Probabilistic Graphical Models & Probabilistic AI

Ben Lengerich

Lecture 15: Review

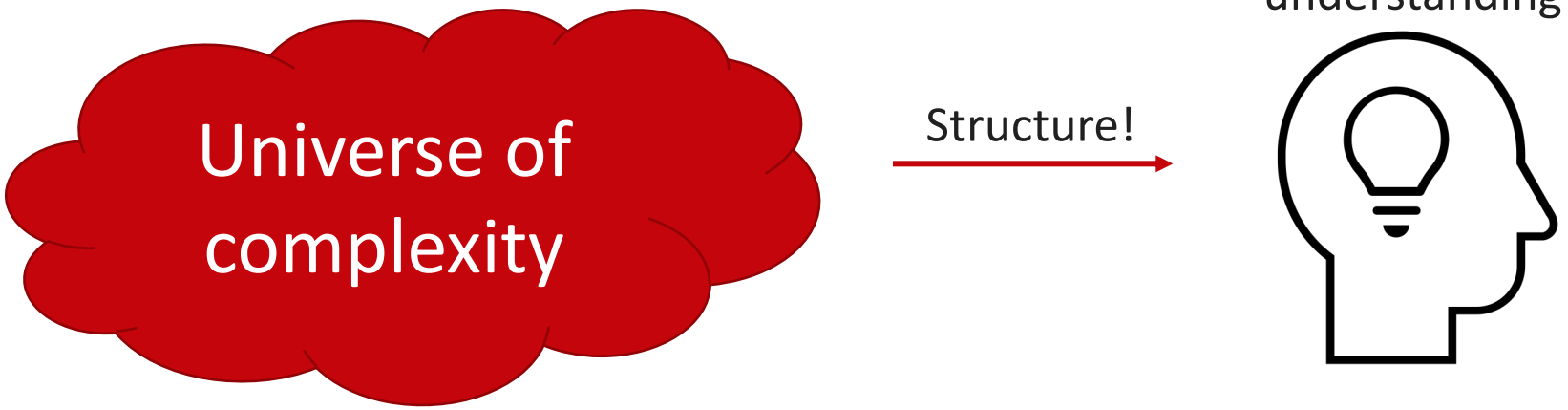
March 18, 2025

Reading: See course homepage

Why GMs?

What's the point of GMs in the AI era?

- A language for communication
- A language for computation
- A language for development



The Fundamental Questions

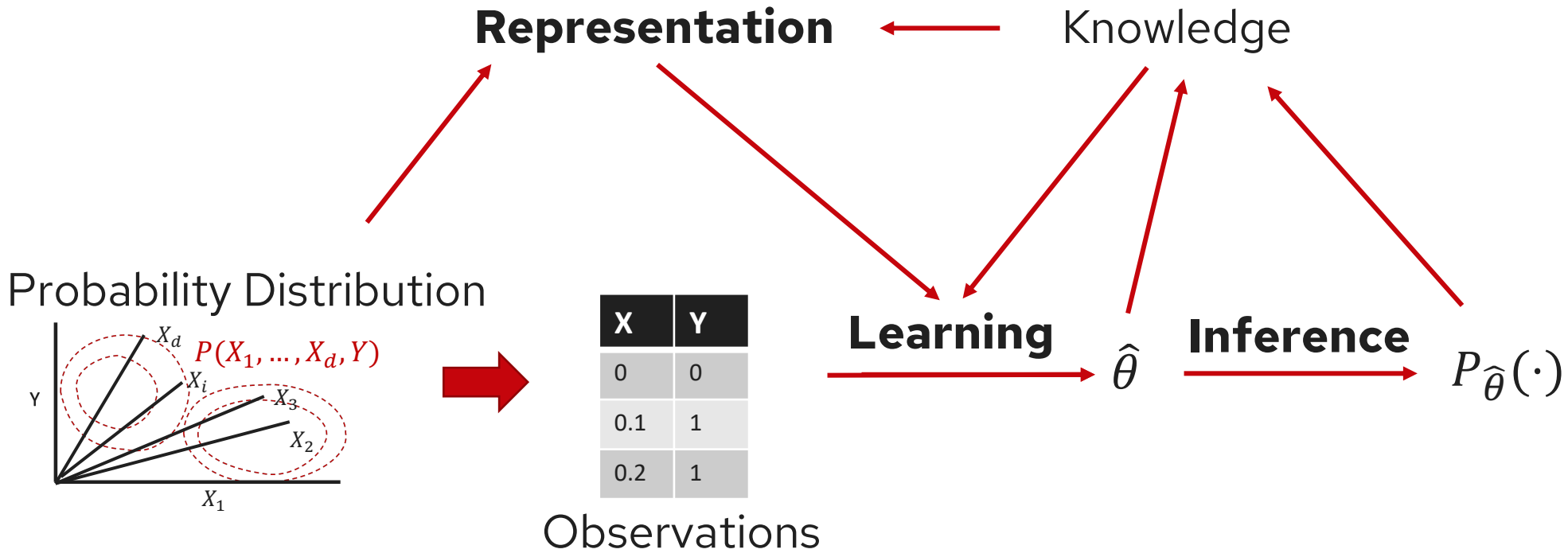
- **Representation**
  - How to encode our domain knowledge/assumptions/constraints?
  - How to capture/model uncertainties in possible worlds?
- **Inference**
  - How do I answer questions/queries according to my model and/or based on observed data?

$$\text{e.g. } P(X_i|D)$$

- **Learning**
  - What model is "right" for my data?

$$\text{e.g. } M = argmax_{M \in \mathcal{H}} F(D; M)$$

A Simplified View of our Roadmap



1. Representing GMs

PGMs allow us to understand and structure data

- GM = Multivariate Objective Function + Structure
- PGM = Multivariate Statistics + Structure

- Formally: A PGM is a **family of distributions** on a set of random variables that are compatible with all the probabilistic independence propositions encoded by a **graph** that connects these variables.

Conditional Independence

- Variables X and Y are **independent** if:

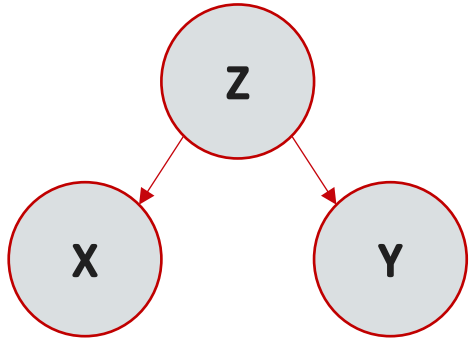
$$P(X, Y) = P(X)P(Y)$$

  - Notation: $X \perp Y$

- Variables X and Y are **conditionally independent given Z** if:

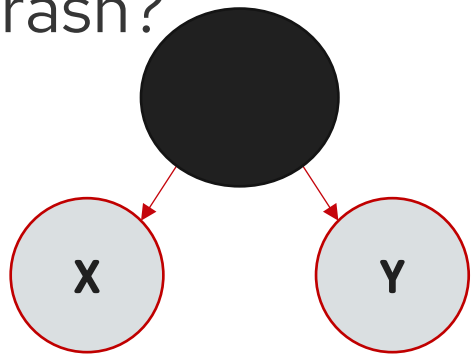
$$P(X, Y|Z) = P(X|Z)P(Y|Z)$$

  - Equivalently: $P(X|Y, Z) = P(X, Z)$
  - Notation: $X \perp Y \mid Z$



Example of Conditional Independence

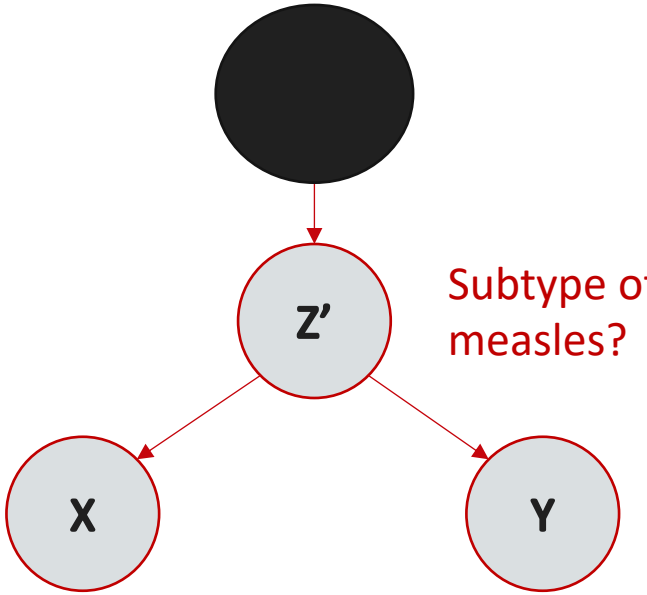
- Let X = Fever, Y = Rash, Z = Measles
- Given that a patient has measles, does knowing if they have a fever give us any additional information about whether they have a rash?



$$P(X,Y|Z) = \frac{P(X,Y,Z)}{P(Z)}$$

$$= \frac{P(X|Z)P(Y|Z)P(Z)}{P(Z)}$$

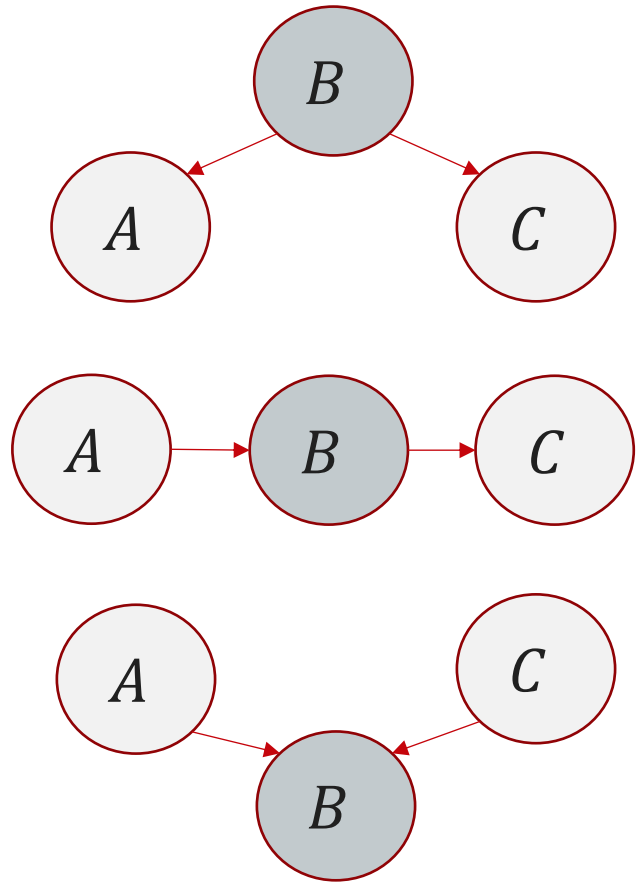
$$= P(X \mid Z)P(Y \mid Z)$$



$$P(X,Y|Z) = \sum_{Z'} P(Z' \mid Z)P(X \mid Z')P(Y \mid Z')$$

Local Structures compose graphs

- Common parent
  - Knowing B **decouples** A and C
  - $A \perp C \mid B$
- Cascade
  - Knowing B **decouples** A and C
  - $A \perp C \mid B$
- V-structure
  - Knowing B **couples** A and C
  - A can "explain away" C



Three foundational building blocks for creating complex BNs

1a. Representing PGMs: ->Directed GMs / Bayesian Networks

Bayesian Network (BN)

- A BN is a **directed acyclic graph** whose nodes represent the random variables and whose edges represent direct influence of one variable on another
- Provides the skeleton for representing a joint distribution compactly in a **factorized** way
- Compact representation of a set of **conditional independence** assumptions
- We can view the graph as encoding a **generative sampling process** executed by nature.

Bayesian Network (BN)

Factorization Theorem:

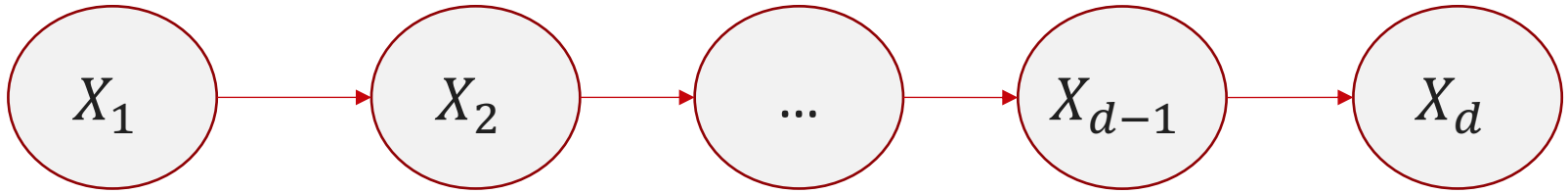
Given a DAG, the most general form of the probability distribution that is consistent with the graph factors according to:

$$P(X) = \prod_i P\left(X_i \mid X_{\pi_i} \right)$$

where X_{π_i} is the set of parents of X_i.

A simple BN: Markov Chain

- Markov Chain
- **Markov property:** "The future state depends only on the present state, and not on past states"
- Parameters:
  - Transition Probability Matrix: $M_{ij} = P(X_t = j \mid X_{t-1} = i)$
  - Initial State Distribution: $\pi_i = P(X_1 = i)$

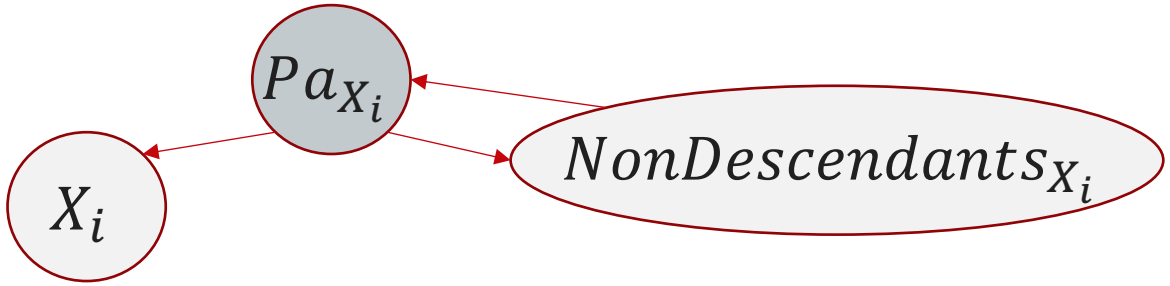


$$\mathrm{P(X)} = P(X_1) \prod_{t=2} P(X_t \mid X_{t-1})$$

Independence assertions of BNs

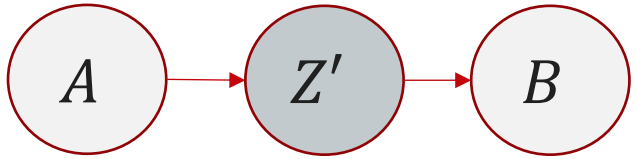
- In a BN, each node is independent of its non-descendants given its parents.
- Let Pa_{X_i} denote the parents of X_i in G and $NonDescendants_{X_i}$ denote the variables in the graph that are not descendants of X_i. Then G encodes the following set of *local conditional independence assumptions* $I_l(G)$:

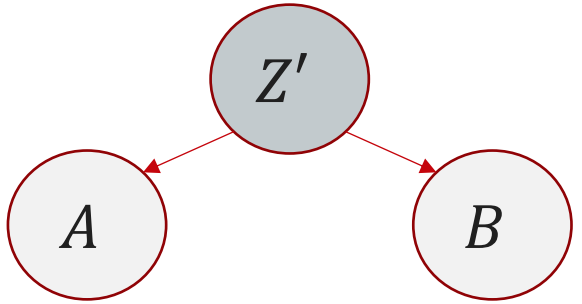
$$I_l(G) = \{X_i \perp NonDescendants_{X_i} | Pa_{X_i} : \quad \forall\, i\}$$

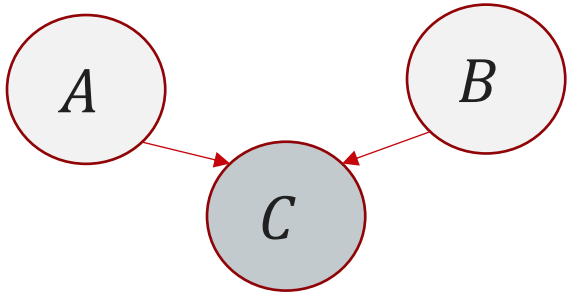


Independence assertions: Graph separation

- D-separation criterion for Bayesian networks [Pearl, 1988]
  - D for "directed" edges
  - **Definition:** A set of nodes X is d-separated (conditionally independent) from a set of nodes Y given a conditioning set Z iff every path between any nodes in X and any node in Y is **blocked** by Z.
    - A path between nodes A and B is **blocked** by Z if it contains at least one of the following structures:
      - Chain: $A \rightarrow Z' \rightarrow B$ for $Z' \in Z$
      - Fork: $A \leftarrow Z' \rightarrow B$ for $Z' \in Z$
      - Collider: $A \rightarrow C \leftarrow B$ for $C \notin Z$ AND no descendant of C is in Z

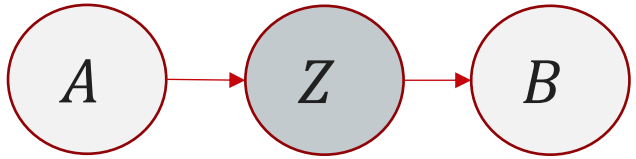


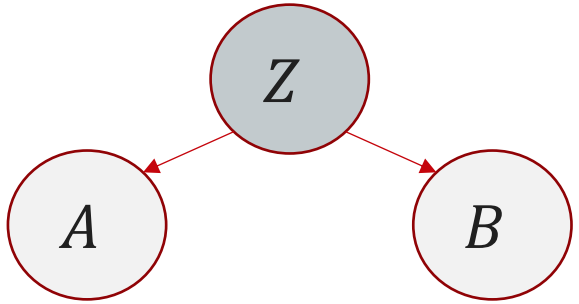


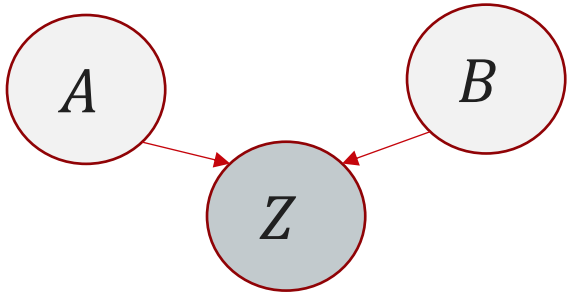


Non-Independence: Active Trails

- Causal: $A \rightarrow Z \rightarrow B$
  - Active iff Z is not observed.
- Common Cause: $A \leftarrow Z \rightarrow B$
  - Active iff Z is not observed.
- Collider: $A \rightarrow Z \leftarrow B$
  - Active iff Z OR one of Z's descendants is observed.

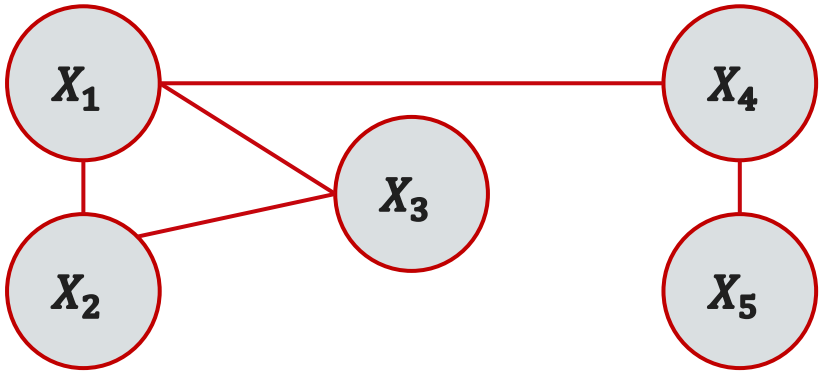






1b. Representing PGMs: Undirected GMs / Markov Random Fields

Undirected Graphical Models



- Pairwise relationships
- No explicit way to generate samples
- Contingency constraints on node configurations

Representing Undirected Graphical Models

- An ***undirected graphical model*** represents a distribution $P(X)$ defined by an undirected graph H and a set of positive ***potential functions*** ψ associated with the cliques of H such that:

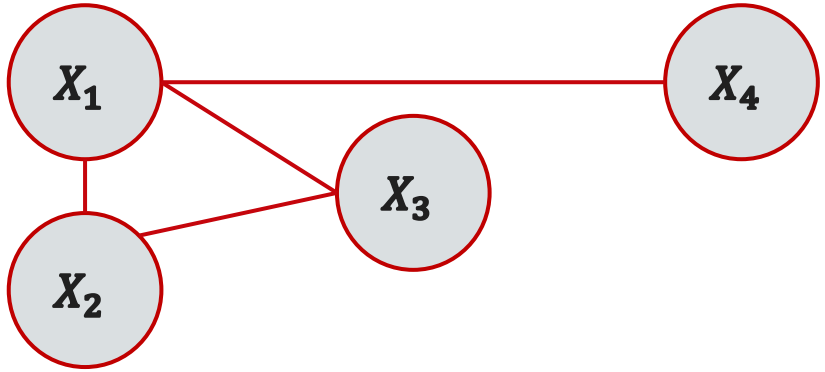
$$\mathrm{P}(\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{n}) = \frac{1}{\mathrm{Z}} \prod_c \psi_c(X_C)$$

where Z represents the **partition function**: $Z = \sum_X \prod_c \psi_c(X_c)$.

- The potential function can be understood as a "score" of the joint configuration

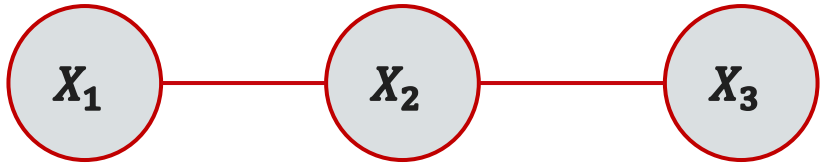
What is a clique?

- For $G = \{V, E\}$, a clique (complete subgraph) is a subgraph $G' = \{V' \subseteq V, E' \subseteq E\}$ such that nodes in V' are **fully connected**.
- A **maximal** clique is a clique such that any superset $V'' \supset V$ is *not* a clique.



Maximal cliques: $\{X_1, X_2, X_3\}, \{X_1, X_4\}$
Sub-cliques: $\{X_1, X_2\}, \{X_2, X_3\}, \{X_1, X_3\}, \{X_1\}, \{X_2\}, \{X_3\}, \{X_4\}$

Interpretation of Clique Potentials



- This model implies $X_1 \perp X_3 | X_2$, so joint must factorize as:
$$P(X_1, X_2, X_3) = P(X_2)P(X_1|X_2)P(X_3|X_2)$$
- We could write as $P(X_1, X_2)P(X_3 \mid X_2)$ or $P(X_2, X_3)P(X_1 \mid X_2)$, but:
  - Cannot have all potentials be marginals
  - Cannot have all potential be conditionals
- Clique potentials can be thought of as general "compatibility" of their variables, but not as probability distributions.

Features

- A “feature” is a function that is non-zero for a few particular inputs and zero otherwise.
- Key idea: Instead of modeling all possible feature values in a big table, model specific groupings of feature values together.
- Example:
  - Let a clique correspond to three consecutive characters.
  - How would we define p(c1, c2, c3)?
    - All possible character combinations we need 26^3 – 1 parameters.
    - But there are sequences that are unlikely: kfd
    - Define a feature like “ing”: 1 if c1=i,c2=n,c3=g. 0 otherwise.

Features as Potentials

- Each feature function can be converted to a potential by exponentiating it. We can multiply these together to get a clique potential.
- Example:

$$\psi_c(c_1,c_2,c_3) = e^{\theta_{\text{ing}} f_{\text{ing}}} \times e^{\theta_{?\text{ed}} f_{?\text{ed}}} \times \ldots$$

$$= \exp\left\{\sum_{k=1}^{K} \theta_k f_k(c_1,c_2,c_3)\right\}$$

- There is still an exponential number of settings, but only K parameters (θ_k)
- A nice benefit of undirected graphical models: we don't have to normalize each feature.

Combining Features

- Each feature function has a weight θ_k which represents the numerical strength of the feature and whether it increases or decreases the probability of a clique.
- The marginal over the clique is a generalized exponential family distribution (a GLM):

$$p(c_1,c_2,c_3) \propto \exp\left\{\begin{array}{l}\theta_{\text{ing}} f_{\text{ing}}(c_1,c_2,c_3)+\theta_{?\text{ed}} f_{?\text{ed}}(c_1,c_2,c_3)+ \\ \theta_{\text{qu?}} f_{\text{qu?}}(c_1,c_2,c_3)+\theta_{\text{zzz}} f_{\text{zzz}}(c_1,c_2,c_3)+\cdots\end{array}\right\}$$

- The features may be overlapping across cliques

$$\psi_c(\mathbf{x}_c) \overset{\text{def}}{=} \exp\left\{\sum_{i\in I_c} \theta_k f_k(\mathbf{x}_{c_i})\right\}$$

Feature-based model

- Joint distribution: $$p(\mathbf{x}) = \frac{1}{Z(\theta)} \prod_c \psi_c(\mathbf{x}_c) = \frac{1}{Z(\theta)} \exp\left\{ \sum_c \sum_{i \in I_c} \theta_k f_k(\mathbf{x}_{c_i}) \right\}$$
- We can drop sum over c: $$p(\mathbf{x}) = \frac{1}{Z(\theta)} \exp\left\{ \sum_i \theta_i f_i(\mathbf{x}_{c_i}) \right\}$$

- What are the sufficient statistics for this model?
- The features

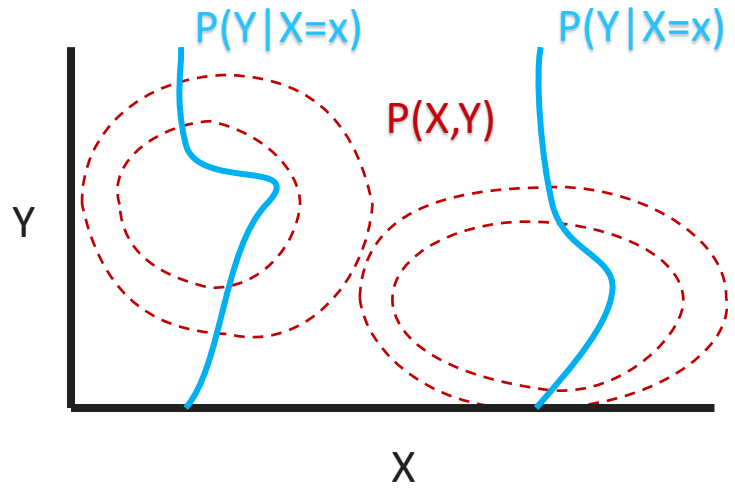
- We need to learn weighting parameters θ_k

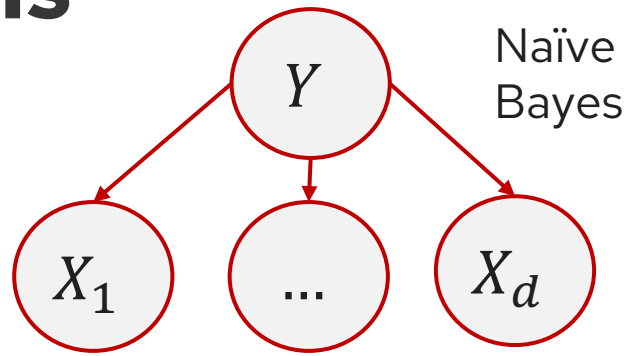
1c. Implications of GM Structure

Generative and Discriminative Models

- **Generative**:
  - Models the joint distribution $\mathrm{P}(X, Y)$.

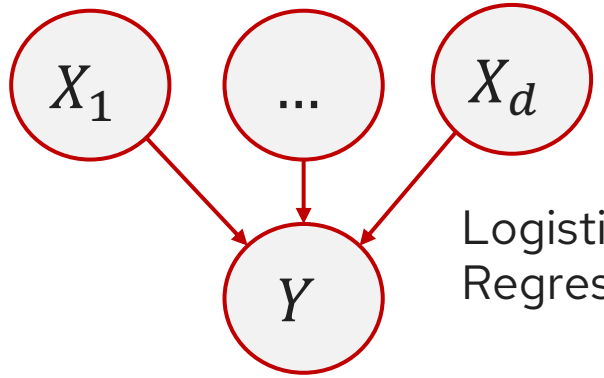
- **Discriminative**:
  - Models the conditional distribution $\mathrm{P}(Y|X)$.





Naïve Bayes

$$P(X,Y) = P(Y)\prod_i P(X_i \mid Y)$$

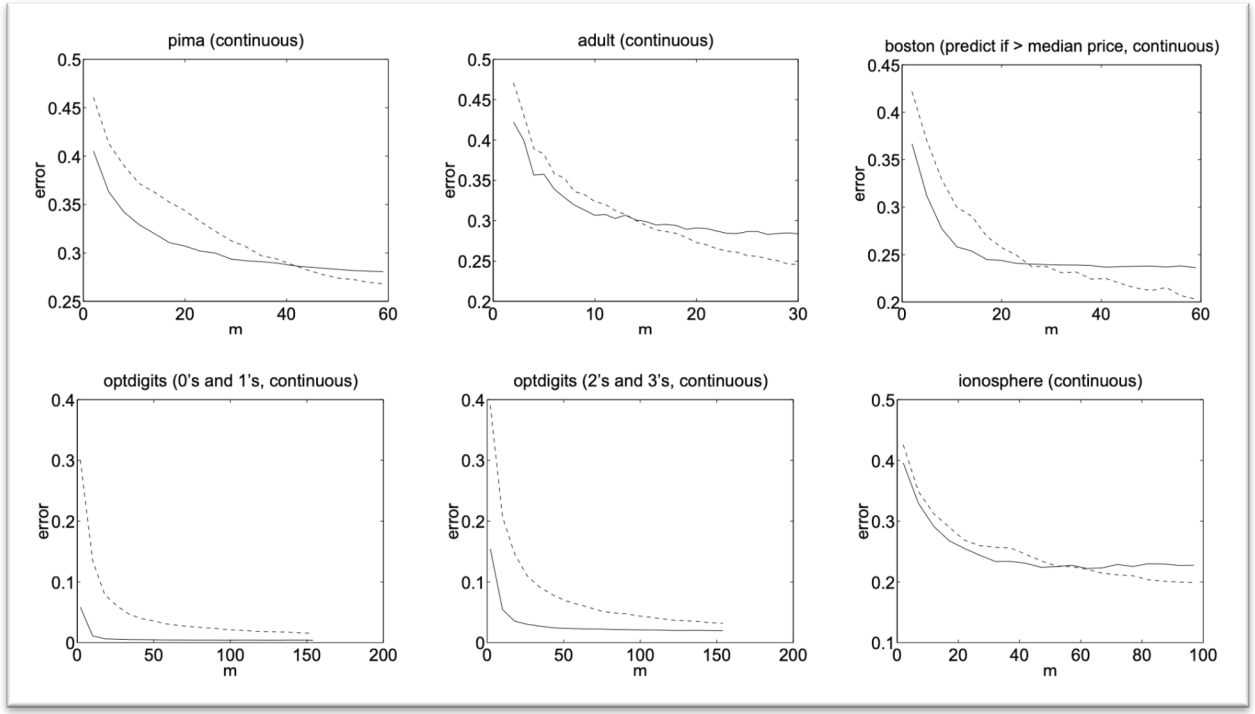


Logistic Regression

$$P(X,Y) = P(Y \mid X)\prod_i P(X_i)$$

Andrew Ng's Insight

- "While discriminative learning has lower asymptotic error, a generative classifier may also approach its (higher) asymptotic error much faster."



Ng & Jordan 2001

Andrew Ng's Insight

- "While discriminative learning has lower asymptotic error, a generative classifier may also approach its (higher) asymptotic error much faster."
- Underlying assumption of this statement:
  - Generative models of the form $\boldsymbol{P(X, Y, \theta)}$ make **more simplifying assumptions** than do discriminative models of the form $\boldsymbol{P(Y|X, \theta)}$**.**
  - **Not always true**
  - "So far there is no theoretically correct, general criterion for choosing between the discriminative and the generative approaches to classification of an observation **x** into a class *y*; the choice depends on the relative confidence we have in the correctness of the specification of either $p(y|\mathbf{x})$ or $p(\mathbf{x}, y)$ for the data."

Xue & Tittering 2008

Modern Deep Generative Models

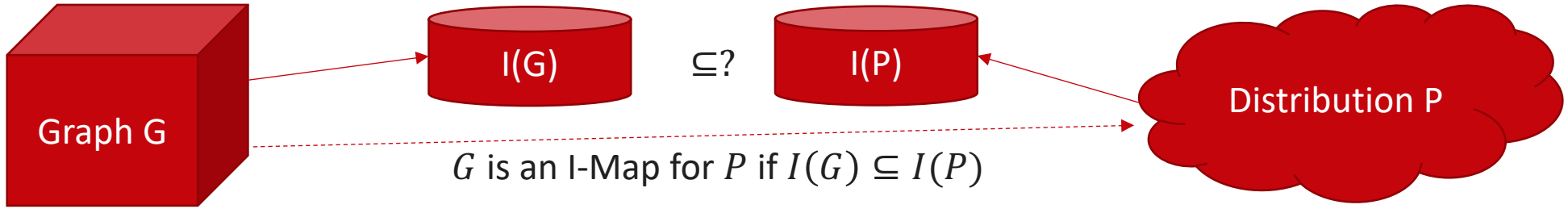
Can we build massive-scale deep generative models that no longer require the constraints of simplifying assumptions?

January 27, 2025:

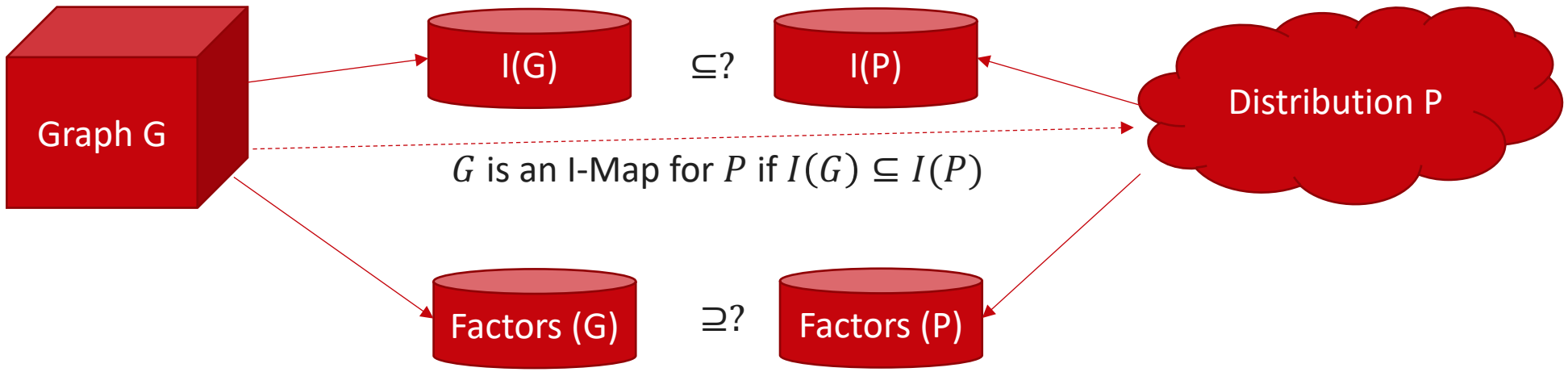
1d. I-Maps of PGMs

I-Maps

- Independence set: Let P be a distribution over X. We define $I(P)$ to be the set of independences $(X \perp Y \mid Z)$ that hold in P.
- I-Map: Let G be any graph object with an associated independence set $I(G)$. We say that G is an **I-map** for an independence set I if $I(G) \subseteq I$.
- I-Map of Distribution: We say G is an I-map for P if G is an I-map for $I(P)$, when we use $I(G)$ as the associated independence set.

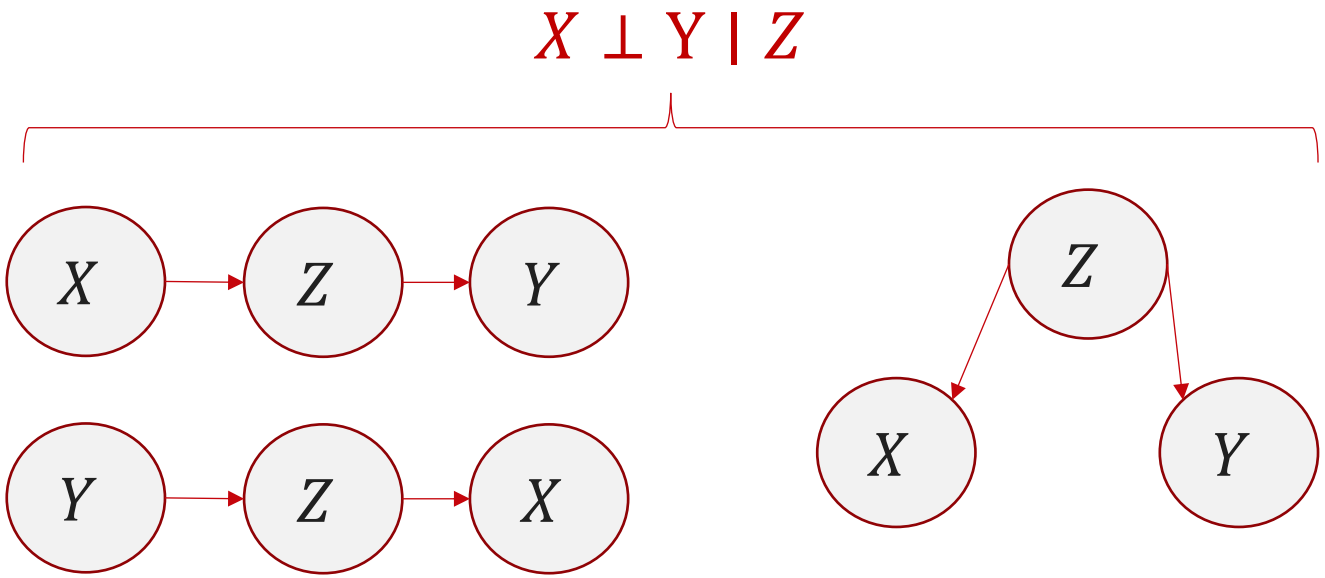


I-Maps and Factorization



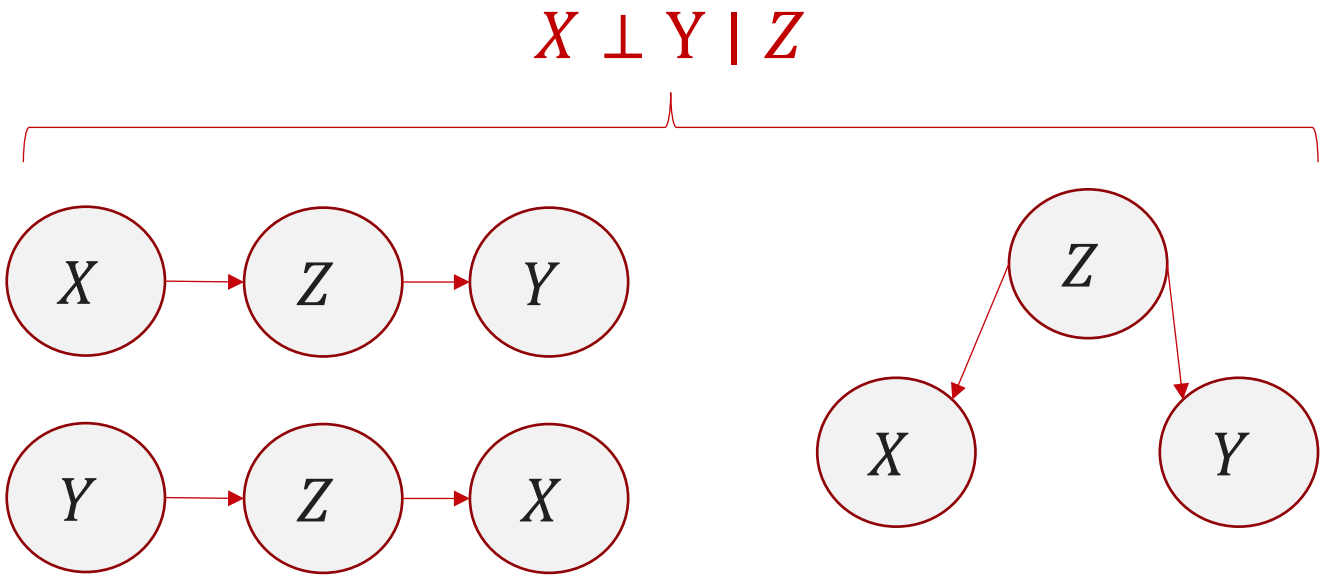
I-equivalence

- Definition of I-Equivalence: Two BN graphs G_1 and G_2 over X are *I-equivalent* if $I(G_1) = I(G_2)$.



Uniqueness of Graphs

- Very different graphs can be equivalent in that they encode the same set of conditional independence assertions.



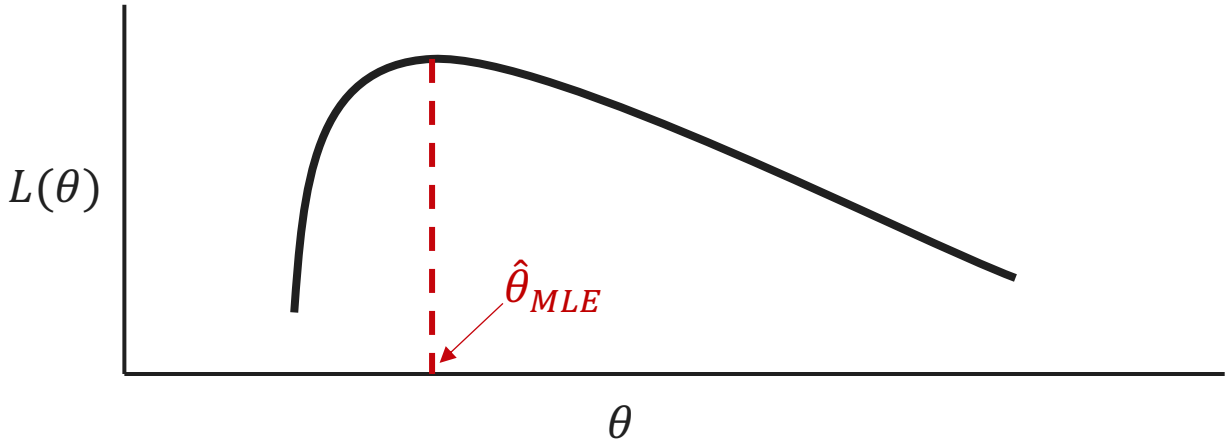
2. Learning GMs

Maximum Likelihood Estimation (MLE)

- **Definition:**
  - Find $\widehat{\boldsymbol{\theta}}$ that maximizes the likelihood of observing the given data.

$$\widehat{\boldsymbol{\theta}} = \text{argmax}_{\theta} L(\theta) \text{ where } L(\theta) = P(\text{data}|\theta).$$

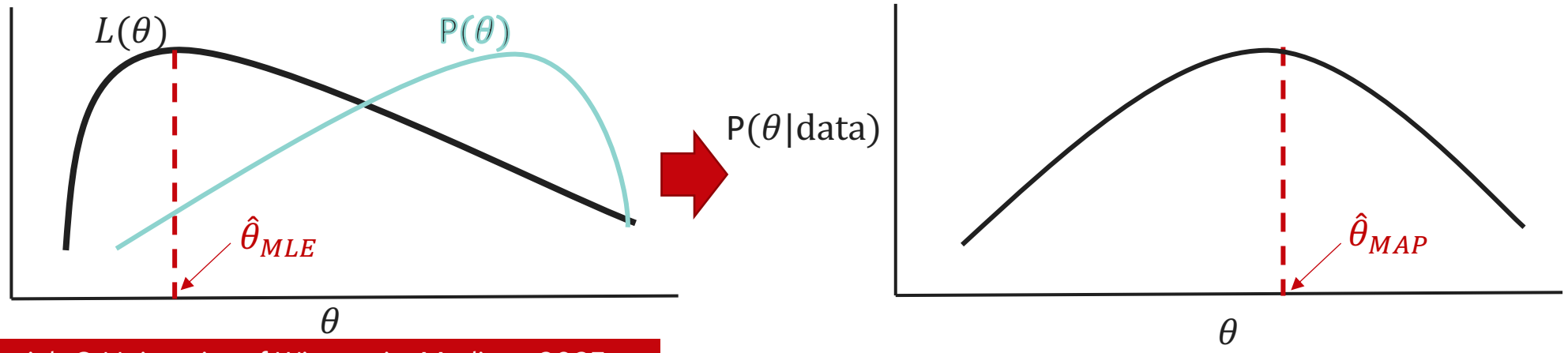
- **Interpretation:**
  - L(θ): Probability of the observed data given θ.
  - MLE chooses the parameter that makes the data most "likely."



Maximum A Posteriori (MAP) Estimation

- Find
$$\hat{\theta}_{MAP} = argmax_{\theta} P(\theta|\text{data}) \propto argmax_{\theta} P(\text{data}|\theta)P(\theta)$$
- $P(\text{data}|\theta)$: Likelihood
- $P(\theta)$: Prior belief about θ

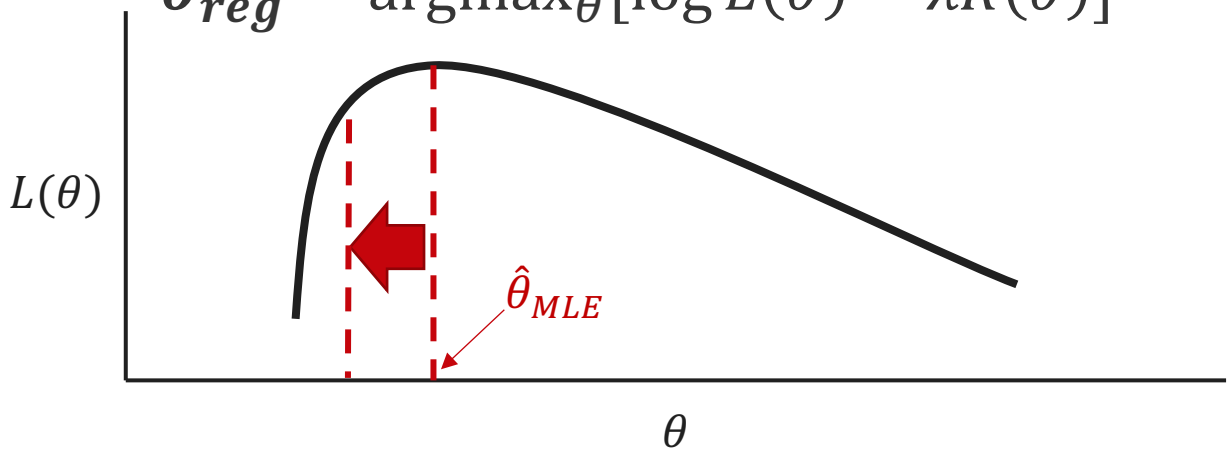
- MLE ignores $P(\theta)$
- MAP incorporates prior information.



Regularization is MAP

- **MLE with Regularization:**
  - Adds a penalty to avoid overfitting

$$\widehat{\boldsymbol{\theta}_{\boldsymbol{reg}}} = \text{argmax}_\theta[\log L(\theta) - \lambda R(\theta)]$$



- **MAP as Penalized MLE:**
  - Let $P(\theta) \propto e^{-\lambda R(\theta)}$. Then

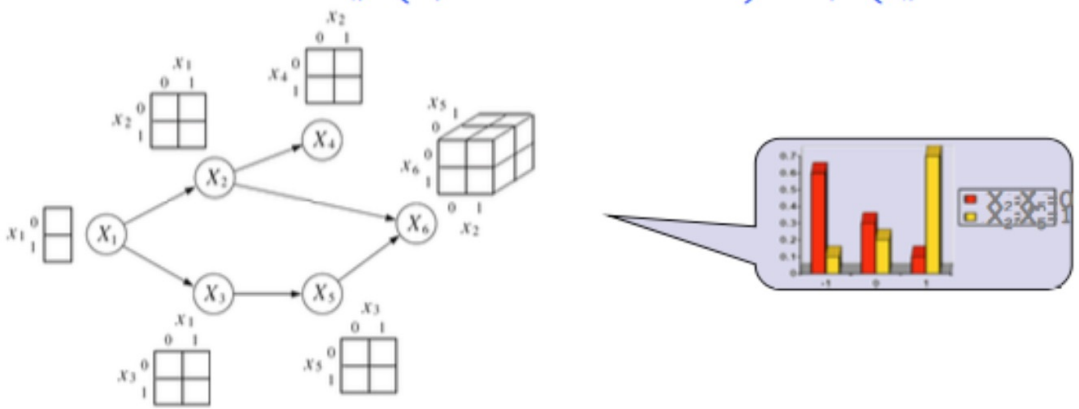
$$\widehat{\boldsymbol{\theta}}_{MAP} = argmax_\theta[\log L(\theta) + \log P(\theta)] = \hat{\theta}_{reg}$$

2a. Parameter Learning in Fully-Observed BNs

MLE for general BNs

- If we assume the parameters for each CPD are globally independent, and all nodes are fully observed, then the log-likelihood function decomposes into a sum of local terms, one per node

$$\ell(\theta; D) = \log p(D \mid \theta) = \log \prod_n \left(\prod_i p(x_{n,i} \mid \mathbf{x}_{n,\pi_i}, \theta_i) \right) = \sum_i \left(\sum_n \log p(x_{n,i} \mid \mathbf{x}_{n,\pi_i}, \theta_i) \right)$$

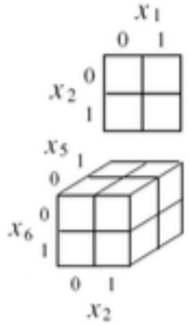


- MLE-based parameter estimation of GM reduces to local est. of each GLIM.

MLE for BNs with tabular CPDs

- Each CPD is represented as a table (multinomial) with

$$\theta_{ijk} \overset{\text{def}}{=} p(X_i = j \mid X_{\pi_i} = k)$$



  - In case of multiple parents the CPD is a high-dimensional table
  - The sufficient statistics are counts of variable configurations

$$n_{ijk} \overset{\text{def}}{=} \sum_n x_{n,i}^j x_{n,\pi_i}^k$$

- The log-likelihood is $\ell(\theta; D) = \log \prod_{i,j,k} \theta_{ijk}^{n_{ijk}} = \sum_{i,j,k} n_{ijk} \log \theta_{ijk}$

- And using a Lagrange multiplier to enforce that conditionals sum up to 1 we have:

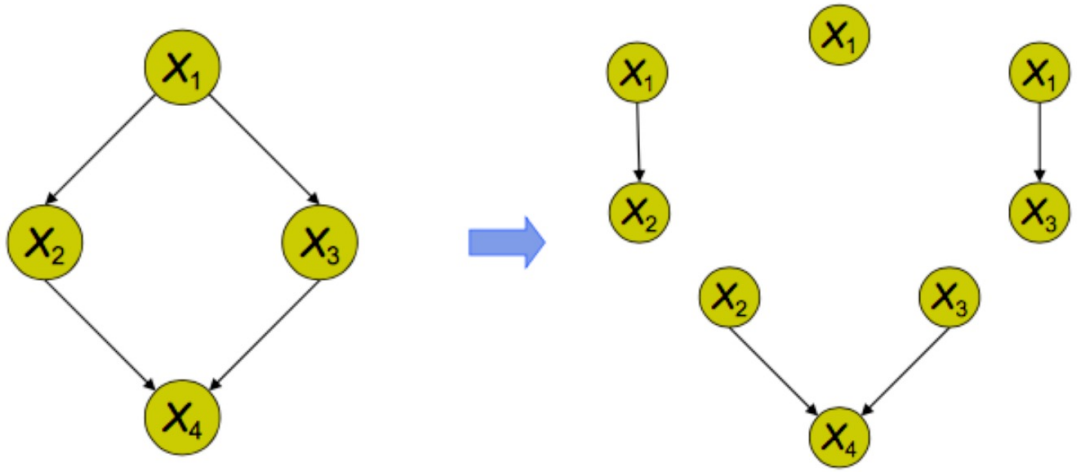
$$\theta_{ijk}^{ML} = \frac{n_{ijk}}{\sum_{j'} n_{ij'k}}$$

Decomposable likelihood of a BN

- Consider the GM:

$$p(x \mid \theta) = p(x_1 \mid \theta_1)\, p(x_2 \mid x_1, \theta_2)\, p(x_3 \mid x_1, \theta_3)\, p(x_4 \mid x_2, x_3, \theta_4)$$

- This is the same as learning four separate smaller BNs each of which consists of a node an its parents.



2b. Parameter Learning in Fully-Observed Undirected GMs

MLE for Undirected GMs

- For undirected models, the log-likelihood does not decompose, because the normalization constant Z is a function of **all** parameters.

$$P(x_1,\ldots,x_n) = \frac{1}{Z}\prod_{c\in C}\psi_c(\mathbf{x}_c) \qquad Z = \sum_{x_1,\ldots,x_n}\prod_{c\in C}\psi_c(\mathbf{x}_c)$$

- The likelihood decomposes to give MLE conditions on clique probabilities:

$$p^*_{MLE}(\mathbf{x}_c) = \frac{m(\mathbf{x}_c)}{N}$$

- But UGMs are parameterized by ψ_c not p.
- In general, we need to do inference to learn parameters for undirected models, even in the fully observed case.

Case 1: The model is decomposable

- If the model is **decomposable** and **all the clique potentials are defined on maximal cliques**, then:
  - The MLE of clique potentials are equal to the empirical marginals (or conditionals) of the corresponding clique.
- Example: Chain $X_1 - X_2 - X_3$

$$p_{MLE}(X1, X2, X3) = \frac{\tilde{p}(X1, X2)\tilde{p}(X2, X3)}{\tilde{p}(X2)}$$

$$p_{MLE}(X1, X2) = \sum_{X3} \tilde{p}(X1, X2, X3) = \tilde{p}(X1|X2) \sum_{X3} \tilde{p}(X2, X3) = \tilde{p}(X1, X2)$$

$$p_{MLE}(X2, X3) = \tilde{p}(X2, X3)$$

$$\hat{\psi}_{12}^{MLE}(x_1, x_2) = \tilde{p}(x_1, x_2) \qquad \hat{\psi}_{23}^{MLE}(x_2, x_3) = \frac{\tilde{p}(x_2, x_3)}{\tilde{p}(x_2)} = \tilde{p}(x_3|x_2)$$

Case 2: The model is NON-decomposable

- If the model is **non-decomposable** (clique potentials are defined on non-maximal cliques), then we cannot equate MLE of clique potentials to empirical marginals (or conditionals).
- **Iterative Proportional Fitting (IPF)**:

$$\psi_c^{t+1}(x_c) = \psi_c^t(x_c)\frac{\tilde{p}(x_c)}{p^t(x_c)}$$

Empirical distribution

Current model distribution

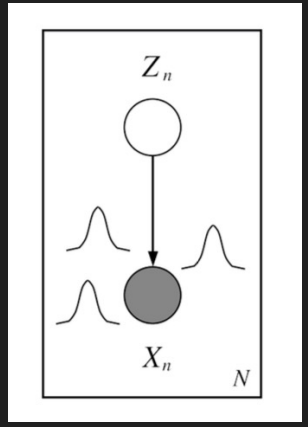
- **Generalized Iterative Scaling (GIS)**:

$$\theta_i^{t+1} = \theta_i^t + \log\frac{E_{data}[f_i(x)]}{E_{p(x;\theta^t)}[f_{i(x)}]}$$

Empirical expectation

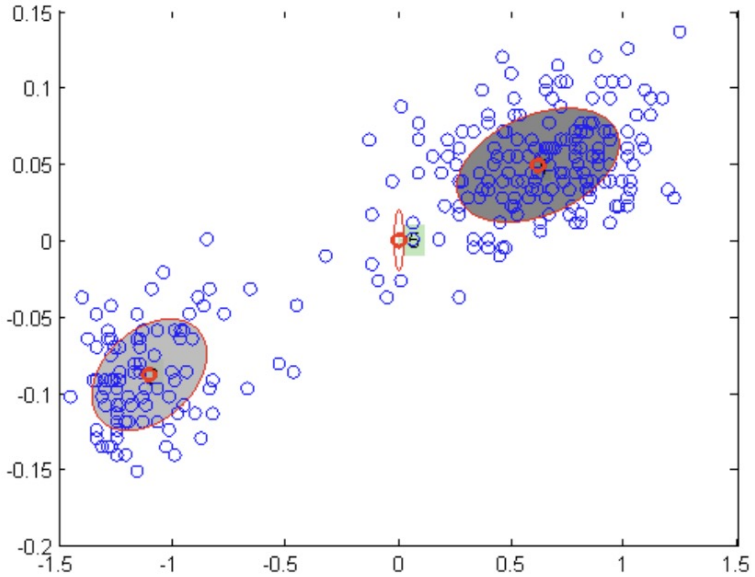
Expectation under current model distribution

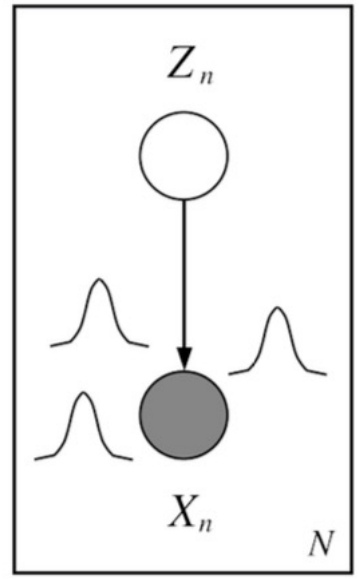
2c. Parameter Learning in Partially-Observed GMs



Partially-Observed GMs: Mixture models

- A density model $p(x)$ may be multi-modal
- Can we model it as a mixture of uni-modal distributions?





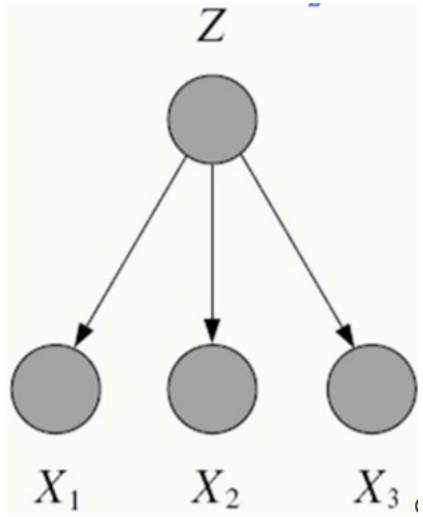
Unobserved Variables

- A variable can be unobserved (latent) because:
  - It is difficult or impossible to measure
    - e.g. Causes of a disease, evolutionary ancestors
  - It is only sometimes measured
    - e.g. faulty sensors
  - It is an imaginary quantity meant to provide some simplified but useful view of the data generation process
    - e.g. Mixture assignments
- Discrete latent variables can be used for as cluster assignments
- Continuous latent variables can be used for dimensionality reduction

Why is learning with latent variables harder?

- In fully-observed IID settings, the log-likelihood decomposes into a sum of local terms:

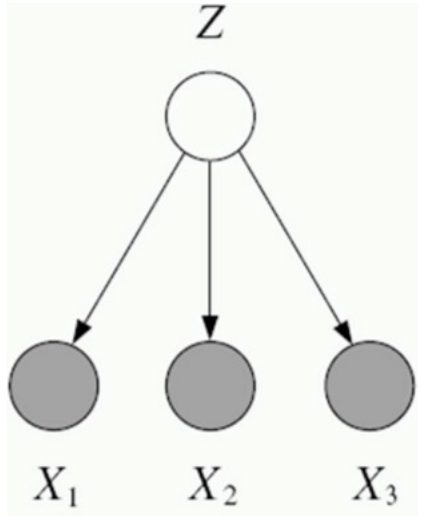
$$\ell_c(\theta; D) = \log p(x, z \mid \theta) = \log p(z \mid \theta_z) + \log p(x \mid z, \theta_x)$$



- With latent variables, all parameters become coupled via marginalization

$$\ell_c(\theta; D) = \log \sum_z p(x, z \mid \theta) = \log \sum_z p(z \mid \theta_z) p(x \mid z, \theta_x)$$

Sum over z is inside log



Strategy:

1. Guess value of Z
2. Apply MLE to estimate best model parameters based on Z
3. Inference most likely Z based on MLE parameter estimates
4. Return to step 2 until Z stops changing

Expectation-Maximization algorithm

- **E-step**:
  - Compute the expected value of the sufficient statistics of the hidden variables under current estimates of parameters

- **M-step**:
  - Using the current expected value of the hidden variables, compute the parameters that maximize the likelihood.

Why does EM work?

$$p(z|x,\theta) = \frac{p(x,z \mid \theta)}{p(x \mid \theta)}$$

$$\log p(z|x,\theta) = \log p(x,z \mid \theta) - \log p(x \mid \theta)$$

$$\mathrm{E}_{z\sim q}[\log p(z|x,\theta)] = E_{z\sim q}[\log p(x,z \mid \theta)] - \log p(x \mid \theta)$$

$$KL(q(z \mid x) \,||\, p(z \mid x,\theta)) = E_{z\sim q}\left[\log \frac{q(z \mid x)}{p(z \mid x,\theta)}\right]$$

$$E_{z\sim q}[\log p(z \mid x,\theta)] = E_{z\sim q}[\log q(z \mid x)] - KL(q(z \mid x) \,||\, p(z \mid x,\theta))$$

$$E_{z\sim q}[\log p(x,z \mid \theta)] - \log p(x \mid \theta) = E_{z\sim q}[\log q(z \mid x)] - KL(q(z \mid x) \,||\, p(z \mid x,\theta))$$

$$\log p(x \mid \theta) = E_{z\sim q}[\log p(x,z \mid \theta)] - E_{z\sim q}[\log q(z \mid x)] + KL(q(z \mid x) \,||\, p(z \mid x,\theta))$$

$$\log p(x \mid \theta) = E_{z\sim q}[\log p(x,z \mid \theta)] + H(q) + KL(q(z \mid x) \,||\, p(z \mid x,\theta))$$

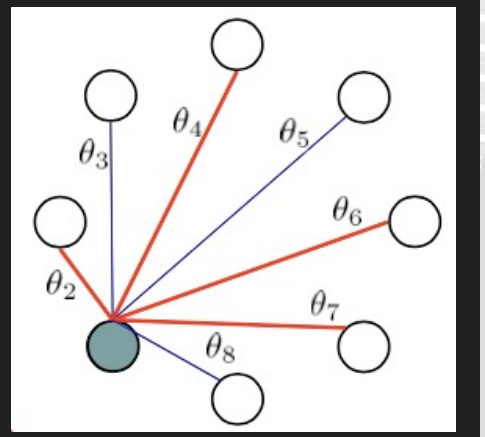
EM: Let $q_t(z \mid x) = p(z \mid x,\theta_t)$. Then at convergence:

$$\log p(x \mid \theta) = E_{z\sim q_t}[\log p(x,z \mid \theta)] + H(q_t) + 0$$

$$Q(\theta',\theta_t) = E_{z\sim p(z|\theta_t)}[\log p(x,z \mid \theta')]$$

$$\theta_{t+1} = \mathrm{argmax}_{\theta'} Q(\theta',\theta_t)$$

2d. Structure Learning



Max Likelihood doesn't inform us about structure

$$\ell(\theta_G, G; D) = \log p(D \mid \theta_G, G)$$
$$= M \sum_i \hat{I}(x_i, \mathbf{x}_{\pi_i(G)}) - M \sum_i \hat{H}(x_i)$$

Mutual information between x_i and its parents

Entropy of x_i

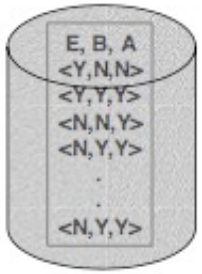
- As we match x_i and parents better, the mutual information increases.
- Problems?
- Adding edges always helps!

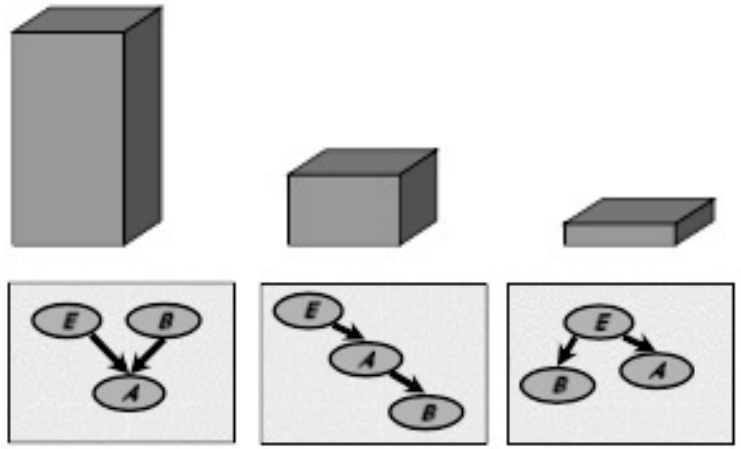
Different approaches to structure learning

- Two main problems:
  - **Likelihood** is **maximized for fully-connected graph**, so we don't want to just maximize likelihood alone.
  - Finding optimal BN structure is an **NP-hard** problem if allowed to be non-tree.
- Many heuristics but no "guarantees" of returning the perfect structure.
- Can get some guarantees if we make assumptions:
  - For tree BNs: Chow-Liu algorithm
  - For pairwise MRFs: Covariance selection, neighborhood-selection

Score-based Learning

- Define a scoring function that evaluates how well a structure matches the data:

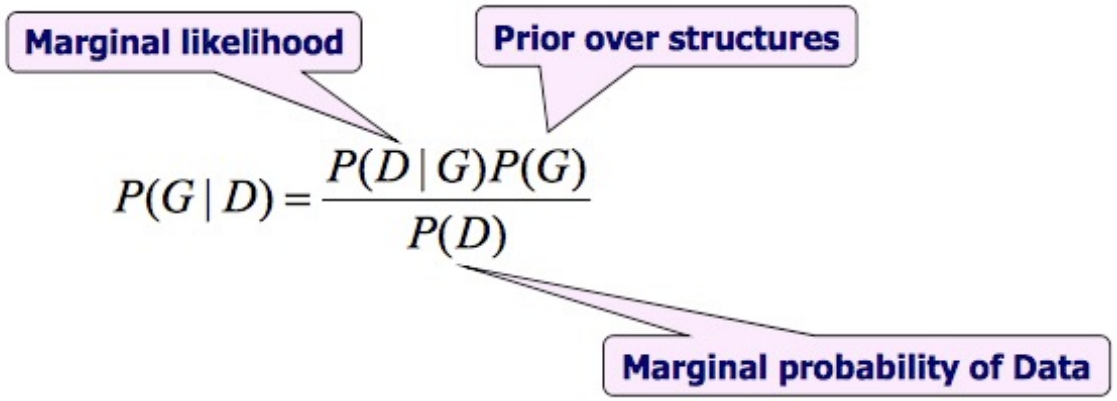




- Search for a structure that maximizes the score

Bayesian Score

- Let's take a Bayesian approach
  - Place a distribution over our "uncertain" elements (G and θ)



- Bayesian score for G

$$Score_B(G:D) = \log P(D \mid G) + \log P(G)$$

Bayesian Score cont'd

- Bayesian score for G

$$Score_B(G:D) = \log P(D \mid G) + \log P(G)$$

- Our choice of prior $P(G)$ has implications.
- Example: Let the edges have Dirichlet priors. Then as the number of configurations $M \to \infty$,

$$\log P(D \mid G) = l(\hat{\theta}_G : D) - \frac{\log M}{2} Dim(G) + O(1)$$

Dim(G): number of independent parameters in G

Tradeoff between fit to vs. data and complexity

Bayesian Information Criterion (BIC)

- Bayesian score gives Bayesian Information Criterion:

$$Score_{BIC}(G:D) = l(\hat{\theta}_G : D) - \frac{\log M}{2} Dim(G)$$

Chow-Liu tree learning algorithm

- Assume true structure is a tree
- Start by calculating Mutual Information between every pair of variables X_i and X_j

$$\hat{p}(X_i, X_j) = \frac{count(x_i, x_j)}{M}$$

$$\hat{I}(X_i, X_j) = \sum_{x_i, x_j} \hat{p}(x_i, x_j) \log \frac{\hat{p}(x_i, x_j)}{\hat{p}(x_i)\hat{p}(x_j)}$$

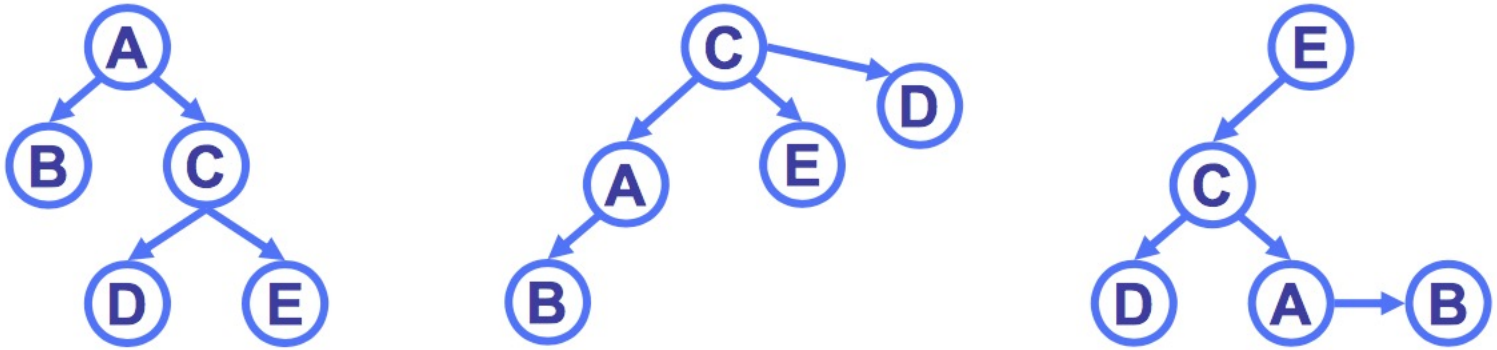
- Compute maximum weight spanning tree (Kruskal)
- Guarantees to maximize objective function:

$$\ell(\theta_G, G; D) = \log \hat{p}(D \mid \theta_G, G)$$
$$= M \sum_i \hat{I}(x_i, \mathbf{x}_{\pi_i(G)}) - M \sum_i \hat{H}(x_i)$$

$$\Rightarrow \quad \boxed{C(G) = M \sum_i \hat{I}(x_i, \mathbf{x}_{\pi_i(G)})}$$

Chow-Liu tree learning algorithm: directionality

- How to pick direction of edges?
- Pick any node as root, do BFS to define directions



$$C(G) = I(A,B) + I(A,C) + I(C,D) + I(C,E)$$

- Can't tell the difference between competing root nodes

Pairwise MRFs = Gaussian Graphical Model

- Pairwise MRF:

$$P(X) \propto \prod_i \psi_i(X_i) \prod_{i,j} \psi_{i,j}(X_i, X_j)$$

- Gaussian Graphical Model:
  - Let $\psi_i(X_i) = \exp(\theta_i X_i)$, $\psi_{i,j}(X_i, X_j) = \exp(\theta_{ij} X_i X_j)$
  - Then:

$$\mathrm{P}(\mathrm{X} \mid \theta) \propto \exp\left(\sum_i \theta_i X_i + \sum_{i,j} \theta_{ij} X_i X_j\right)$$

Gaussian Graphical Model

- Gaussian Graphical Model:
  - Let $\psi_i(X_i) = \exp(\theta_i X_i)$, $\psi_{i,j}(X_i, X_j) = \exp(\theta_{ij} X_i X_j)$
  - Then:

$$\mathrm{P}(\,\mathrm{X} \mid \theta\,) \propto \exp\left(\sum_i \theta_i X_i + \sum_{i,j} \theta_{ij}\, X_i X_j\right)$$

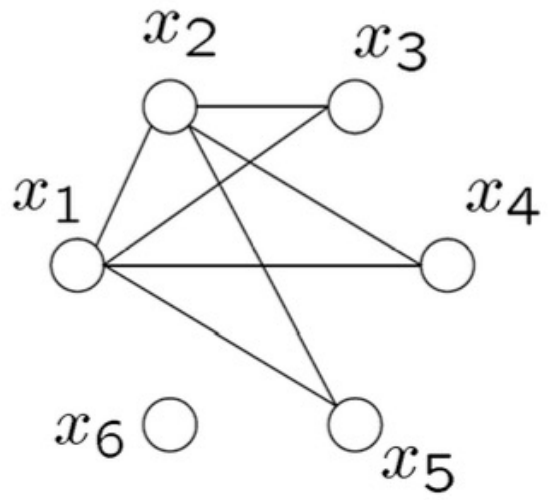
- This is a Multivariate Gaussian density:

$$p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)]$$

- for $\mu = 0$ and $\theta = \Sigma^{-1} = Q$.

So estimating Precision Mat. gives MRF structure

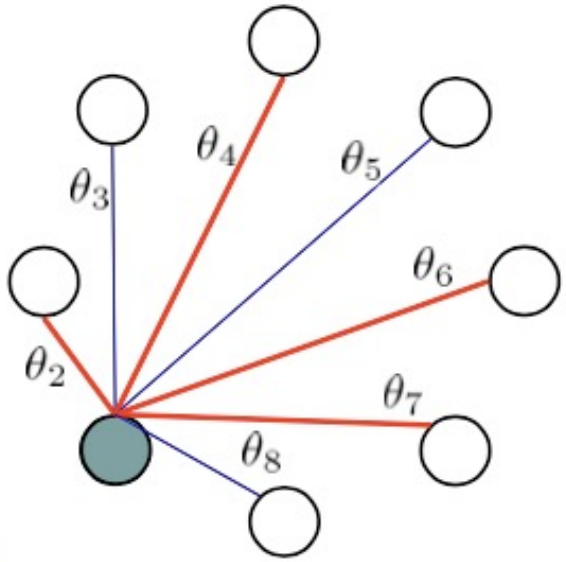
$$Q = \begin{pmatrix} * & * & * & * & * & 0 \\ * & * & * & * & * & 0 \\ * & * & * & 0 & 0 & 0 \\ * & * & 0 & * & 0 & 0 \\ * & * & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * \end{pmatrix}$$



If we can estimate a sample covariance, then we can estimate $Q = \hat{\Sigma}^{-1}$

What if the number of dimensions > number of data points?

Graph Regression: Learn graph structure per node



Neighborhood selection

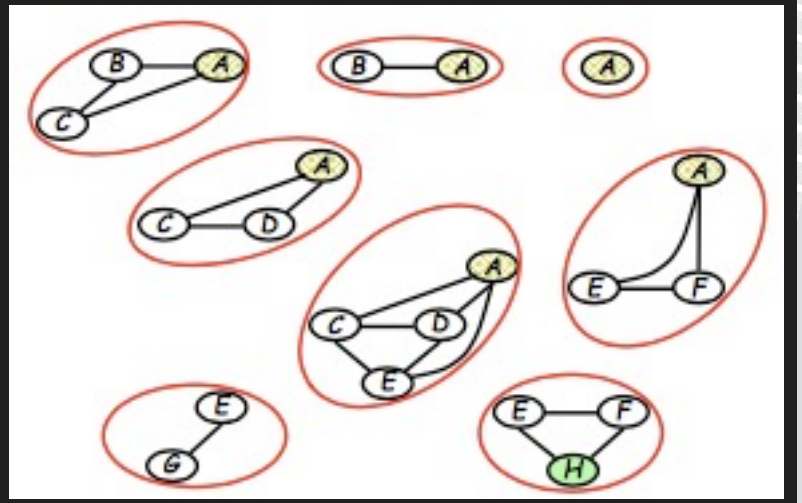
Lasso:

$$\hat{\theta} = \arg\min_{\theta} \sum_{t=1}^{T} l(\theta) + \lambda_1 \| \theta \|_1$$

Gives graph structure.

3. Inference

3a. Exact Inference



Complexity of inference

- Computing $P(X = x \mid e)$ in a GM is **NP-hard**

What does this mean for us?

- Inference cannot be solved in polynomial time unless P=NP.
- No general procedure that works efficiently for arbitrary GMs.
  - For families of GMs, we can have provably efficient procedures.
- Exponential worst-case performance for exact inference.
  - Motivates approximate inference.

Variable Elimination: General form

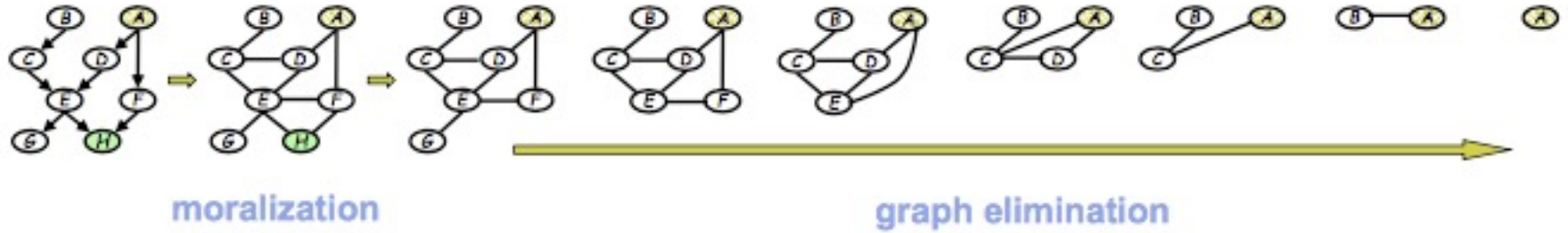
- Write query in the form

$$P(X_1, e) = \sum_{x_d} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i)$$

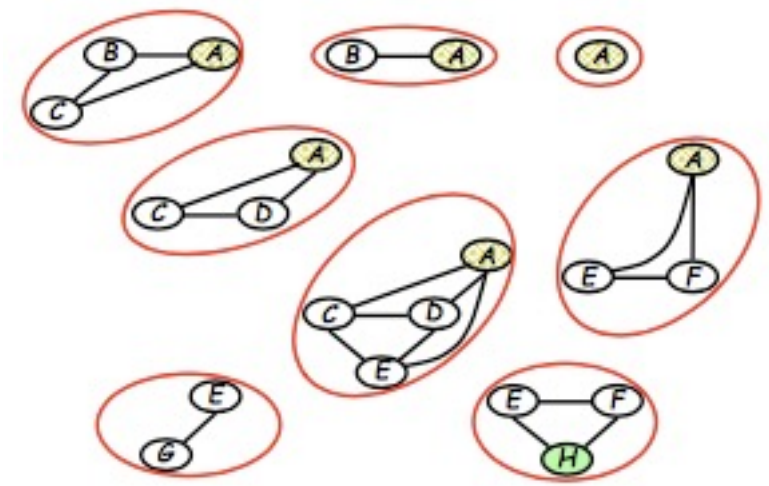
- Then iteratively:
  - Move all irrelevant terms outside of innermost sum.
  - Perform innermost sum, getting a new term.
  - Insert the new term into the product.

Understanding Variable Elimination

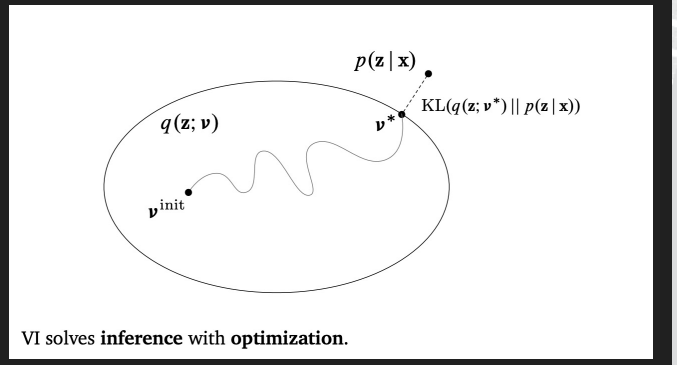
- A graph elimination algorithm



- Intermediate terms correspond to the **cliques** resulted from elimination

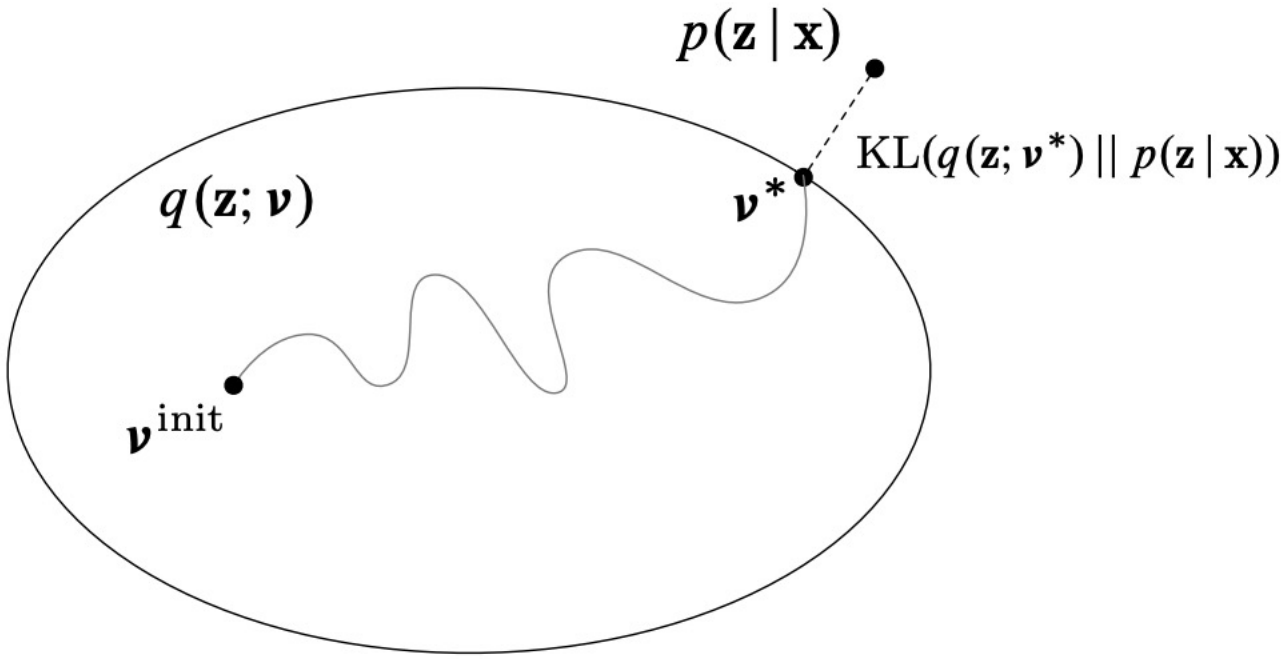


3b. Variational Inference



VI solves **inference** with **optimization**.

Variational Inference



VI solves **inference** with **optimization**.

Variational Inference

$$\log p(x \mid \theta) \geq \underbrace{E_{z \sim q}[\log p(x, z \mid \theta)] + H(q)}_{\text{“ELBO”: Evidence Lower Bound}} \quad + KL(q(z \mid x) \,||\, p(z \mid x, \theta))$$

- We choose a family of variational distributions (i.e., a parameterization of a distribution of the latent variables) such that the expectations are computable.
- Then, we **maximize the ELBO** to find the parameters that gives as tight a bound as possible on the marginal probability of x.

Mean-field VI

- In mean field variational inference, we assume that the variational family factorizes

$$q(z_1, \ldots, z_m) = \prod_{j=1}^{m} q(z_j).$$

- Each variable is independent. (We are suppressing the parameters νj .)
- This is more general that it initially appears—the hidden variables can be grouped and the distribution of each group factorizes.
- Optimize by coordinate ascent:

$$q^*(z_k) \propto \exp\{\mathrm{E}_{-k}[\log p(z_k, Z_{-k}, x)]\}$$

3c. Monte Carlo Methods

Monte Carlo methods: define dist by samples

- Draw random samples from desired distribution
- Yield a stochastic representation of desired distribution
  - $E_p[f(x)] = \frac{\sum_m f(X_m)}{|m|}$
- **Asymptotically exact**
- Challenges:
  - How to draw samples from desired distribution?
  - How to know we've sampled enough?

Monte Carlo Methods

- Direct sampling
- Rejection sampling
- Likelihood weighting
- Markov chain Monte Carlo (MCMC)

Rejection Sampling

- Instead of sampling from $P(X)$, sample x^* from $Q(X)$ and accept sample with probability:
  - $P_{accept}(x^*) = \frac{\mathrm{P}(\mathrm{x}^*)}{MQ(x^*)}$, where M is some constant such that $P(x) \leq MQ(x)\ \forall\ x$
- Works with un-normalized $P(X)$, too.

Unnormalized Importance Sampling

- Instead of hard **rejecting** samples, we can just **reweight** them:

$$E_P[f(X)] = \int_x P(x)f(x)dx = \int_x \frac{P(x)}{Q(x)}Q(x)f(x)dx = E_Q\left[\frac{P(x)}{Q(x)}f(x)\right]$$

- Approximate with empirical:

$$E_P[f(X)] \approx \frac{1}{n}\sum_{i=1,\ldots,n} f(x_i)w(x_i)$$

where $x_i \sim Q$ and $w_i = \frac{P(x_i)}{Q(x_i)}$

What characteristic do we need for this to work?

Normalized Importance Sampling

- Instead of needing access to the normalized probability distribution P, we can also perform importance sampling with an un-normalized $\tilde{P} = aP$ by normalizing the weights according to the sample:
- $\widetilde{w_i} = \frac{w_i}{\sum_i w_i}$

Weighted resampling

- Problem of importance sampling:
  - Performance depends on how well Q matches P.
  - If P(x)f(x) is strongly varying and has a significant proportion of its mass concentrated in a small region, ratio will be dominated by a few samples.
- Solution: use a heavy-tailed Q and weighted resampling.

Limitations of “simple” Monte Carlo

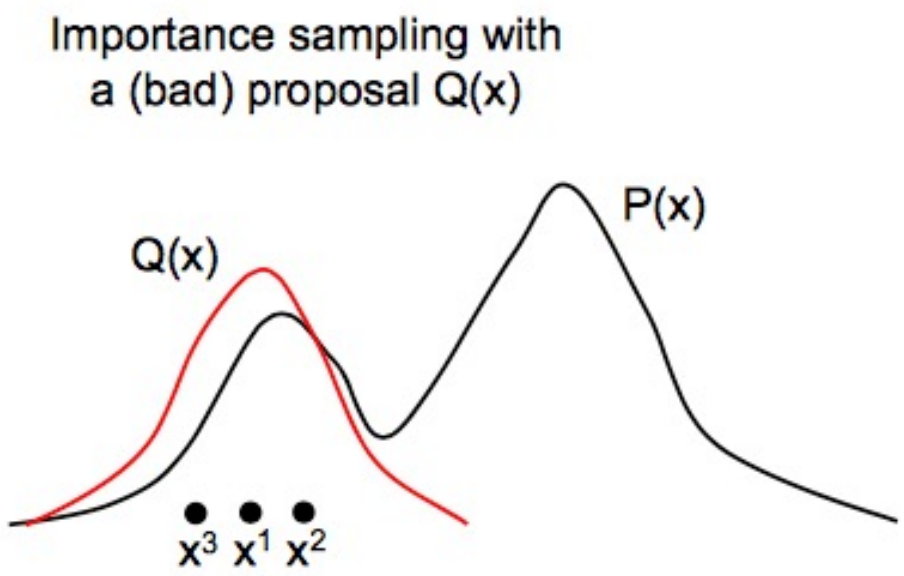
- Hard to get rare events in high-dimensional spaces
- We need a good proposal Q(x) that is not very different than P(x)

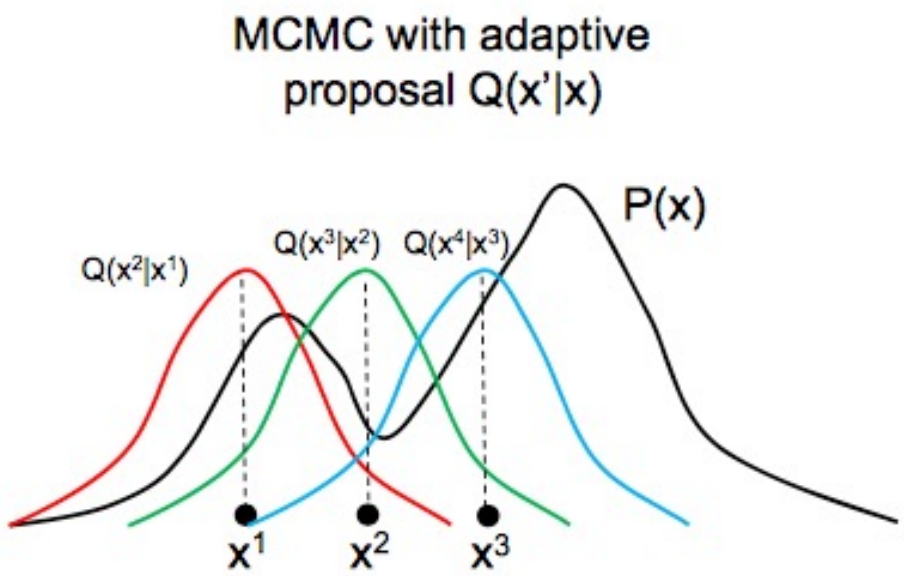
- What if we had an adaptive proposal Q(x)?

Markov Chain Monte Carlo

MCMC algorithms feature adaptive proposals

- Instead of Q(x') use Q(x'|x) where x' is the new state being sampled and x is the previous sample
- As x changes Q(x'|x) can also change





MCMC: Metropolis-Hastings

1. Initialize starting state $x^{(0)}$, set $t = 0$
2. Burn-in: while samples have "not converged"
   - $x = x^{(t)}$
   - $t = t + 1$,
   - sample $x^* \sim Q(x^* | x)$ // draw from proposal
   - sample $u \sim$ Uniform(0,1) // draw acceptance threshold
     - if $u < A(x^* | x) = \min\left(1, \frac{P(x^*)Q(x | x^*)}{P(x)Q(x^* | x)}\right)$
       - $x^{(t)} = x^*$ // transition
     - else
       - $x^{(t)} = x$ // stay in current state

   } Function Draw sample ($x(t)$)

- Take samples from P(x) = : Reset t=0, for $t = 1{:}N$
  - $x(t+1) \leftarrow$ Draw sample ($x(t)$)

Summary

- We can represent complex distributions by composing local structures.
- Structures have complex implications for knowledge, learning, and inference.
- Given a structure, we can learn optimal parameters to match data even if some of the nodes do not have observed data.
- We can estimate GM structure by optimizing a tradeoff between likelihood and a structural cost.
- Given a fitted GM (i.e. a distribution), we can query it for exact statistical answers or approximate the the distribution for faster answers.

Questions?