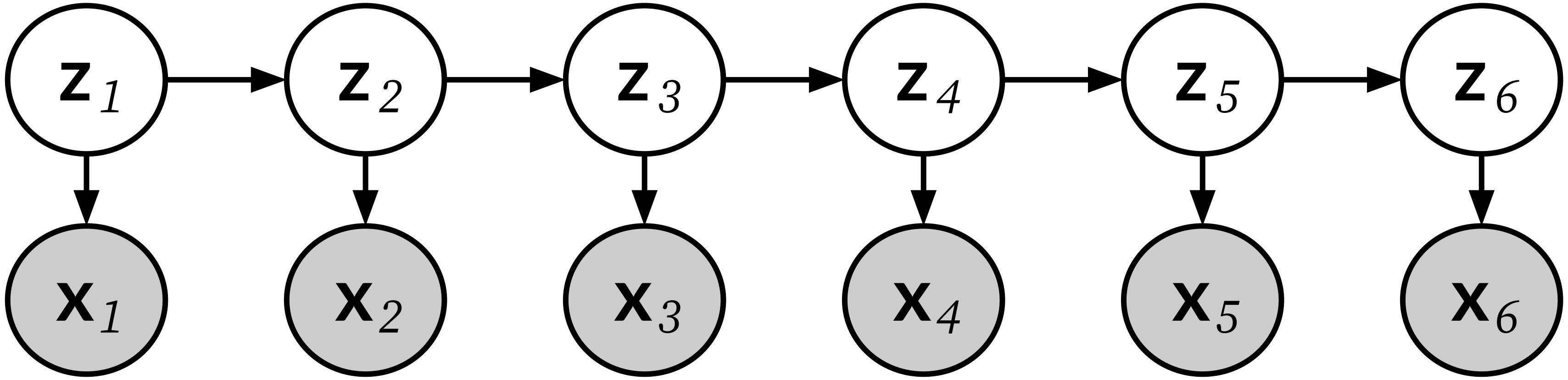
Bayesian Networks: **Markov models / Markov chains**



Review — Last Class on PGMs

Famous PGMs

Bayesian Networks: **Hidden Markov Models (HMMs)**

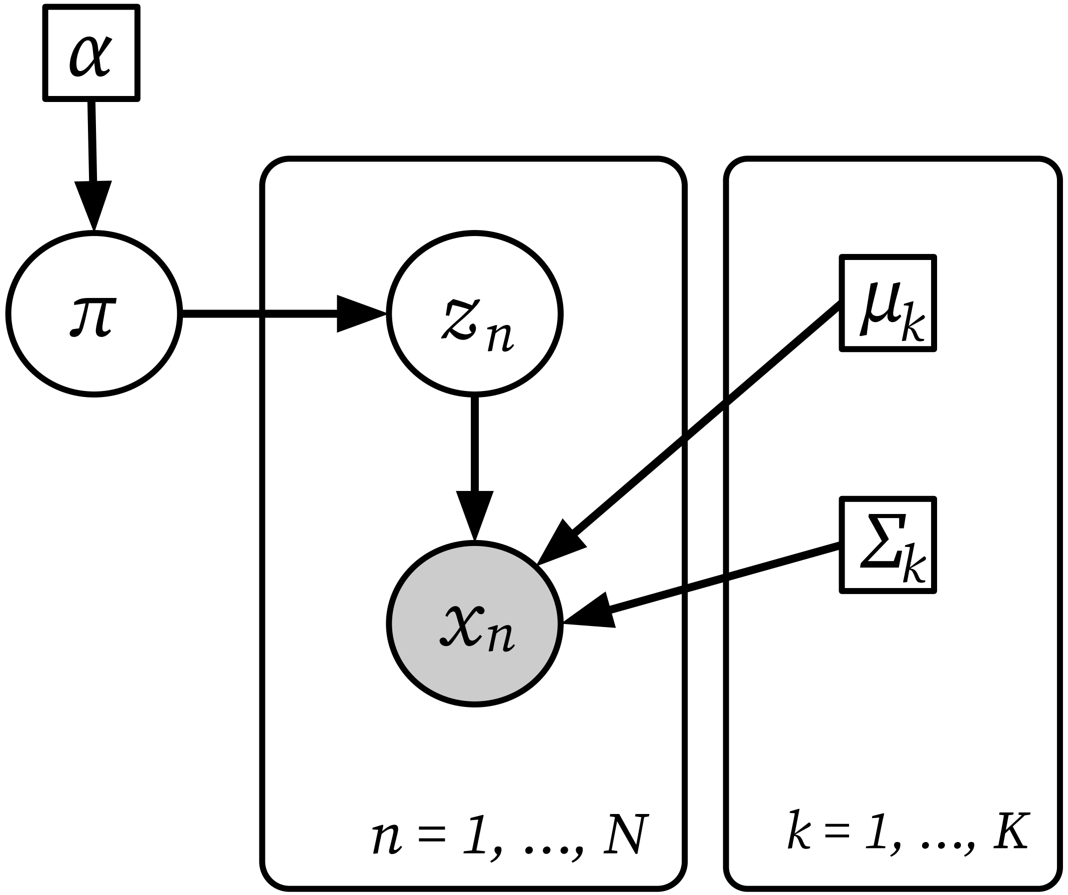


Review — Last Class on PGMs

Famous PGMs

Bayesian Networks:

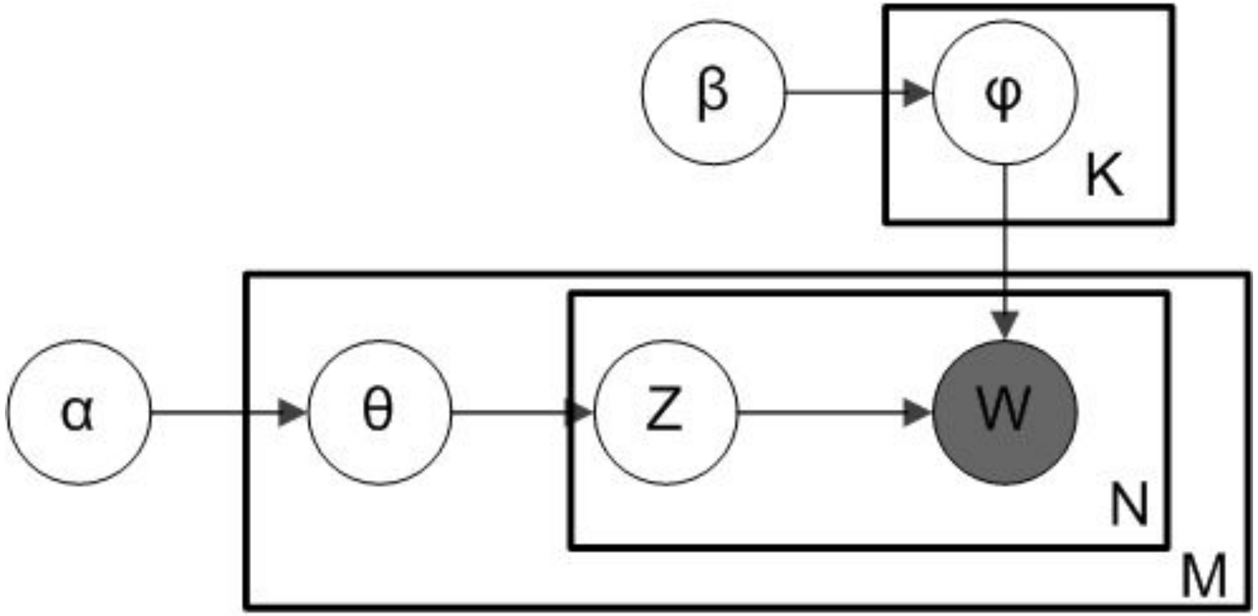
Gaussian Mixture Models (GMMs)



Review — Last Class on PGMs

Famous PGMs

Bayesian Networks: **Latent Dirichlet Allocation (LDA)**

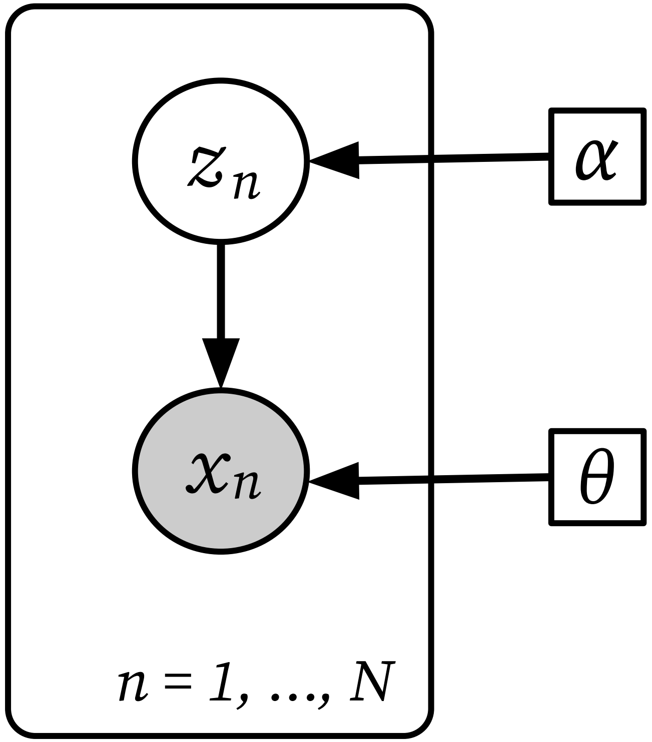


Review — Last Class on PGMs

Famous PGMs

Bayesian Networks:

Variational Autoencoders (VAEs)



Review — Last Class on PGMs

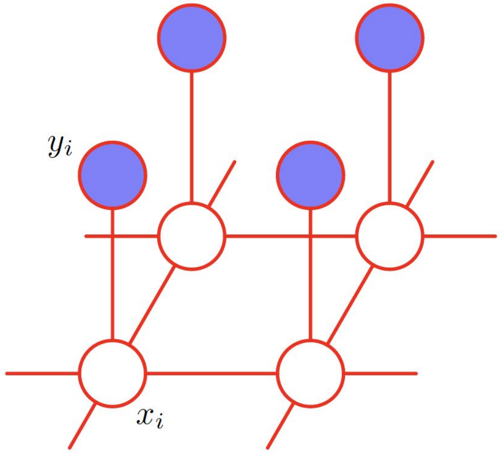
Famous PGMs

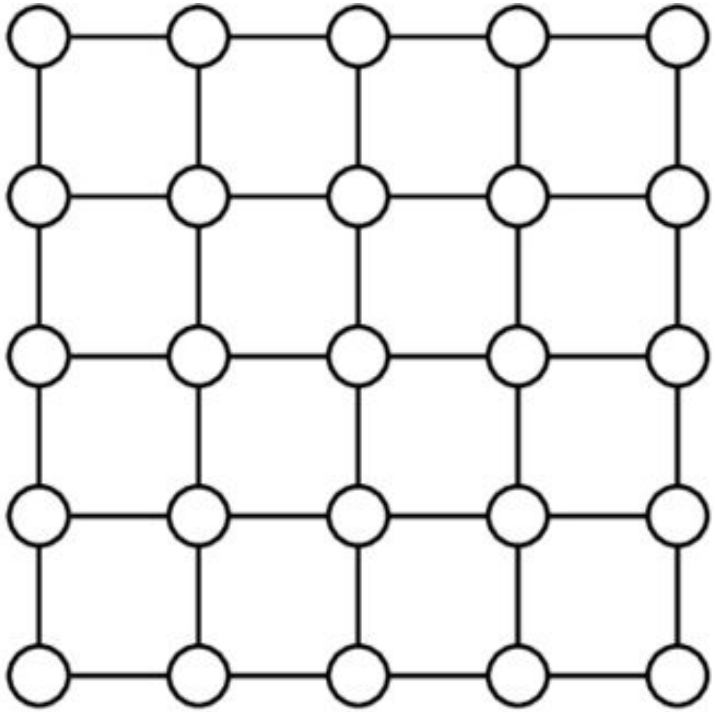
Markov Random Fields:

Review — Last Class on PGMs

Famous PGMs

Markov Random Fields:
Ising Models

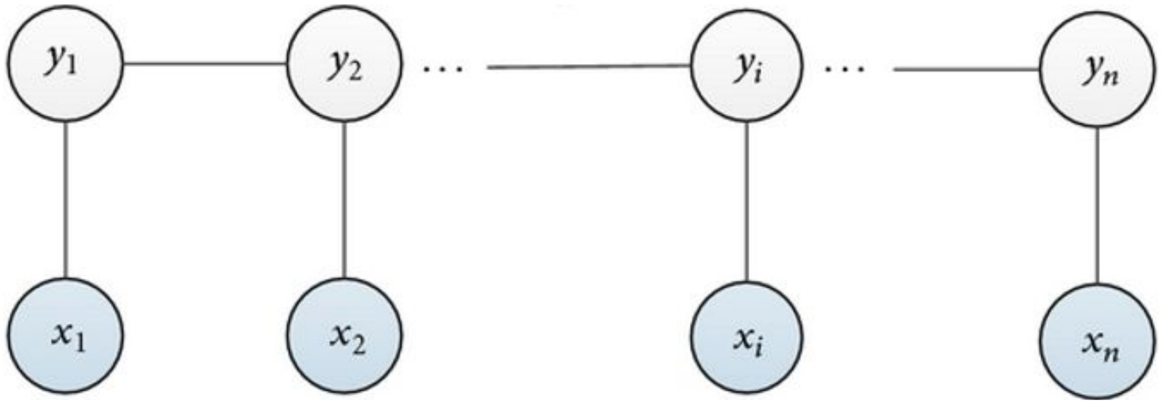




Review — Last Class on PGMs

Famous PGMs

Markov Random Fields: **Conditional Random Fields (CRFs)**

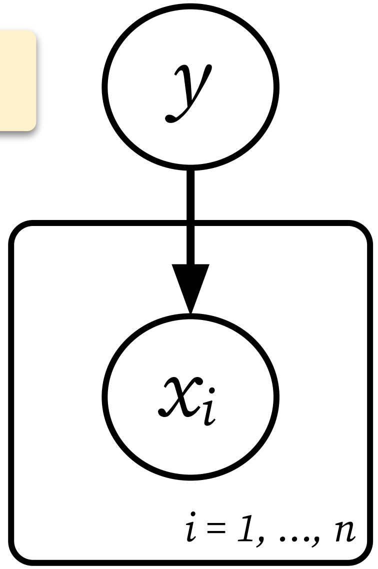


Source: "Conditional Random Fields", codersarts.com

Exact Inference and Learning in PGMs

Marginal, Conditional, and MAP Inference

Example PGM



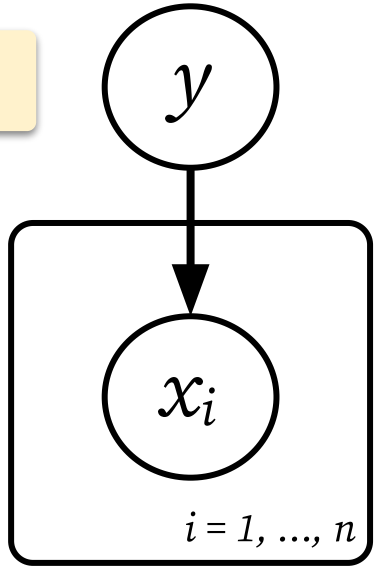
Consider the following three types of inference queries.

- *Marginal Inference*: what is the probability of a given variable in our model after we sum everything else out?

$$p(y) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_n} p(y, x_1, \ldots, x_n)$$

Marginal, Conditional, and MAP Inference

Example PGM



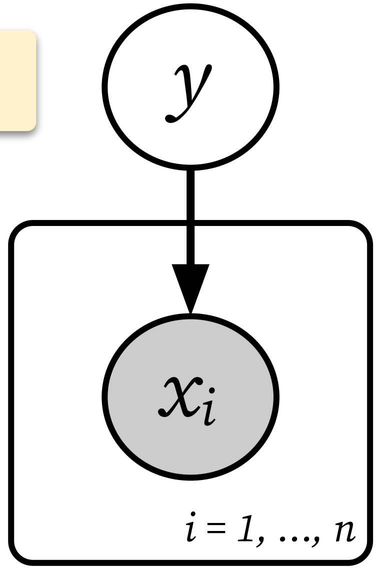
Consider the following three types of inference queries.

- *Marginal Inference*: what is the probability of a given variable in our model after we sum everything else out?

$$p(y=1) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_n} p(y=1, x_1, \ldots, x_n)$$

Marginal, Conditional, and MAP Inference

Example PGM



Consider the following three types of inference queries.

- *Marginal Inference*: what is the probability of a given variable in our model after we sum everything else out?

$$p(y) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_n} p(y, x_1, \ldots, x_n)$$

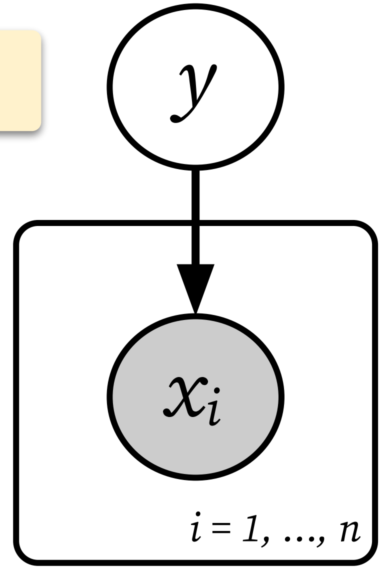
- *Conditional Inference*: what is the probability of a given variable in our model, conditioned on some observations?

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

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Marginal, Conditional, and MAP Inference

Example PGM



Consider the following three types of inference queries.

- *Marginal Inference*: what is the probability of a given variable in our model after we sum everything else out?

$$p(y) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_n} p(y, x_1, \ldots, x_n)$$

- *Conditional Inference*: what is the probability of a given variable in our model, conditioned on some observations?

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

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Quick Aside — Bayes Rule and Bayesian Networks

Bayes Rule in the context of Bayesian Networks:

Bayes Rule and Bayesian Networks

Bayes Rule in the context of Bayesian Networks:

Recall *Bayes Rules* (from first lecture):

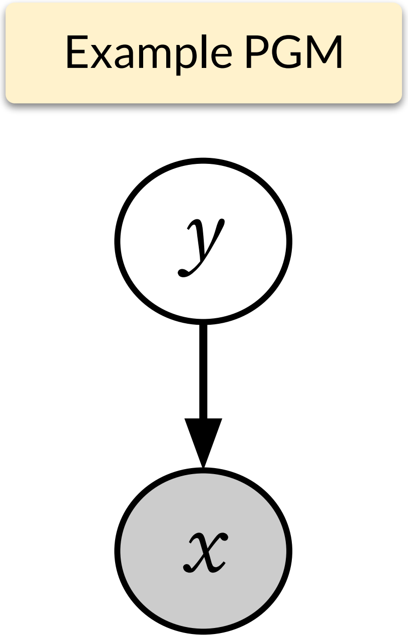
Definition. Bayes Rule is defined to be

$$f_{Y|X}(y \mid x) = \frac{f_{XY}(x, y)}{f_X(x)} = \frac{f_{X|Y}(x \mid y) f_Y(y)}{\int_{-\infty}^{\infty} f_{X|Y}(x \mid y') f_Y(y') dy'}$$

Bayes Rule and Bayesian Networks

Bayes Rule in the context of Bayesian Networks:

- Consider a simple Bayesian network PGM here on the right.
- Consisting of latent variables y and observed variables x.



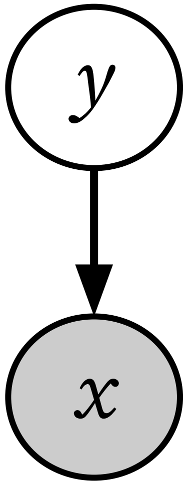
Bayes Rule and Bayesian Networks

Bayes Rule in the context of Bayesian Networks:

- Consider a simple Bayesian network PGM here on the right.
- Consisting of latent variables y and observed variables x.
- **Recall**: we often view/define a Bayesian network as a "generative process" that produces observations.

$$y \sim p(y)$$
$$x \sim p(x \mid y)$$

Example PGM

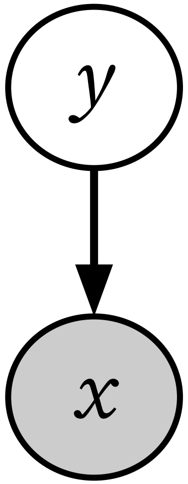


Bayes Rule and Bayesian Networks

So suppose we want to infer a conditional distribution of this probabilistic model.

Probability of latent variables given observed variables:

Example PGM



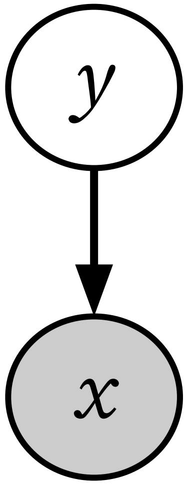
Bayes Rule and Bayesian Networks

So suppose we want to infer a conditional distribution of this probabilistic model.

Probability of latent variables given observed variables:

$$p(y \mid x)$$

Example PGM



Bayes Rule and Bayesian Networks

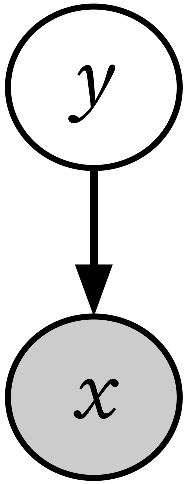
So suppose we want to infer a conditional distribution of this probabilistic model.

Probability of latent variables given observed variables:

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

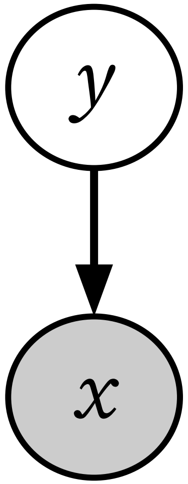
Definition of conditional dist.

Example PGM



Bayes Rule and Bayesian Networks

Example PGM



So suppose we want to infer a conditional distribution of this probabilistic model.

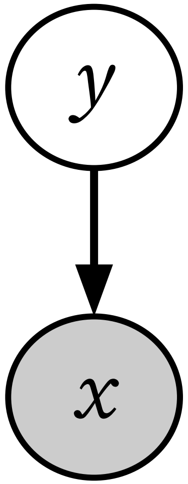
Probability of latent variables given observed variables:

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

Factorization based on PGM

Bayes Rule and Bayesian Networks

Example PGM



So suppose we want to infer a conditional distribution of this probabilistic model.

Probability of latent variables given observed variables:

Prior

Likelihood

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

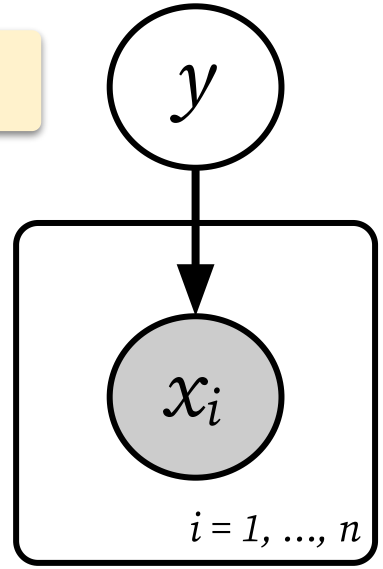
Marginal / Evidence

Factorization based on PGM

This conditional distribution naturally decomposes into *prior*, *likelihood*, *evidence*.

Marginal, Conditional, and MAP Inference

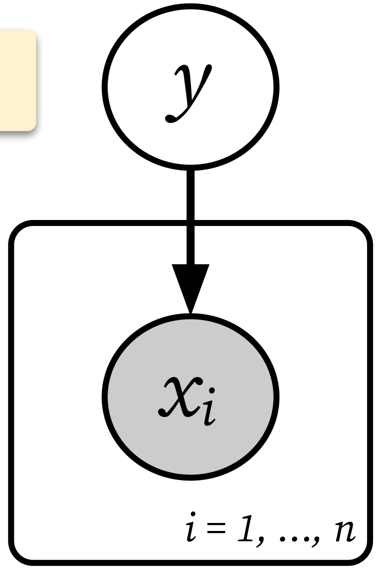
Example PGM



Consider the following three types of inference queries.

Marginal, Conditional, and MAP Inference

Example PGM



Consider the following three types of inference queries.

- *Marginal Inference*: what is the probability of a given variable in our model after we sum everything else out?

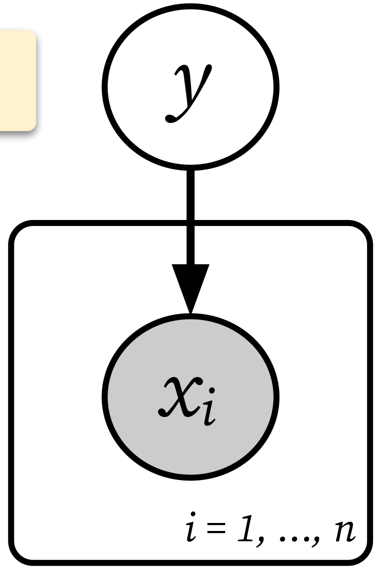
$$p(y) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_n} p(y, x_1, \ldots, x_n)$$

- *Conditional Inference*: what is the probability of a given variable in our model, conditioned on some observations?

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

Marginal, Conditional, and MAP Inference

Example PGM



Consider the following three types of inference queries.

- *Maximum a Posteriori (MAP) Inference*: what is the most likely assignment of the variables in the model (possibly when conditioned on observations)?

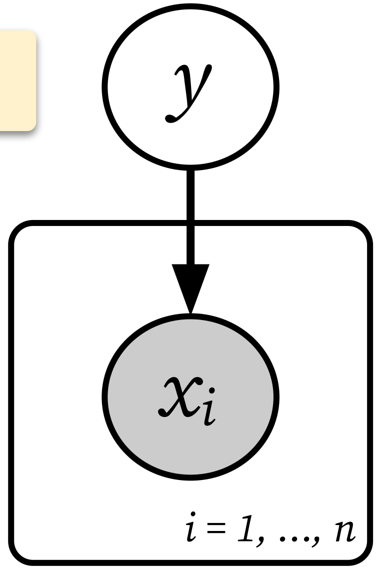
$$\max_{y} p(y \mid x_1, \ldots, x_n)$$

or

$$\max_{x_1, \ldots, x_n} p(y, x_1, \ldots, x_n)$$

Marginal, Conditional, and MAP Inference

Example PGM



Consider the following three types of inference queries.

- *Maximum a Posteriori (MAP) Inference*: what is the most likely assignment of the variables in the model (possibly when conditioned on observations)?

$$\arg\max_{y} p(y \mid x_1, \ldots, x_n)$$

or

$$\arg\max_{x_1, \ldots, x_n} p(y, x_1, \ldots, x_n)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Marginal and MAP Inference

Inference is challenging!

Marginal and MAP Inference

Inference is challenging!

- In many PGMs, for many problems of interest, exact inference is NP-hard.
- Whether inference is *tractable* (efficiently computable) depends on the structure of the graph underlying the PGM.
- (Though if problem is intractable, can still obtain useful answers via approximate inference algorithms).

Variable Elimination Algorithm

Variable Elimination

The first algorithm we will cover here is called: ***variable elimination***.

Variable Elimination

The first algorithm we will cover here is called: ***variable elimination***.

It is an exact inference algorithm, for discrete-variable graphical models.

- Though the principles of variable elimination also apply to many continuous distributions (*e.g.*, Gaussians), but we will not discuss those extensions here.

Variable Elimination

The first algorithm we will cover here is called: ***variable elimination***.

It is an exact inference algorithm, for discrete-variable graphical models.

- Though the principles of variable elimination also apply to many continuous distributions (*e.g.*, Gaussians), but we will not discuss those extensions here.

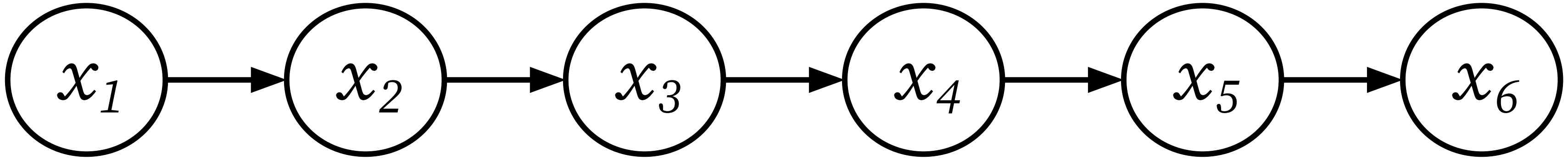
In the following slides, assume that variables x_i are discrete, each taking k possible values.

Variable Elimination — An Illustrative Example

Variable Elimination — An Illustrative Example

Consider first the problem of marginal inference.

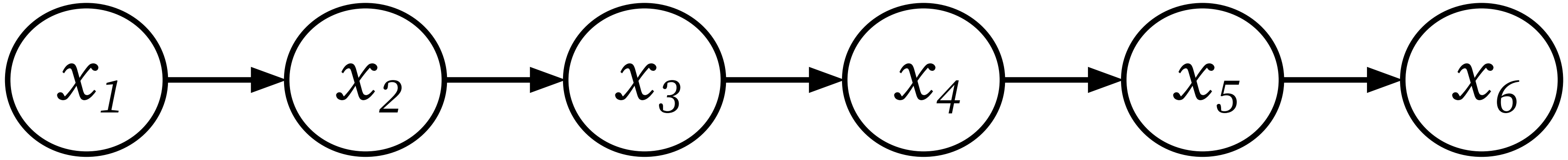
Suppose we are given this Markov chain PGM.



Variable Elimination — An Illustrative Example

Consider first the problem of marginal inference.

Suppose we are given this Markov chain PGM.



Joint PDF can be written:

Markovian property

$$p(x_1, \ldots, x_n) = p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — An Illustrative Example

Suppose our **goal** is to compute the marginal probability $p(x_n)$.

Variable Elimination — An Illustrative Example

Suppose our **goal** is to compute the marginal probability $p(x_n)$.

Naive way: sum probability over all assignments of $x_1, \ldots, x_{n-1}$:

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

$O(k^n)$ time

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — An Illustrative Example

However, we can do much better by taking advantage of our factorization!

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

Variable Elimination — An Illustrative Example

However, we can do much better by taking advantage of our factorization!

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

Swap in the factorization

Variable Elimination — An Illustrative Example

However, we can do much better by taking advantage of our factorization!

i.e., can "push" relevant sums into the product…

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

Swap in the factorization

Variable Elimination — An Illustrative Example

However, we can do much better by taking advantage of our factorization!

i.e., can "push" relevant sums into the product…

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

$$= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_1} p(x_2 \mid x_1) p(x_1)$$

"Push in" relevant sums

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — An Illustrative Example

However, we can do much better by taking advantage of our factorization!

i.e., can "push" relevant sums into the product…

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

$$= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_1} p(x_2 \mid x_1) p(x_1)$$

Start by computing an *intermediate factor* $\tau(x_2) = \sum_{x_1} p(x_2 \mid x_1) p(x_1)$

Variable Elimination — An Illustrative Example

However, we can do much better by taking advantage of our factorization!

i.e., can "push" relevant sums into the product…

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

$$= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_1} p(x_2 \mid x_1) p(x_1)$$

Start by computing an *intermediate factor* $\tau(x_2) = \sum_{x_1} p(x_2 \mid x_1) p(x_1)$

$O(k^2)$ time, since we must sum over x_1 for each assignment of x_2.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — An Illustrative Example

However, we can do much better by taking advantage of our factorization!

i.e., can "push" relevant sums into the product…

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

$$= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_1} p(x_2 \mid x_1) p(x_1)$$

Start by computing an *intermediate factor* $\tau(x_2) = \sum_{x_1} p(x_2 \mid x_1) p(x_1)$

$\Rightarrow \tau(x_2)$ can be viewed as a table of k values.

Variable Elimination — An Illustrative Example

H

How to view the factor $\tau(x_2)$ **:**

Can view it as a table of k values:

$\tau(x_2 = 1)$	$\tau(x_2 = 2)$	$\tau(x_2 = 3)$	$\tau(x_2 = 4)$	...	$\tau(x_2 = k)$

Start by computing an *intermediate factor* $\tau(x_2) = \sum_{x_1} p(x_2 \mid x_1) p(x_1)$

$\Rightarrow \tau(x_2)$ can be viewed as a table of k values.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — An Illustrative Example

However, we can do much better by taking advantage of our factorization!

i.e., can "push" relevant sums into the product…

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

Start by computing an *intermediate factor* $\tau(x_2) = \sum_{x_1} p(x_2 \mid x_1) p(x_1)$

$\Rightarrow$ $\tau(x_2)$ can be viewed as a table of k values.

Variable Elimination — An Illustrative Example

We can then rewrite our marginal probability (using factor $\tau(x_2)$) as:

Variable Elimination — An Illustrative Example

We can then rewrite our marginal probability (using factor $\tau(x_2)$) as:

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

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — An Illustrative Example

We can then rewrite our marginal probability (using factor $\tau(x_2)$) as:

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

$$= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_2} p(x_3 \mid x_2) \tau(x_2)$$

(just removed the middle line for brevity)

Variable Elimination — An Illustrative Example

We can then rewrite our marginal probability (using factor $\tau(x_2)$) as:

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

$$= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_2} p(x_3 \mid x_2) \tau(x_2)$$

⇒ Same form as previous expression, but summing over one fewer variable.

⇒ An example of *dynamic programming*.

Variable Elimination — An Illustrative Example

We can then rewrite our marginal probability (using factor $\tau(x_2)$) as:

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

$$= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_2} p(x_3 \mid x_2) \tau(x_2)$$

Next step:
Compute the next factor $\tau(x_3) = \sum_{x_2} p(x_3 \mid x_2) \tau(x_2)$, and repeat the process.

Variable Elimination — An Illustrative Example

We can then rewrite our marginal probability (using factor $\tau(x_2)$) as:

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

$$= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_2} p(x_3 \mid x_2) \tau(x_2)$$

Next step:
Compute the next factor $\tau(x_3) = \sum_{x_2} p(x_3 \mid x_2) \tau(x_2)$, and repeat the process.

⇒ Each step takes $O(k^2)$ time, and there are $O(n)$ steps ⇒ $O(nk^2)$ total time.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — An Illustrative Example

We can then rewrite our marginal probability (using factor $\tau(x_2)$) as:

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

$$= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_2} p(x_3 \mid x_2)\tau(x_2)$$

Next step:
Compute the next factor $\tau(x_3) = \sum_{x_2} p(x_3 \mid x_2)\tau(x_2)$, and repeat the process.

⇒ Each step takes $O(k^2)$ time, and there are $O(n)$ steps ⇒ $O(nk^2)$ total time.

(much better than $O(k^n)$ time).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — Note

Note: at each step we are eliminating a variable, which gives the algorithm its name.

Previous example showed the algorithm for Markov chains.

Next we'll show it in its more general form.

Variable Elimination — General Form

Variable Elimination — General Form

Assume we are given a graphical model as a product of factors:

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

View each factor $\phi_c(x_c)$ as a multi-dimensional table assigning a value to each assignment of variables in x_c, *i.e.*,

- In a Bayesian network: conditional probability tables, and
- In a Markov random field: specification of an unnormalized density.

Variable Elimination — General Form

The variable elimination algorithm repeatedly performs two operations: (1) product, and (2) marginalization.

Variable Elimination — General Form

The variable elimination algorithm repeatedly performs two operations: (1) product, and (2) marginalization.

(1) Product Operation:

Variable Elimination — General Form

The variable elimination algorithm repeatedly performs two operations: (1) product, and (2) marginalization.

(1) Product Operation:

- If we have two factors, each over a subset of variables, we can form a third factor, via a product, over the union of their variables.
- Defines a product of two factors ϕ_1, ϕ_2 as: $\phi_3(x_c) = \phi_1(x_c^{(1)}) \times \phi_2(x_c^{(2)})$
- Where $x_c^{(i)}$ denotes an assignment of the variables defined by the restriction of x_c to the scope of ϕ_i.

Variable Elimination — General Form

The variable elimination algorithm repeatedly performs two operations: (1) product, and (2) marginalization.

(2) Marginalization Operation:

Variable Elimination — General Form

The variable elimination algorithm repeatedly performs two operations: (1) product, and (2) marginalization.

(2) Marginalization Operation:

- Locally eliminates a set of variables from a factor.
- For example, if we have a factor $\phi(X, Y)$ over two sets of variables X and Y, then marginalizing Y produces a new factor:

$$\tau(x) = \sum_{y} \phi(x, y)$$

τ denotes the *marginalized factor*

where the sum is over all joint assignments of the set of variables in Y.

Variable Elimination — General Form

Question: Do variable orderings matter?

Variable Elimination — General Form

Question: Do variable orderings matter?

Short answer: Yes.

Variable Elimination — General Form

Question: Do variable orderings matter?

Short answer: Yes.

In previous example we used topological sort of the DAG.

In general, different variable orderings can dramatically alter the runtime of the algorithm.

… and it is NP-hard to determine the optimal ordering 😔 .

(Though there are heuristics that work well in practice).

For now, we will just assume the ordering is given and fixed.

Variable Elimination — General Form

Full variable elimination algorithm:

Variable Elimination — General Form

Full variable elimination algorithm:

For each variable x_i,

(1) Form a product of all factors ϕ_i containing x_i.

(2) Marginalize out x_i to obtain a new factor τ.

A function of all other variables in factor.

(3) Replace the factors ϕ_i with τ.

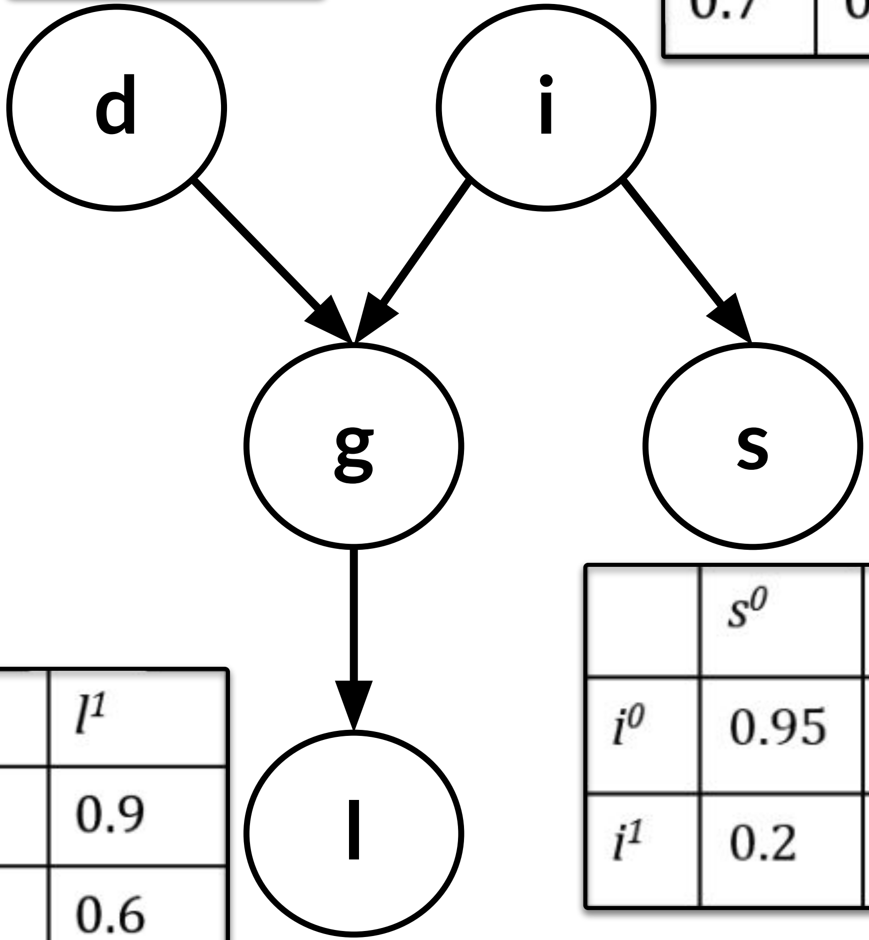
Variable Elimination — General Form

Recall the
Student PGM
from last class:

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

d - Class difficulty.

i - Student's intelligence.

g - Student's grade in the class.

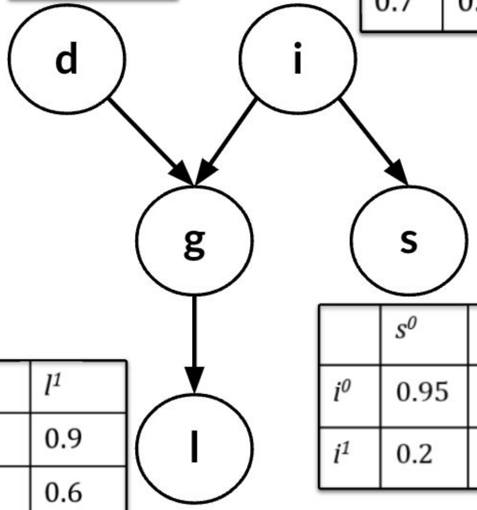
s - Student's SAT test score.

l - Professor's letter of recommendation.
(good vs bad letter)

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — General Form

Suppose we want to compute the marginal: $p(l)$



d^0	d^1
0.6	0.4

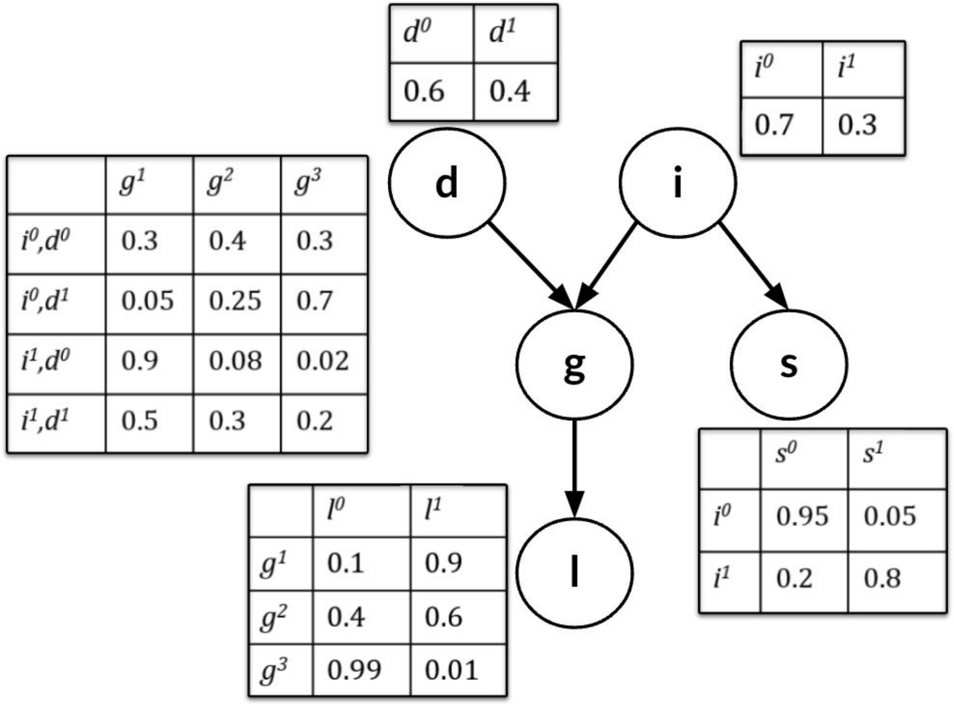
i^0	i^1
0.7	0.3

	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

Variable Elimination — General Form



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

Suppose we want to compute the marginal: $p(l)$

Recall the joint PDF (given topological ordering):

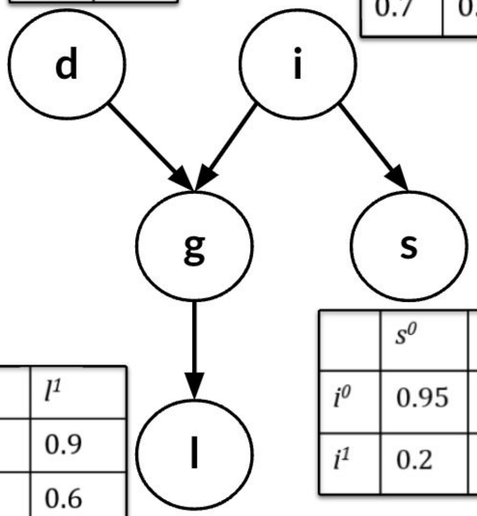
$$p(l, g, i, d, s) = p(d)\ p(i)\ p(s \mid i)\ p(g \mid i, d)\ p(l \mid g)$$

We'll use this order as the variable elimination order: d, i, s, g

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — General Form

Eliminate one variable at a time!



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

Joint PDF:

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

Variable Elimination — General Form

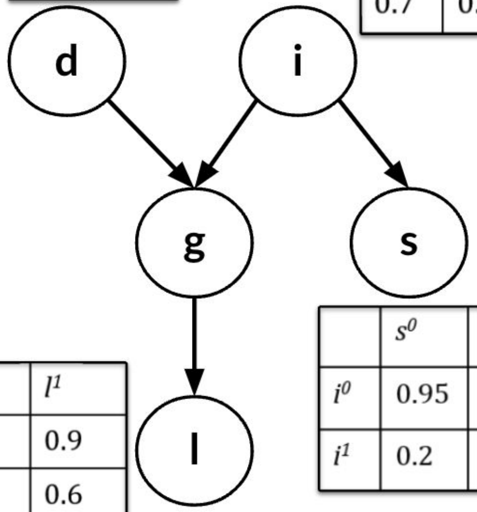
First, eliminate d:

$$\tau_1(g, i) = \sum_d p(g \mid i, d) p(d)$$

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

Joint PDF:

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

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — General Form

First, eliminate d:

$$\tau_1(g, i) = \sum_d p(g \mid i, d) p(d)$$

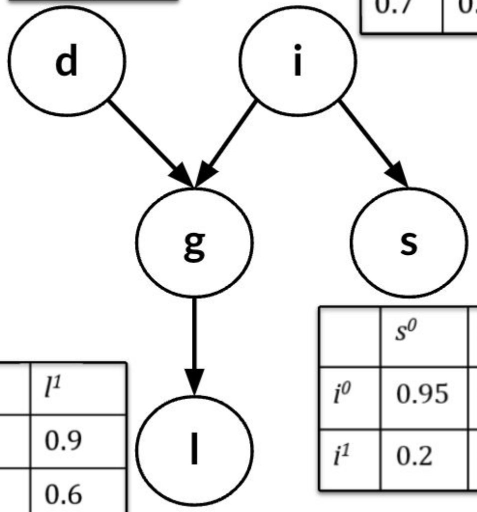
Next, eliminate i:

$$\tau_2(g, s) = \sum_i \tau_1(g, i) p(i) p(s \mid i)$$

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



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

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

Joint PDF:

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

Variable Elimination — General Form

First, eliminate d:

$$\tau_1(g, i) = \sum_d p(g \mid i, d) p(d)$$

Next, eliminate i:

$$\tau_2(g, s) = \sum_i \tau_1(g, i) p(i) p(s \mid i)$$

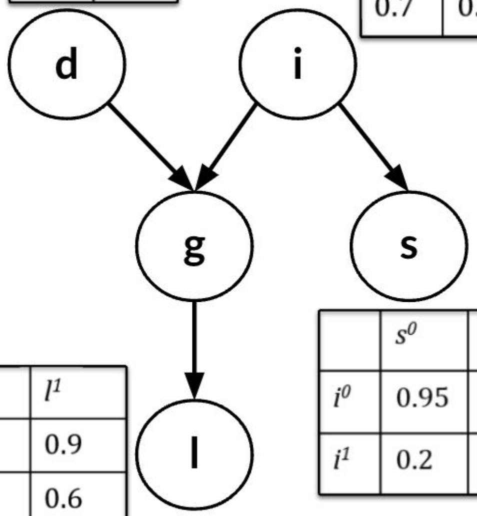
Next, eliminate s:

$$\tau_3(g) = \sum_s \tau_2(g, s)$$

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



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

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

Joint PDF:

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

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — General Form

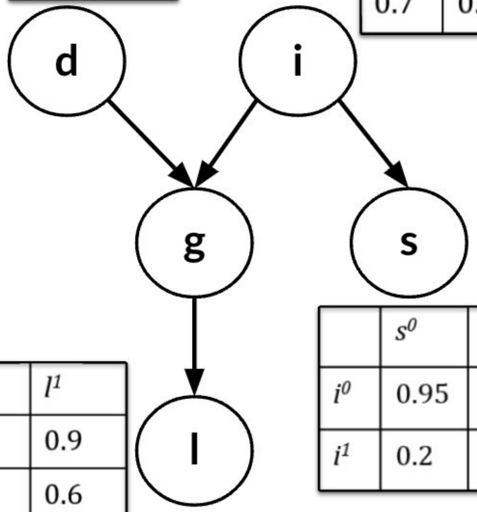
Finally, eliminate g:

$$\tau_4(l) = \sum_{g} \tau_3(g) p(l \mid g)$$

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

Joint PDF:

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

Variable Elimination — General Form

Finally, eliminate g:

$$\tau_4(l) = \sum_g \tau_3(g) p(l \mid g)$$

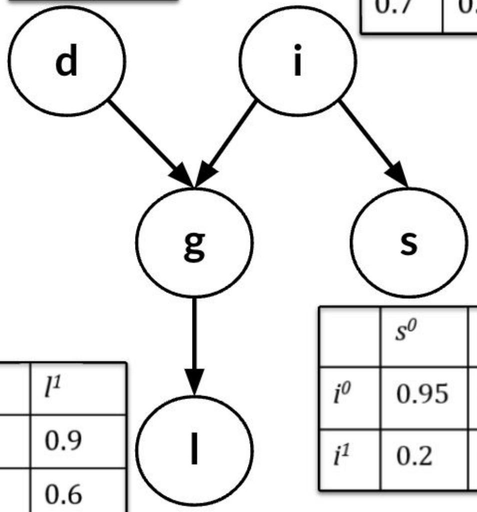
In total: at most, k^3 operations per step.

(because we have factors of ≤3 variables)

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

Joint PDF:

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

Variable Elimination — Extensions to Conditional Probabilities

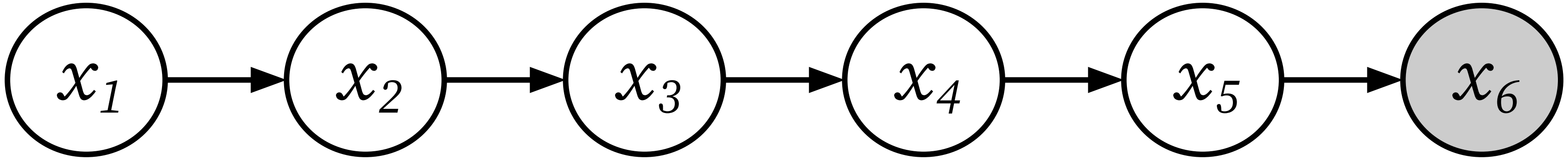
We can extend this algorithm to conditional distributions (*e.g.*, posteriors)!

Variable Elimination — Extensions to Conditional Probabilities

We can extend this algorithm to conditional distributions (*e.g.*, posteriors)!

Suppose we have a joint probability $p(x_1, \ldots, x_n)$.

And our **goal** is to compute a given conditional probability, *e.g.*, $p(x_1 \mid x_n = 5)$.

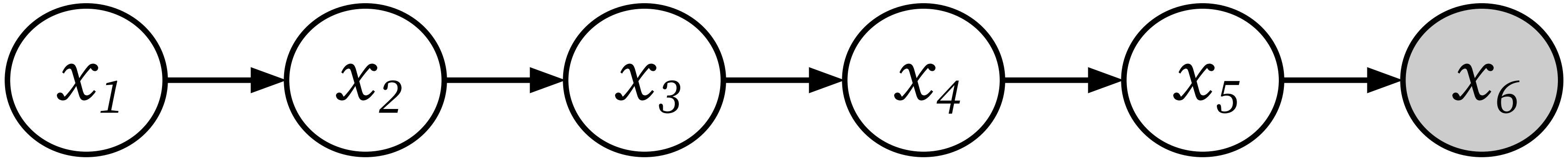


Variable Elimination — Extensions to Conditional Probabilities

We can extend this algorithm to conditional distributions (*e.g.*, posteriors)!

Suppose we have a joint probability $p(x_1, \ldots, x_n)$.

And our **goal** is to compute a given conditional probability, *e.g.*, $p(x_1 \mid x_n = 5)$.



Conditional PDF:

$$p(x_1 \mid x_n = 5) = \frac{p(x_1, x_n = 5)}{p(x_n = 5)}$$

Two marginal probabilities!

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variable Elimination — Extensions to Conditional Probabilities

To apply the variable elimination algorithm here…

Can perform variable elimination *twice*: first on $p(x_1, x_n = 5)$ then on $p(x_n = 5)$.

Variable Elimination — Extensions to Conditional Probabilities

To apply the variable elimination algorithm here...

Can perform variable elimination *twice*: first on $p(x_1, x_n = 5)$ then on $p(x_n = 5)$.

Note:

- The second step is just what we did previously.
- For the first step: begin by replacing any factor containing x_n with $x_n = 5$.
    - Then perform variable elimination as usual.

Belief Propagation Algorithm *(+ Sum-Product, Max-Product)*

Downsides of Variable Elimination

One major issue of the variable elimination algorithm:

Downsides of Variable Elimination

One major issue of the variable elimination algorithm:

Suppose we've computed $p(x_1 \mid x_n = 5)$, and now want to perform a second inference….

Such as $p(x_2 \mid x_4 = -2)$.

We would need to start the algorithm over from scratch 😔.

⇒ wasteful and computationally expensive.

Downsides of Variable Elimination

However, the variable elimination algorithm produces many intermediate factors τ as a side product of the algorithm.

Downsides of Variable Elimination

However, the variable elimination algorithm produces many intermediate factors τ as a side product of the algorithm.

These factors can be used to compute other inference queries!

$\Rightarrow$ By caching them in a first run of variable elimination, we can answer new marginal queries at essentially no additional cost.

Downsides of Variable Elimination

However, the variable elimination algorithm produces many intermediate factors τ as a side product of the algorithm.

These factors can be used to compute other inference queries!

⇒ By caching them in a first run of variable elimination, we can answer new marginal queries at essentially no additional cost.

⇒ This leads to two famous algorithms:

- **Junction Tree Algorithm.** General-purpose, for BNs and MRFs
- **Belief Propagation Algorithm.** Special case for **trees** – we will cover this here.

Belief Propagation Algorithm

Belief Propagation Algorithm

The belief propagation is a **message passing** algorithm.

⇒ *i.e.*, viewed as messages passed between nodes in a graph.

To explain the algorithm, we can start by viewing the *variable elimination algorithm* as a message passing algorithm.

Belief Propagation Algorithm

Suppose we want to compute a marginal distribution: $p(x_i)$

And assume our PGM is a tree graph, consisting of a set of factors (either BN or MRF).

Belief Propagation Algorithm

Suppose we want to compute a marginal distribution: $p(x_i)$

And assume our PGM is a tree graph, consisting of a set of factors (either BN or MRF).

We can choose the following ordering of nodes:

- Set x_i to be the root of the graph.
- Iterate through the nodes in "post-order".

i.e., each node is visited only after all of its child nodes have been visited

Belief Propagation Algorithm

Suppose we want to compute a marginal distribution: $p(x_i)$

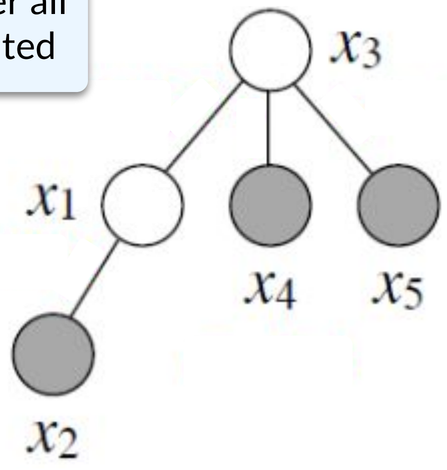
And assume our PGM is a tree graph, consisting of a set of factors (either BN or MRF).

We can choose the following ordering of nodes:

- Set x_i to be the root of the graph.
- Iterate through the nodes in "post-order".

i.e., each node is visited only after all of its child nodes have been visited

E.g., setting x_3 as the root.



Belief Propagation Algorithm

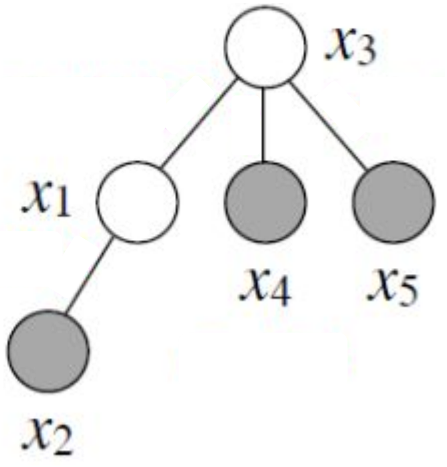
Suppose we want to compute a marginal distribution: $p(x_i)$

And assume our PGM is a tree graph, consisting of a set of factors (either BN or MRF).

We can choose the following ordering of nodes:

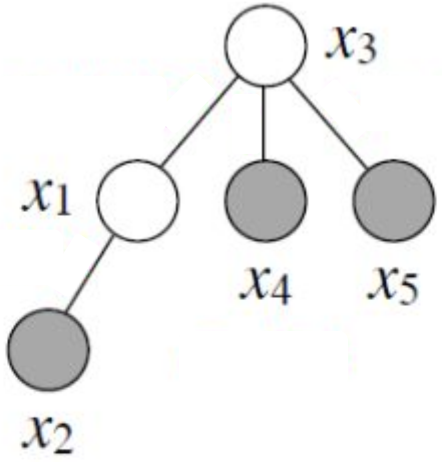
- Set x_i to be the root of the graph.
- Iterate through the nodes in "post-order".

Variable elimination in this graph?



Belief Propagation Algorithm

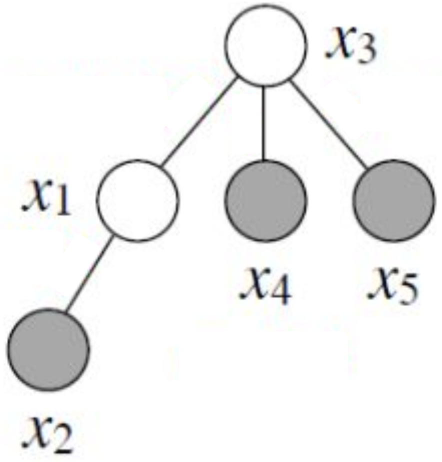
Variable elimination in this graph?



Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Belief Propagation Algorithm

Variable elimination in this graph?

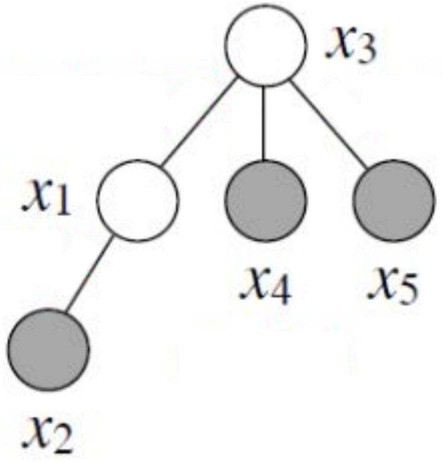


At each step: we eliminate a single node x_j.

⇒ Compute: $\tau_k(x_k) = \sum_{x_j} \phi(x_k, x_j)\tau_j(x_j)$

(where x_j is a child of x_k).

Belief Propagation Algorithm



Variable elimination in this graph?

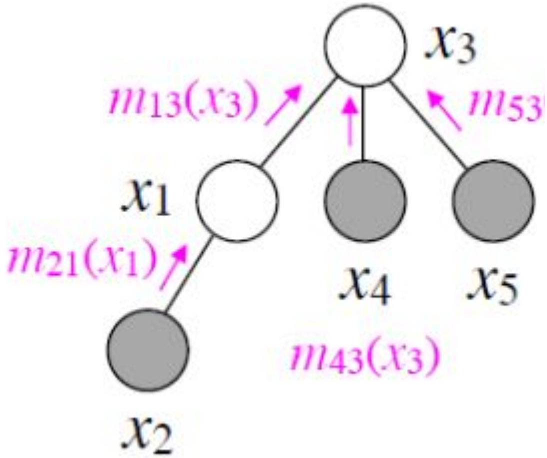
At each step: we eliminate a single node x_j.

⇒ Compute: $\tau_k(x_k) = \sum_{x_j} \phi(x_k, x_j)\tau_j(x_j)$

(where x_j is a child of x_k).

After: x_k will in turn be eliminated, and $\tau(x_k)$ passed up the tree to its parent.

Belief Propagation Algorithm



Variable elimination in this graph?

At each step: we eliminate a single node x_j.

⇒ Compute: $\tau_k(x_k) = \sum_{x_j} \phi(x_k, x_j)\tau_j(x_j)$

(where x_j is a child of x_k).

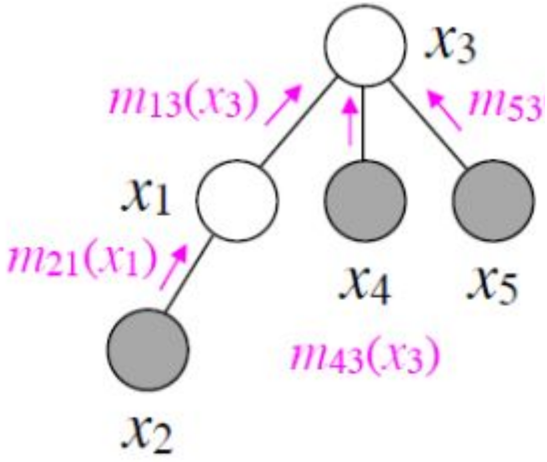
After: x_k will in turn be eliminated, and $\tau(x_k)$ passed up the tree to its parent.

⇒ We can view $\tau_j(x_j)$ as a message that x_j sends to x_k.

Belief Propagation Algorithm

At the end of variable elimination:

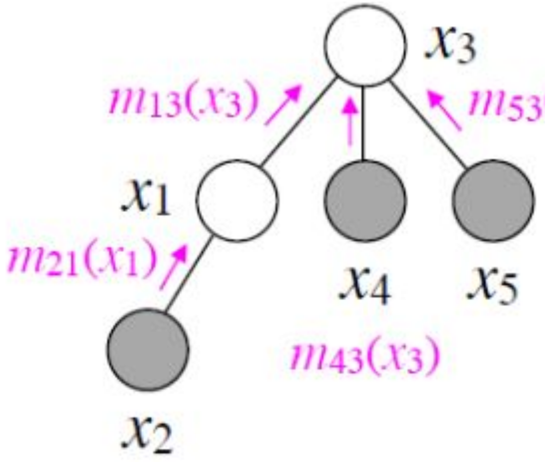
- x_i receives messages from all its children.
- Marginalizes them out.
- And this yields the final marginal, $p(x_i)$.



Belief Propagation Algorithm

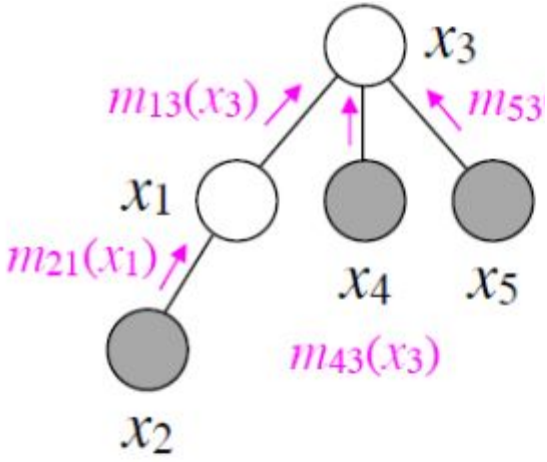
At the end of variable elimination:

- x_i receives messages from all its children.
- Marginalizes them out.
- And this yields the final marginal, $p(x_i)$.



Suppose, after, we want to compute marginal $p(x_k)$ as well.

Belief Propagation Algorithm



At the end of variable elimination:

- x_i receives messages from all its children.
- Marginalizes them out.
- And this yields the final marginal, $p(x_i)$.

Suppose, after, we want to compute marginal $p(x_k)$ as well.

- We would again run variable elimination, with x_k as the root.
- **A key insight**: the messages x_k receives from x_j now are the same as those received when x_i was the root.
- $\Rightarrow$ at least some messages are already computed.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Belief Propagation Algorithm

How do we make sure we've computed all the messages that we need?

Belief Propagation Algorithm

How do we make sure we've computed all the messages that we need?

The answer is very simple — stick to the following procedure:

Each node x_i sends a message to its neighbor x_j whenever it has received a message from all nodes besides x_j.

Belief Propagation Algorithm

How do we make sure we've computed all the messages that we need?

The answer is very simple — stick to the following procedure:

Each node x_i sends a message to its neighbor x_j whenever it has received a message from all nodes besides x_j.

Note that:

In a tree, there is always a node with a message to send, *unless all messages* have already been sent out!

This happens after (exactly) $2|E|$ steps.

Since each edge only "sees" two messages one from $x_i \rightarrow x_j$ and one from the other direction.

Belief Propagation Algorithm — Formal Definition

Belief Propagation Algorithm — Formal Definition

Definition. The belief propagation algorithm on tree graphs has two variants: *sum-product message passing*, and *max-product message passing*.

Belief Propagation Algorithm — Formal Definition

Definition. The belief propagation algorithm on tree graphs has two variants: *sum-product message passing*, and *max-product message passing*.

Sum-product message passing algorithm:

Used for marginal/conditional inference, *i.e.*, computing $p(x_i)$ or $p(x_i \mid x_k)$.

Belief Propagation Algorithm — Formal Definition

Definition. The belief propagation algorithm on tree graphs has two variants: *sum-product message passing*, and *max-product message passing*.

Sum-product message passing algorithm:

Used for marginal/conditional inference, *i.e.*, computing $p(x_i)$ or $p(x_i \mid x_k)$.

Max-product message passing algorithm:

Used for MAP (*maximum a posteriori*) inference, *i.e.*, computing:

$$\max_{x_1,\ldots,x_n} p(x_1,\ldots,x_n) \quad \text{or} \quad \max_{x_i} p(x_i \mid x_k) \quad \text{or} \quad \arg\max_{x_i} p(x_i \mid x_k)$$

Belief Propagation Algorithm — Formal Definition

Sum-product message passing algorithm.

Belief Propagation Algorithm — Formal Definition

Sum-product message passing algorithm.

Once a node x_i is ready to transmit to node x_j, it sends the message:

$$m_{i \to j}(x_j) = \sum_{x_i} \phi(x_i)\phi(x_i, x_j) \prod_{\ell \in N(i) \backslash j} m_{\ell \to i}(x_i)$$

Message is a function of x_j

Belief Propagation Algorithm — Formal Definition

Sum-product message passing algorithm.

Once a node x_i is ready to transmit to node x_j, it sends the message:

$$m_{i \to j}(x_j) = \sum_{x_i} \phi(x_i)\phi(x_i, x_j) \prod_{\ell \in N(i) \backslash j} m_{\ell \to i}(x_i)$$

Sum over all values of x_i.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Belief Propagation Algorithm — Formal Definition

Sum-product message passing algorithm.

Once a node x_i is ready to transmit to node x_j, it sends the message:

$$m_{i \to j}(x_j) = \sum_{x_i} \phi(x_i)\phi(x_i, x_j) \prod_{\ell \in N(i) \setminus j} m_{\ell \to i}(x_i)$$

"Local" conditional probability or factor(s).

(In undirected tree graph, all cliques are either pairs or singletons.)

Belief Propagation Algorithm — Formal Definition

Sum-product message passing algorithm.

Once a node x_i is ready to transmit to node x_j, it sends the message:

$$m_{i\to j}(x_j) = \sum_{x_i} \phi(x_i)\phi(x_i, x_j) \prod_{\ell \in N(i)\backslash j} m_{\ell \to i}(x_i)$$

Product over all "other" neighbors of x_i, (besides x_j).

Belief Propagation Algorithm — Formal Definition

Sum-product message passing algorithm.

Once a node x_i is ready to transmit to node x_j, it sends the message:

$$m_{i \to j}(x_j) = \sum_{x_i} \phi(x_i)\phi(x_i, x_j) \prod_{\ell \in N(i) \backslash j} m_{\ell \to i}(x_i)$$

Message from neighbor.

Belief Propagation Algorithm — Formal Definition

Sum-product message passing algorithm.

And after computing all messages, can answer any inference query in constant time, using:

$$p(x_i) \propto \phi(x_i) \prod_{\ell \in N(i)} m_{\ell \to i}(x_i)$$

Belief Propagation Algorithm — Formal Definition

Sum-product message passing algorithm.

And after computing all messages, can answer any inference query in constant time, using:

$$p(x_i) \propto \phi(x_i) \prod_{\ell \in N(i)} m_{\ell \to i}(x_i)$$

"Proportional to"

Belief Propagation Algorithm — Formal Definition

Sum-product message passing algorithm.

And after computing all messages, can answer any inference query in constant time, using:

$$p(x_i) \propto \phi(x_i) \prod_{\ell \in N(i)} m_{\ell \to i}(x_i)$$

Product of all incoming factors
(along with any local factor)

Belief Propagation Algorithm — Formal Definition

Sum-product message passing algorithm.

And after computing all messages, can answer any inference query in constant time, using:

$$p(x_i) \propto \phi(x_i) \prod_{\ell \in N(i)} m_{\ell \to i}(x_i)$$

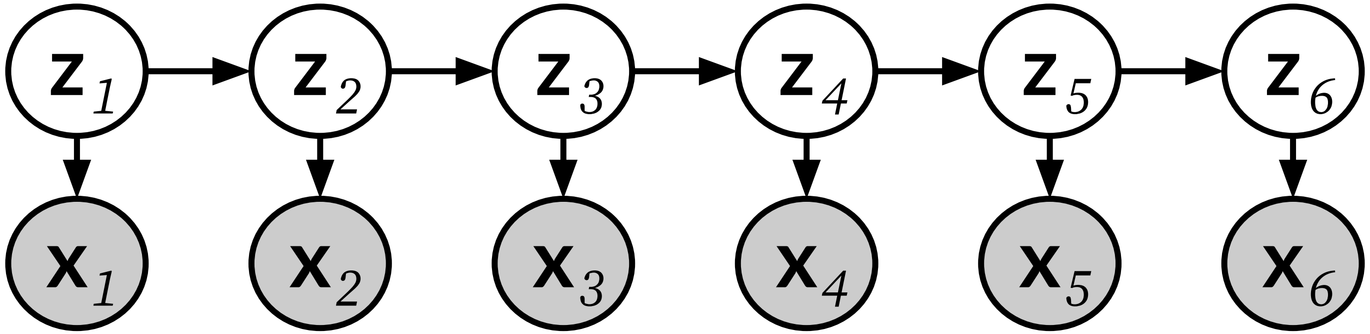
Note: this defines the final product operation (conditional probability table) in variable elimination!

Belief Propagation Algorithm — Famous Example

Belief Propagation Algorithm — Famous Example

Famous Example of Sum-Product Algorithm:

Forward-Backward Algorithm in HMMs!



Suppose we want to query $p(z_i \mid x_1, \ldots, x_n)$ for all $z_i \in \{z_1, \ldots, z_n\}$.

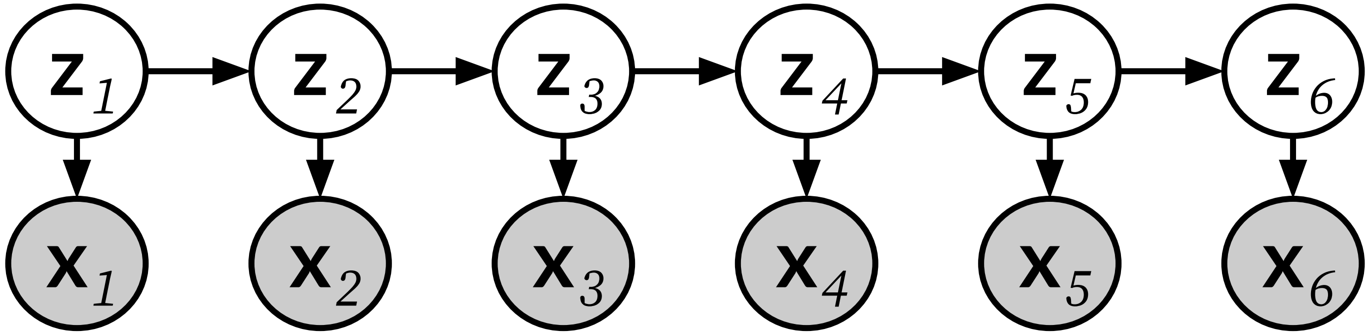
Belief Propagation Algorithm — Example

Partial inference

(1) First take a *forward pass* (message passing), which computes: $p(z_i \mid x_1, \ldots, x_i)$

(2) Then take a *backward pass* (message passing), which computes: $p(z_i \mid x_1, \ldots, x_n)$

Full inference



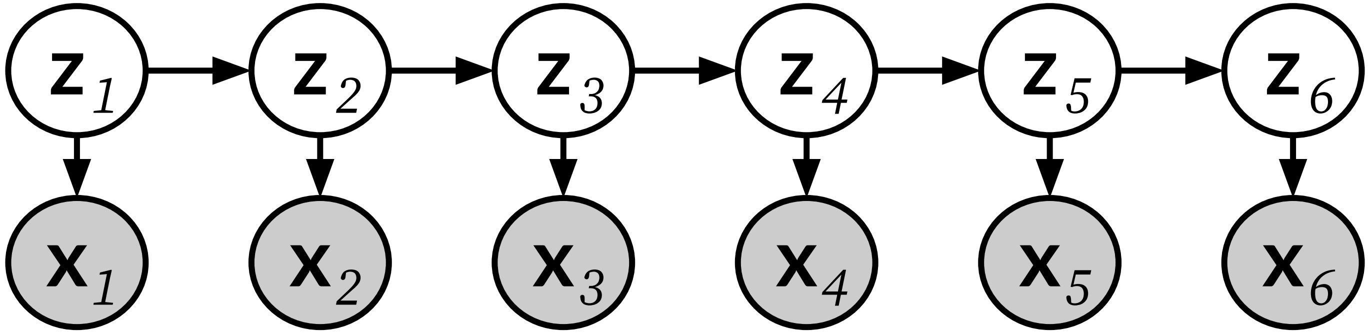
Belief Propagation Algorithm — Example

Partial inference

(1) First take a *forward pass* (message passing), which computes: $p(z_i \mid x_1, \ldots, x_i)$

(2) Then take a *backward pass* (message passing), which computes: $p(z_i \mid x_1, \ldots, x_n)$

Full inference



Often called "smoothing".

Applications: speech recognition, POS tagging, bioinformatics, financial forecasting.

Belief Propagation Algorithm — Formal Definition

Max-product message passing algorithm $\rightarrow$ Pretty similar!

Belief Propagation Algorithm — Formal Definition

Max-product message passing algorithm $\rightarrow$ Pretty similar!

Key observations:

Sum and max operators both distribute over products.

Thus, replacing sums in marginal inference with maxes, we can solve the MAP (*maximum a posteriori*) inference problem.

Belief Propagation Algorithm — Formal Definition

Max-product message passing algorithm $\rightarrow$ Pretty similar!

Once a node x_i is ready to transmit to node x_j, it sends the message:

$$m_{i \to j}(x_j) = \max_{x_i} \phi(x_i)\phi(x_i, x_j) \prod_{\ell \in N(i) \backslash j} m_{\ell \to i}(x_i)$$

Sum changes to max here!

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Belief Propagation Algorithm — Famous Example

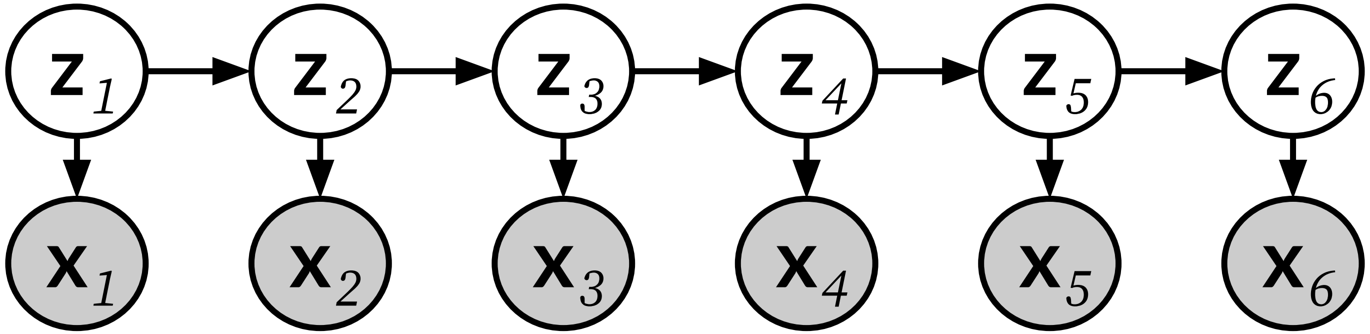
Famous Example of Max-Product Algorithm:

Viterbi Algorithm in HMMs!

Belief Propagation Algorithm — Famous Example

Famous Example of Max-Product Algorithm:

Viterbi Algorithm in HMMs!



MAP estimate of latent z's

Suppose we want to query: $$\max_{z_1,\ldots,z_n} p(z_1,\ldots,z_n \mid x_1,\ldots,x_n)$$

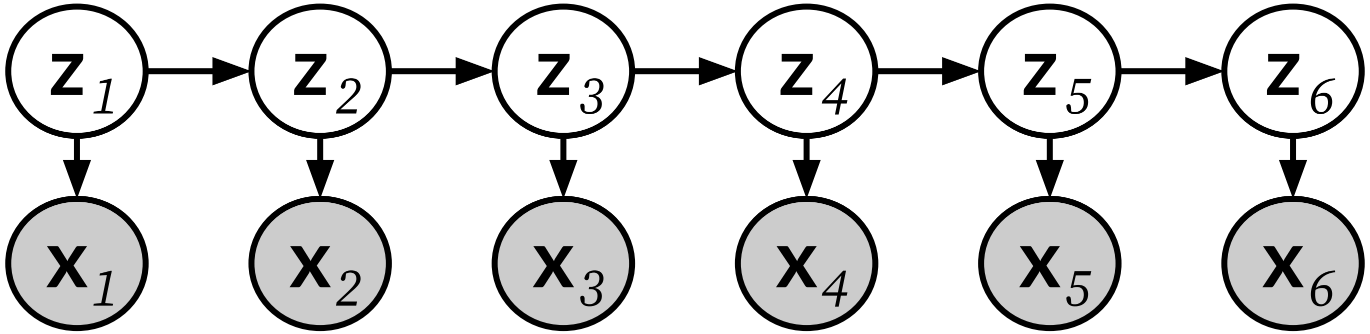
Often called "Viterbi Path"

Joint maximum over all n .

Belief Propagation Algorithm — Famous Example

Note: is a little more complex than max-product algorithm above, as we actually want to compute: $\arg\max_{z_1,\ldots,z_n} p(z_1, \ldots, z_n \mid x_1, \ldots, x_n)$

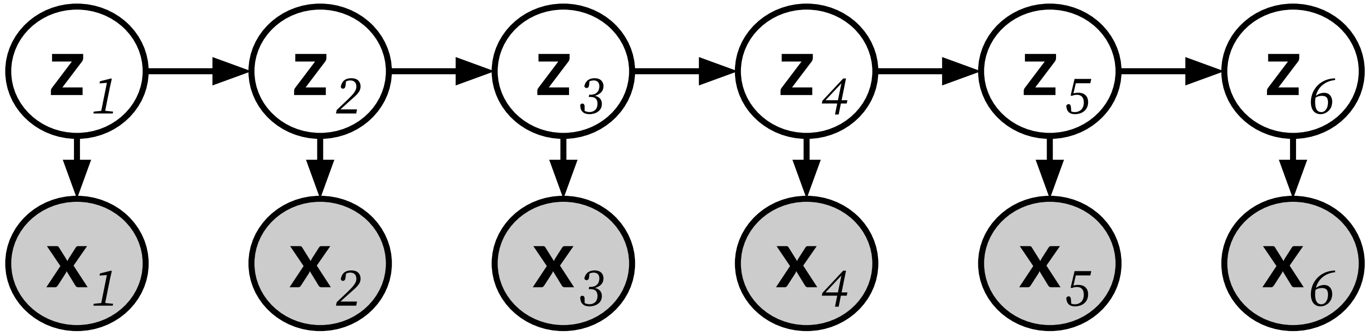
Use a technique called *backpointers* (tracks which index produced max at each node).



Belief Propagation Algorithm — Famous Example

Note: is a little more complex than max-product algorithm above, as we actually want to compute: $\arg\max_{z_1,\ldots,z_n} p(z_1,\ldots,z_n \mid x_1,\ldots,x_n)$

Use a technique called *backpointers* (tracks which index produced max at each node).



Often called "*decoding*" or "*Viterbi decoding*".

Applications: digital cellular, modems, satellite, deep-space communications, LANs, etc.

Expectation Maximization (EM) Algorithm

Learning in PGMs

Learning in PGMs

So far we've covered classic inference algorithms for:

- Marginal distribution inference.
- Conditional distribution inference (*e.g.*, posterior inference).
- MAP (*maximum a posteriori*) inference.

Where we've looked at exact algorithms in PGMs with discrete random variables.

Learning in PGMs

So far we've covered classic inference algorithms for:

- Marginal distribution inference.
- Conditional distribution inference (*e.g.*, posterior inference).
- MAP (*maximum a posteriori*) inference.

Where we've looked at exact algorithms in PGMs with discrete random variables.

Suppose want to do *learning* in PGMs.

⇒ *E.g.*, estimating parameters (not random variables) defining our probabilistic model:

Given $p_\theta(x_1, \ldots, x_n)$, produce an estimate $\hat{\theta}$.

Expectation Maximization Algorithm

Expectation maximization (EM) algorithm is a famous algorithm, which can be remembered as

Expectation Maximization Algorithm

Expectation maximization (EM) algorithm is a famous algorithm, which can be remembered as

Maximum likelihood estimation in models with latent variables.

Expectation Maximization Algorithm

Expectation maximization (EM) algorithm is a famous algorithm, which can be remembered as

Maximum likelihood estimation in models with latent variables.

Note: this is a key parameter estimation algorithm for PGMs — more general than inference/MAP in discrete-variable PGMs with tree structure.

(*E.g.*, which we assumed for belief propagation algorithms)

Expectation Maximization Algorithm — Maximum Likelihood

Expectation Maximization Algorithm — Maximum Likelihood

Definition. Suppose we have a joint PDF p_θ with parameter θ, and we've observed data $x_1, \ldots, x_n$.

Expectation Maximization Algorithm — Maximum Likelihood

Definition. Suppose we have a joint PDF p_θ with parameter θ, and we've observed data $x_1, \ldots, x_n$.

Define the *maximum likelihood estimate (MLE)* for θ, given $x_1, \ldots, x_n$, to be:

$$\hat{\theta}_{\mathrm{MLE}} \;=\; \arg\max_\theta \mathcal{L}_n(\theta) \;=\; \arg\max_\theta p_\theta(x_1, \ldots, x_n)$$

Expectation Maximization Algorithm — Maximum Likelihood

Definition. Suppose we have a joint PDF p_θ with parameter θ, and we've observed data $x_1, \dots, x_n$.

Define the *maximum likelihood estimate (MLE)* for θ , given $x_1, \dots, x_n$, to be:

$$\hat{\theta}_{\text{MLE}} \;=\; \arg\max_\theta \mathcal{L}_n(\theta) \;=\; \arg\max_\theta p_\theta(x_1, \dots, x_n)$$

Where $\mathcal{L}_n(\theta)$ is the *likelihood function.*

⇒ Equal to the joint PDF (given observed values) but viewed as a function of θ .

Expectation Maximization Algorithm

Consider MLE in the context of PGMs.

Expectation Maximization Algorithm

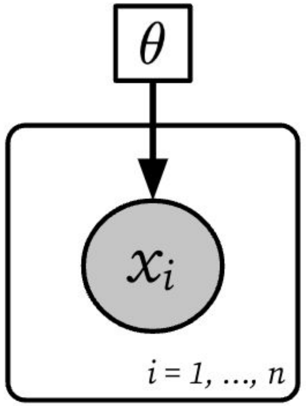
Consider MLE in the context of PGMs.

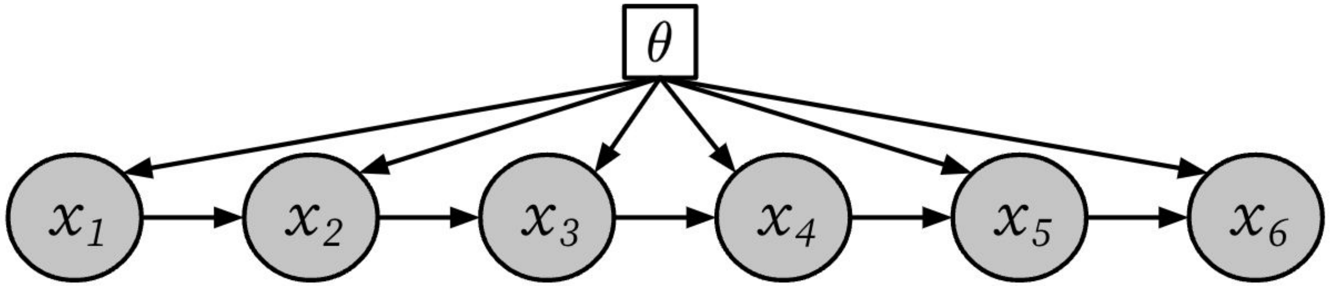
If we have a simple PGM (*e.g.*, fully observed Bayes net below), then MLE is “easy”!

Expectation Maximization Algorithm

Consider MLE in the context of PGMs.

If we have a simple PGM (*e.g.*, fully observed Bayes net below), then MLE is "easy"!



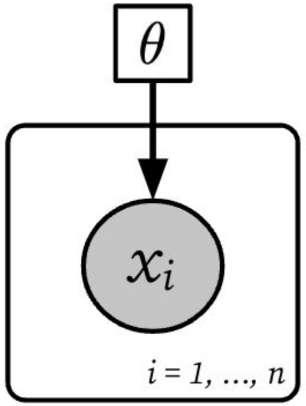


n independent samples from a distribution (*e.g*, with parameter θ)

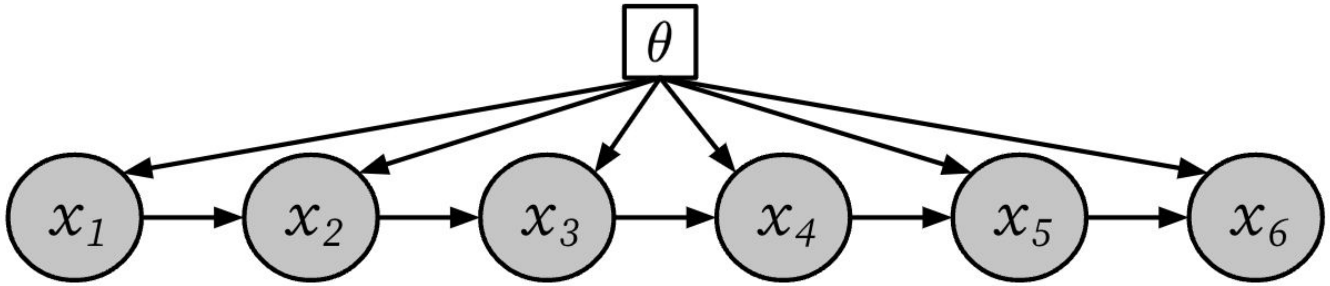
Sample path from a Markov chain (*e.g*, with transition distribution parameterized by θ)

Expectation Maximization Algorithm

⇒ To compute MLE, you optimize θ with respect to (log) likelihood (*i.e.*, joint PDF).



n independent samples from a distribution
(*e.g*, with parameter θ)



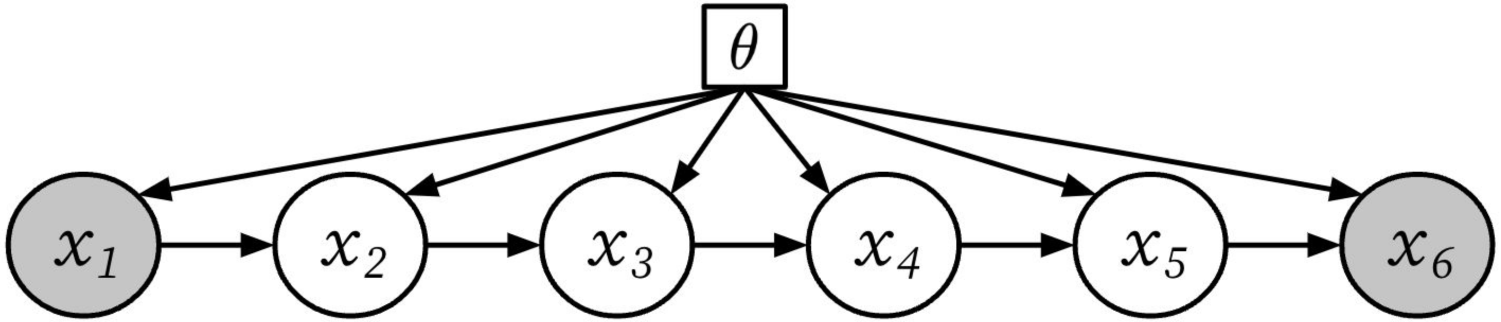
Sample path from a Markov chain
(*e.g*, with transition distribution parameterized by θ)

Expectation Maximization Algorithm

However, everything gets trickier when you have latent variables in your model!

Expectation Maximization Algorithm

However, everything gets trickier when you have latent variables in your model!

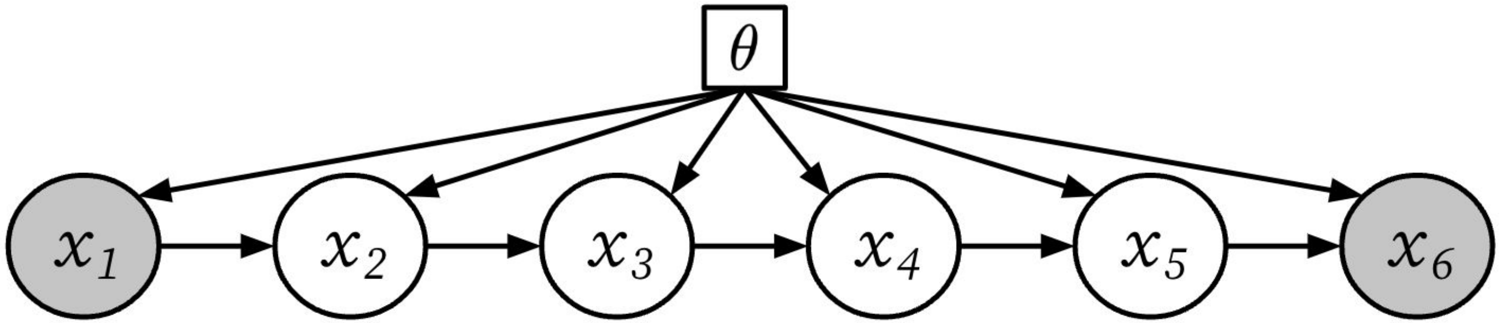


Partially observed sample path from a Markov chain
(*e.g*, with transition distribution parameterized by θ)

Expectation Maximization Algorithm

However, everything gets trickier when you have latent variables in your model!

Then you need to optimize θ while also marginalizing out latent variables…



Partially observed sample path from a Markov chain
(*e.g*, with transition distribution parameterized by θ)

Expectation Maximization Algorithm

E.g., consider our Gaussian Mixture Model (GMM) from last class… **(Quick review!)**

Expectation Maximization Algorithm

E.g., consider our Gaussian Mixture Model (GMM) from last class… **(Quick review!)**

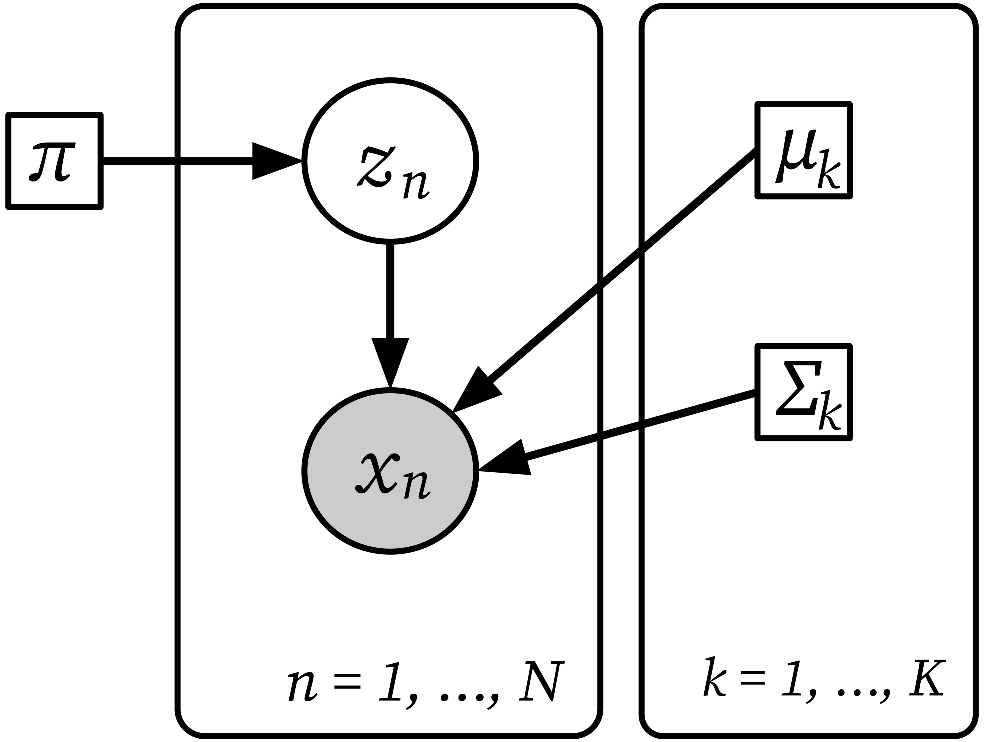
Variables in PGM:

π - Mixture weights parameters.

z_n - Assignment variables
(one per observation)

x_n - Observation variables
(corresponds to dots in prev. image)

μ_k, Σ_k - Parameters of mixture densities.

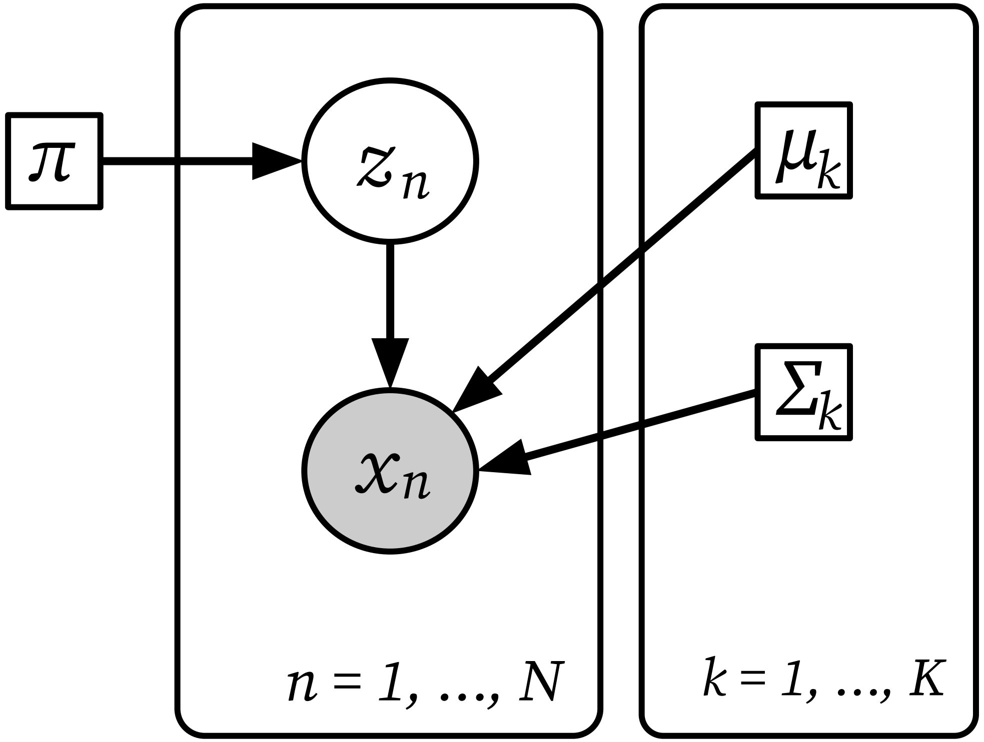


Expectation Maximization Algorithm

E.g., consider our Gaussian Mixture Model (GMM) from last class… **(Quick review!)**

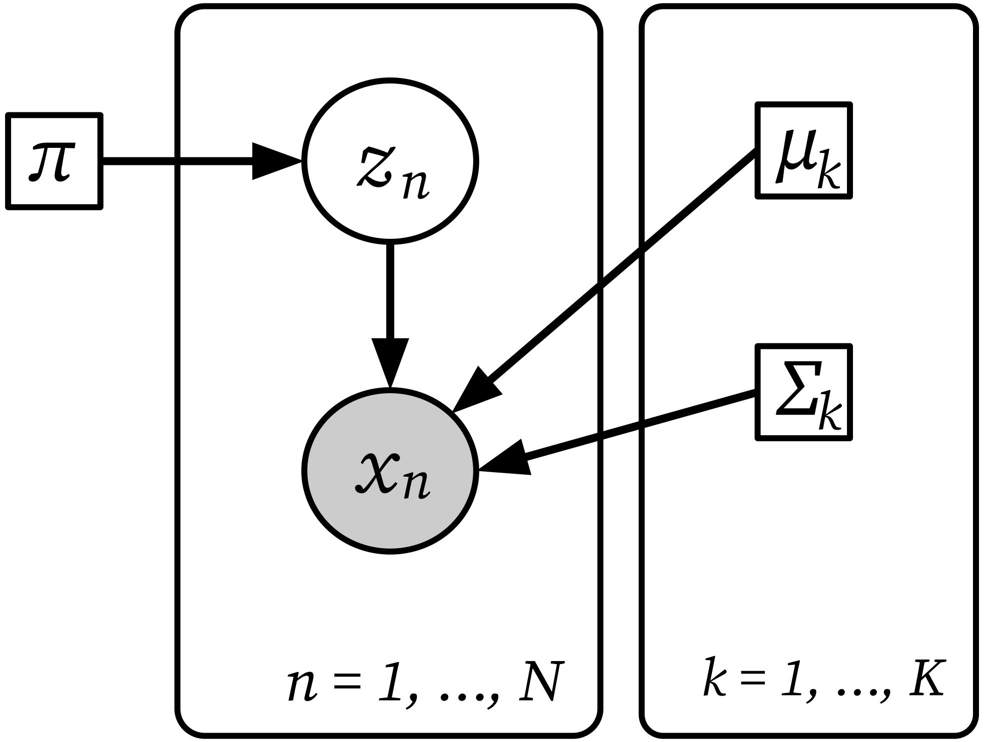
Generative process:

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



Expectation Maximization Algorithm

Suppose we want to estimate the parameters: π, μ_k, Σ_k



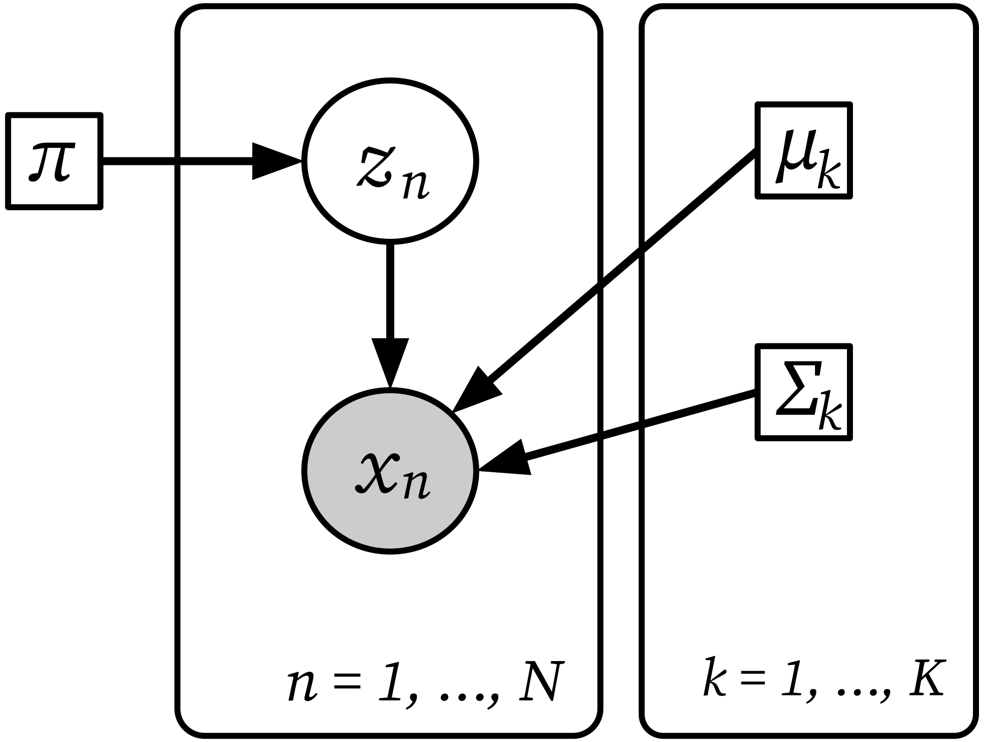
Source: Inspired by Herman Kamper

Expectation Maximization Algorithm

Suppose we want to estimate the parameters: π, μ_k, Σ_k

Chicken and egg problem!

- If we knew the parameters, then we could do inference to infer the latents.
- If we knew the latents, then we could run MLE (optimize) to estimate the parameters.



Source: Inspired by Herman Kamper

Expectation Maximization Algorithm

Intuition behind expectation maximization (EM) algorithm:

Expectation Maximization Algorithm

Intuition behind expectation maximization (EM) algorithm:

We iteratively perform two steps:

Expectation Maximization Algorithm

Intuition behind expectation maximization (EM) algorithm:

We iteratively perform two steps:

- Make an initial guess of the parameters, $\theta^{(0)}$.
- Infer the latent variables.

Expectation Maximization Algorithm

Intuition behind expectation maximization (EM) algorithm:

We iteratively perform two steps:

- Make an initial guess of the parameters, $\theta^{(0)}$.
- Infer the latent variables.
- Run MLE to estimate the parameters, denote them as $\theta^{(1)}$.
- Infer the latent variables.

Source: Inspired by Herman Kamper

Expectation Maximization Algorithm

Intuition behind expectation maximization (EM) algorithm:

We iteratively perform two steps:

- Make an initial guess of the parameters, $\theta^{(0)}$.
- Infer the latent variables.
- Run MLE to estimate the parameters, denote them as $\theta^{(1)}$.
- Infer the latent variables.
- … *repeat until convergence.*

Source: Inspired by Herman Kamper

Expectation Maximization Algorithm

Let’s investigate these two steps:

Expectation Maximization Algorithm

Let's investigate these two steps:

Step (1): "The E Step" – infer the latent variables:

$$\mathbb{E}_{z \sim p_\theta(z|x)} [\log p_\theta(x, z)]$$

⇒ The expected value of the (log) likelihood function, w.r.t. the conditional distribution of latents given observed, and current estimates of the parameters.

Source: Inspired by Herman Kamper

Expectation Maximization Algorithm

Let's investigate these two steps:

Step (1): "The E Step" – infer the latent variables:

Log-likelihood of joint PDF (defined on latents and observations)

$$\mathbb{E}_{z \sim p_\theta(z|x)} \left[\log p_\theta(x, z) \right]$$

⇒ The expected value of the (log) likelihood function, w.r.t. the conditional distribution of latents given observed, and current estimates of the parameters.

Source: Inspired by Herman Kamper

Expectation Maximization Algorithm

Let's investigate these two steps:

Step (1): "The E Step" — infer the latent variables:

$$\mathbb{E}_{z \sim p_\theta(z|x)} \left[\log p_\theta(x, z)\right]$$

In practice, you need to do inference to compute $p_\theta(z \mid x)$ in the expectation.

Depending on the inferred distribution, this expectation might be computed exactly, or estimated via Monte Carlo sampling (discussed in more detail next week) — which is often called *MC (Monte Carlo) EM*.

Expectation Maximization Algorithm

Let’s investigate these two steps:

Expectation Maximization Algorithm

Let’s investigate these two steps:

Step (2): “The M Step” – Run MLE to estimate the parameters:

$$\theta \leftarrow \arg\max_\theta \mathbb{E}_{z\sim p_\theta(z|x)} [\log p_\theta(x, z)]$$

Expectation Maximization Algorithm

Let's investigate these two steps:

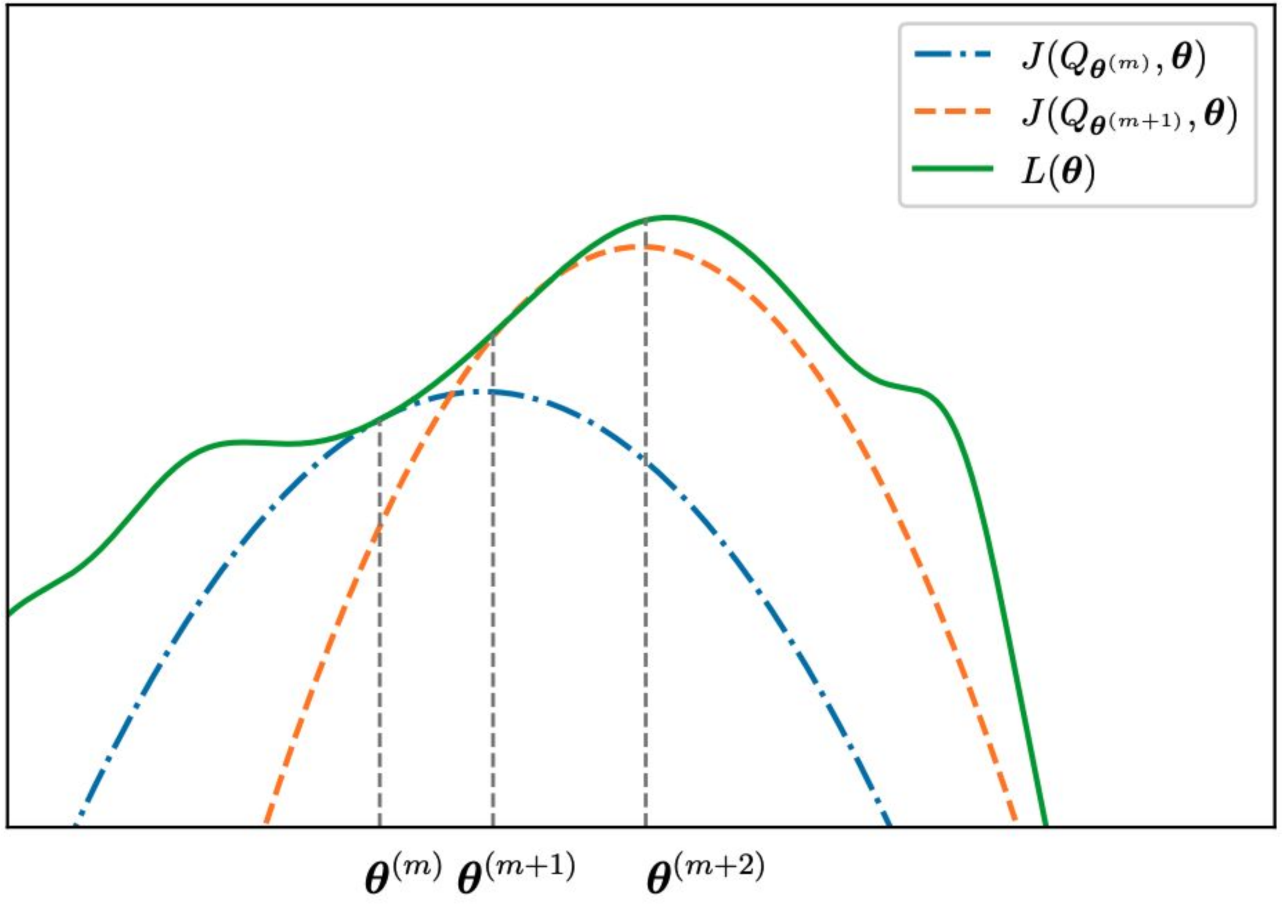
Step (2): "The M Step" – Run MLE to estimate the parameters:

$$\theta \leftarrow \arg\max_\theta \, \mathbb{E}_{z \sim p_\theta(z|x)} \left[\log p_\theta(x, z)\right]$$

Given our expression for the expected log-likelihood (from the E-step), optimize as usual to get an estimate for the MLE.

Expectation Maximization Algorithm

Intuition visualized:



Source: Inspired by Herman Kamper

Expectation Maximization Algorithm

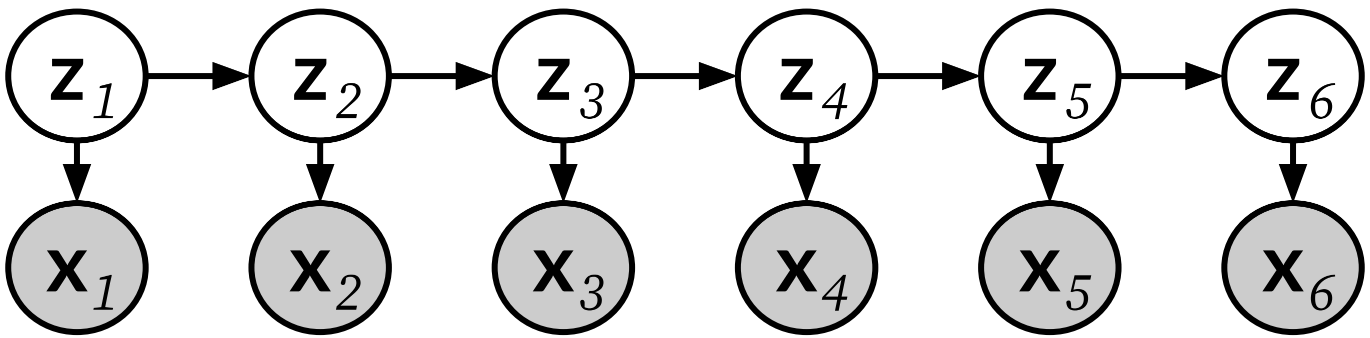
Famous Example of EM Algorithm:

Expectation Maximization Algorithm

Famous Example of EM Algorithm:

Baum–Welch Algorithm in HMMs!

Lloyd Welch is also USC affiliated :-).



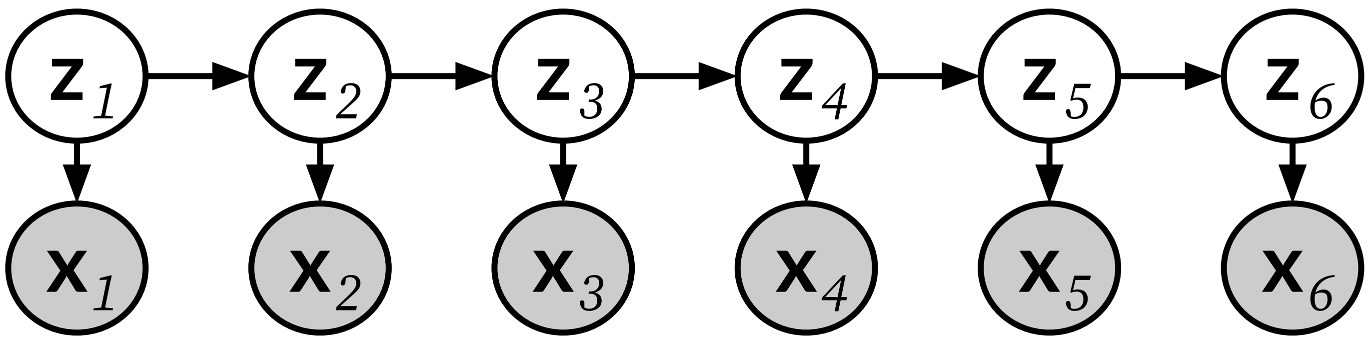
Goal: estimate parameters of HMM (*i.e.*, parameters of: initial state distribution, transition distribution, and emission distributions).

Expectation Maximization Algorithm

Famous Example of EM Algorithm:

Baum–Welch Algorithm in HMMs!

Lloyd Welch is also USC affiliated :-).



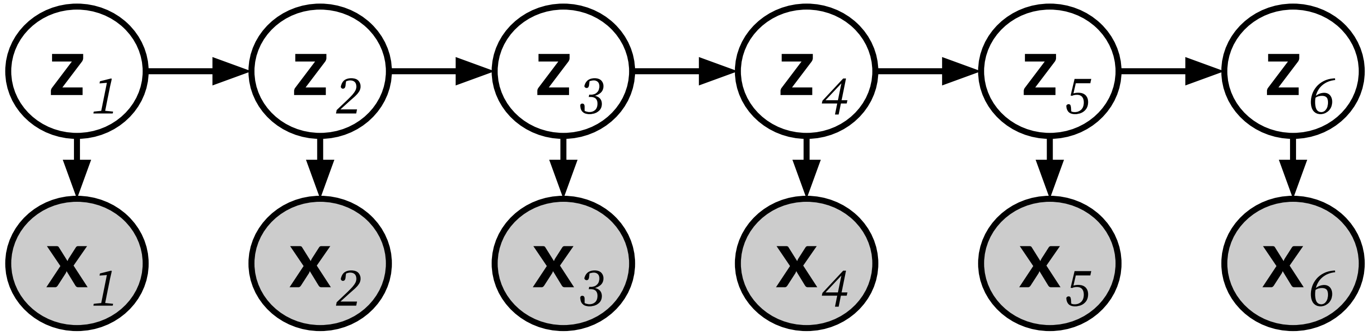
Makes use of the *forward-backward algorithm* to compute the inference for the expectation step!

Expectation Maximization Algorithm

Famous Example of EM Algorithm:

Baum–Welch Algorithm in HMMs!

Lloyd Welch is also USC affiliated :-).



Applications: cryptanalysis, speech recognition (using HMM models), genomics (identifying coding regions in DNA)

Summary of Today

Summary of Today

Lecture: Classic algorithms in probabilistic graphical models (PGMs) for exact and approximate inference & learning.

- Exact inference algorithms in PGMs.
- Variable Elimination algorithm.
- Belief Propagation algorithm (sum/max-product message passing).
- Famous algorithms: Forward-Backward & Viterbi.
- Expectation-Maximization (EM) algorithm.

Source: USC Viterbi Magazine, “The Viterbi Algorithm at 50”

Summary of Today

Lecture: Classic algorithms in probabilistic graphical models (PGMs) for exact and approximate inference & learning.

- Exact inference algorithms in PGMs.
- Variable Elimination algorithm.
- Belief Propagation algorithm (sum/max-product message passing).
- Famous algorithms: Forward-Backward & Viterbi.
- Expectation-Maximization (EM) algorithm.

Source: USC Viterbi Magazine, "The Viterbi Algorithm at 50"

After:

- Check-in on group formation and project ideation.
- Tutorial on CARC access and cluster/slurm usage.

Info on Project Pitches

Assignments and Grading — To Summarize…

Assignment	**% of Grade**
1. Paper Presentation *(Individual)*	20%
2. In-class Participation and Discussion *(Individual)*	
2a. Role 1 — Discussion Lead 1	8%
2a. Role 2 — Discussion Lead 2	8%
2a. Role 3 — Scribe	9%
3. Course Project *(Group)*	
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.

Course Project — Project Pitches

- **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 — Project Pitches

- **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 — Project Pitches

- **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**)
  - **(1) Main idea or topic(s) of your project.**
    - Feel free to give us a little background or overview on the chosen topic (making sure to keep the full presentation to about 10 minutes long).
  - **(2) How it relates to probabilistic or generative modeling.**
  - **(3) Why you find the topic interesting (and/or how it ties in with your research).**
  - **(4) Tentative or potential goals/plans for the project that you might want to accomplish by the end of the semester.**
    - In terms of demonstration/implementation, algorithm/theory development, or other contributions.

Next Class

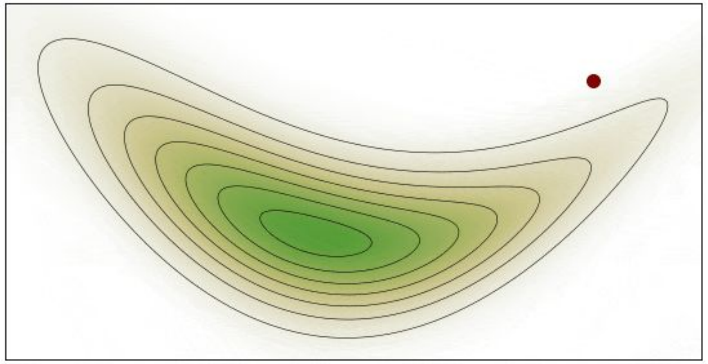
Next Class

Lecture: Approximate inference algorithms in probabilistic models, including the classics — Markov chain Monte Carlo (MCMC), and variational inference (VI).

- Markov chain Monte Carlo (MCMC) methods, including:
  - Metropolis–Hastings algorithm, Gibbs sampling, slice sampling.
  - Gradient-based methods: Langevin Monte Carlo, Hamiltonian Monte Carlo.
- Variational inference (VI) methods, including
  - Evidence lower bound (ELBO), Stochastic VI, black-box VI.

After:

- Project Pitches #1: Groups 1-5



Source: "Creating animations with MCMC", Krepl 2018, Wikimedia commons,