Directed and Undirected Graphical Models

Adrian Weller

MLSALT4 Lecture
Feb 14, 2018

With thanks to David Sontag (MIT) and Tony Jebara (Columbia)
for use of many slides and illustrations

For more information, see
http://mlg.eng.cam.ac.uk/adrian/

High level overview of our 3 lectures

- 1. Directed and undirected graphical models (today)
- 2. LP relaxations for MAP inference (Friday)
- 3. Junction tree algorithm for exact inference, belief propagation, variational methods for approximate inference (next Wed)

Further reading / viewing:

- Murphy, Machine Learning: a Probabilistic Perspective
- Barber, Bayesian Reasoning and Machine Learning
- Bishop, Pattern Recognition and Machine Learning
- Koller and Friedman, Probabilistic Graphical Models `https://www.coursera.org/course/pgm`
- Wainwright and Jordan, Graphical Models, Exponential Families, and Variational Inference

Key ideas for graphical models

1. **Represent** the world as a collection of random variables $X_1, \ldots, X_n$ with joint distribution $p(X_1, \ldots, X_n)$
2. **Learn** the distribution from data
3. Perform **inference** (typically MAP or marginal)

Challenges

1. **Represent** the world as a collection of random variables $X_1, \ldots, X_n$ with joint distribution $p(X_1, \ldots, X_n)$
   - How can we compactly describe this joint distribution?
   - Directed graphical models (Bayesian networks)
   - Undirected graphical models (Markov random fields, factor graphs)
2. **Learn** the distribution from data
   - Maximum likelihood estimation, other methods?
   - How much data do we need?
   - How much computation does it take?
3. Perform **inference** (typically MAP or marginal)
   - Exact inference: Junction Tree Algorithm
   - Approximate inference (belief propagation, variational methods...)

How can we compactly describe the joint distribution?

Example: Medical diagnosis

- Binary variable for each **symptom** (e.g. "fever", "cough", "fast breathing", "shaking", "nausea", "vomiting")
- Binary variable for each **disease** (e.g. "pneumonia", "flu", "common cold", "bronchitis", "tuberculosis")
- Diagnosis is performed by **inference** in the model:

$$p(\text{pneumonia} = 1 \mid \text{cough} = 1, \text{fever} = 1, \text{vomiting} = 0)$$

- One famous model, Quick Medical Reference (QMR-DT), has 600 diseases and 4000 symptoms

Representing the distribution

- Naively, we could represent the distribution with a big table of probabilities for every possible outcome
- How many outcomes are there in QMR-DT? 2^{4600}
- **Learning** of the distribution would require a huge amount of data
- **Inference** of conditional probabilities, e.g.

$$p(\text{pneumonia} = 1 \mid \text{cough} = 1, \text{fever} = 1, \text{vomiting} = 0)$$

  would require summing over exponentially many values
- Moreover, gives no way to make predictions with previously unseen observations
- We need structure

Structure through independence

- If $X_1, \ldots, X_n$ are independent, then

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

- For binary variables, probabilities for 2^n outcomes can be described by how many parameters?

Structure through independence

- If $X_1, \ldots, X_n$ are independent, then

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

- For binary variables, probabilities for 2^n outcomes can be described by how many parameters? *n*
- However, this is not a very *useful* model – observing a variable X_i cannot influence our predictions of X_j

- Instead: if $X_1, \ldots, X_n$ are **conditionally independent** given Y, denoted as $X_i \perp \mathbf{X}_{-i} \mid Y$, then

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

- This is a simple yet powerful model

Example: naive Bayes for classification

- Classify e-mails as spam ($Y = 1$) or not spam ($Y = 0$)
  - Let $i \in \{1, \ldots, n\}$ index the words in our vocabulary
  - $X_i = 1$ if word i appears in an e-mail, and 0 otherwise
  - E-mails are drawn according to some distribution $p(Y, X_1, \ldots, X_n)$
- Suppose that the words are conditionally independent given Y Then,

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

Easy to **learn** the model with maximum likelihood. **Predict** with:

$$p(Y = 1 \mid x_1, \ldots x_n) = \frac{p(Y = 1) \prod_{i=1}^{n} p(x_i \mid Y = 1)}{\sum_{y=\{0,1\}} p(Y = y) \prod_{i=1}^{n} p(x_i \mid Y = y)}$$

- Is conditional independence a reasonable assumption?

Example: naive Bayes for classification

- Classify e-mails as spam ($Y = 1$) or not spam ($Y = 0$)
  - Let $i \in \{1, \ldots, n\}$ index the words in our vocabulary
  - $X_i = 1$ if word i appears in an e-mail, and 0 otherwise
  - E-mails are drawn according to some distribution $p(Y, X_1, \ldots, X_n)$
- Suppose that the words are conditionally independent given Y
  Then,

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

  Easy to **learn** the model with maximum likelihood. **Predict** with:

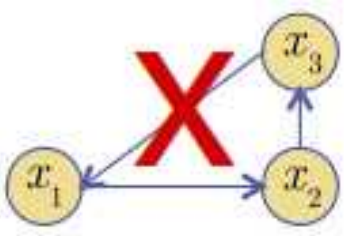
$$p(Y = 1 \mid x_1, \ldots x_n) = \frac{p(Y = 1) \prod_{i=1}^{n} p(x_i \mid Y = 1)}{\sum_{y=\{0,1\}} p(Y = y) \prod_{i=1}^{n} p(x_i \mid Y = y)}$$

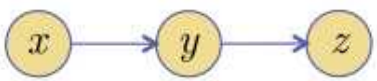
- Is conditional independence a reasonable assumption?
- A model may be "wrong" but still useful

Directed graphical models = Bayesian networks

- A **Bayesian network** is specified by a directed *acyclic* graph DAG$= (V, \vec{E})$ with:
  1. One node $i \in V$ for each random variable X_i
  2. One conditional probability distribution (CPD) per node, $p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$, specifying the variable's probability conditioned on its parents' values
- The DAG corresponds 1-1 with a particular factorization of the joint distribution:

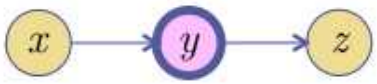
$$p(x_1, \ldots x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$$



Markov chain: 

$$p(x, y, z) = p(x)\,p(y \mid x)\,p(z \mid y)$$

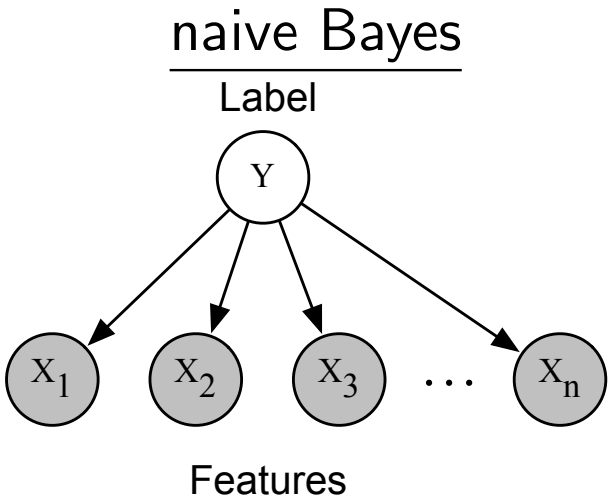
Example binary events:
x = president says war
y = general orders attack
z = soldier shoots gun



$$x \perp\!\!\!\perp z \mid y$$

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

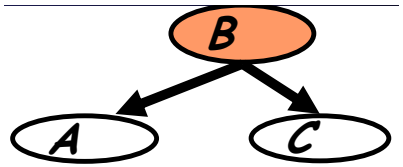
Bayesian networks are *generative models*



- Evidence is denoted by shading in a node
- Can interpret Bayesian network as a **generative process**. For example, to *generate* an e-mail, we
  1. Decide whether it is spam or not spam, by samping $y \sim p(Y)$
  2. For each word $i = 1$ to n, sample $x_i \sim p(X_i \mid Y = y)$

Bayesian network structure ⇒ conditional independencies

- Generalizing earlier example, can show that a variable is independent from its non-descendants given its parents

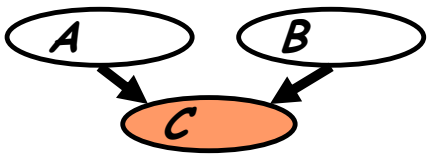


- **Common parent** – fixing B decouples A and C

tail to tail

- **Cascade** – knowing B decouples A and C

head to tail



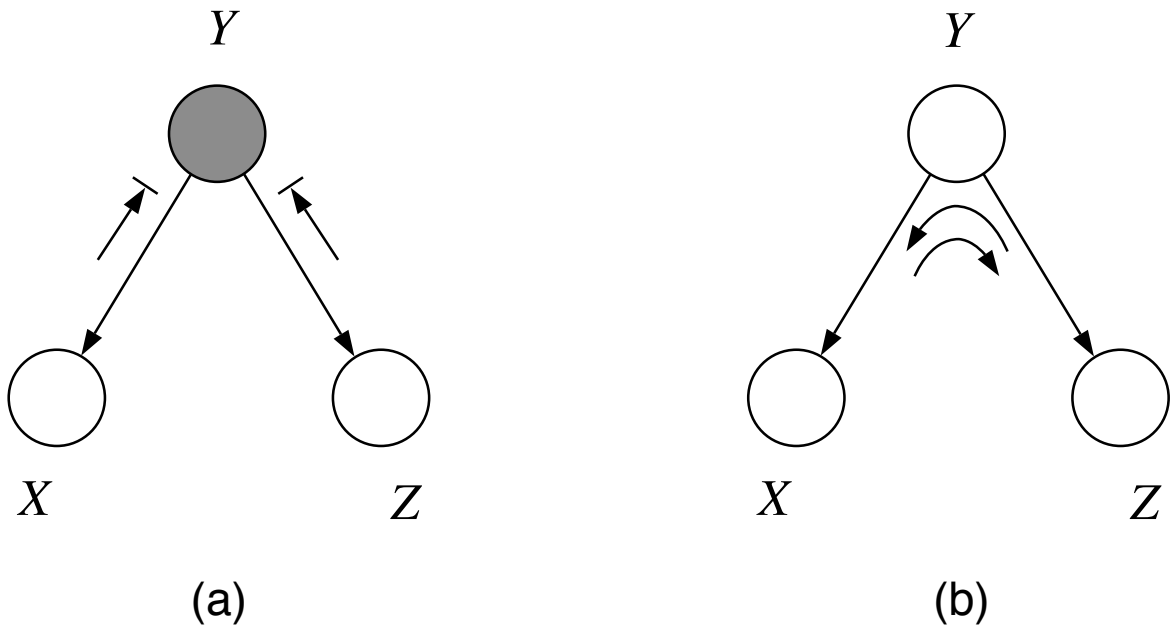
- **V-structure** – Knowing C couples A and B
  - This important phenomona is called **explaining away**

$p(A, B, C) = p(A)p(B)p(C \mid A, B)$

head to head

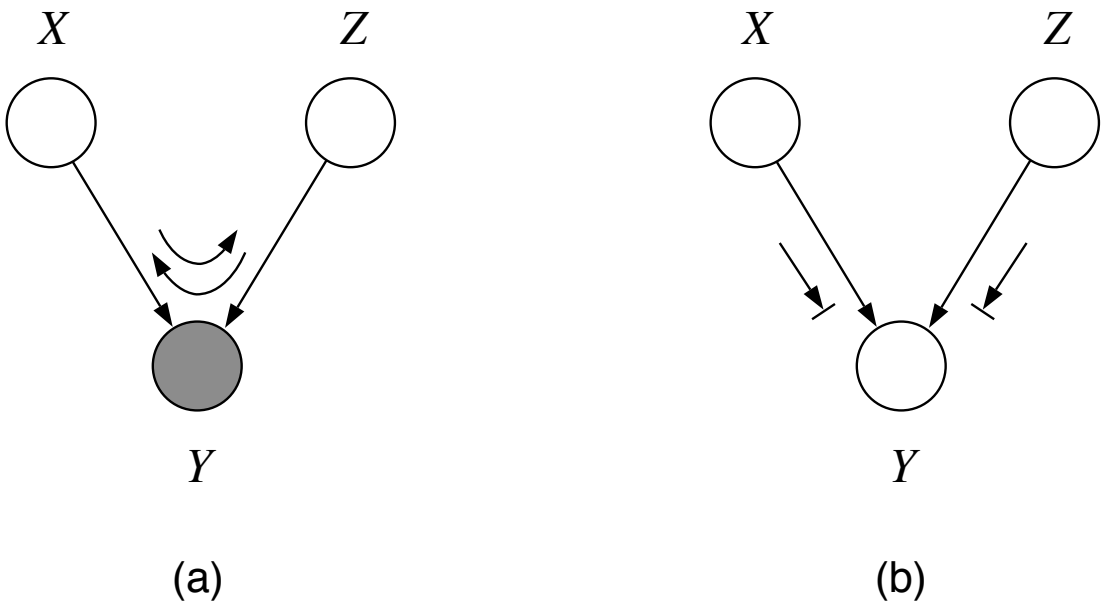
D-separation ("directed separation") in Bayesian networks

- Bayes Ball Algorithm to determine whether $X \perp Z \mid \mathbf{Y}$ by looking at graph d-separation
- Look to see if there is **active path** between X and Z when variables $\mathbf{Y}$ are observed:



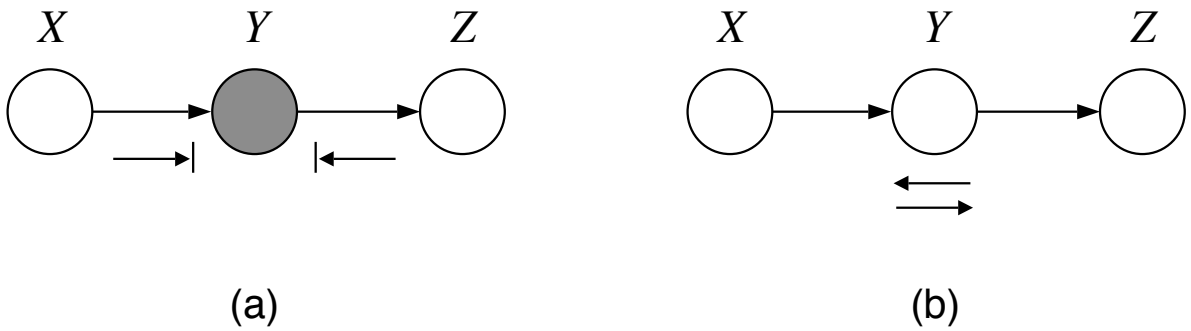
D-separation ("directed separation") in Bayesian networks

- Bayes Ball Algorithm to determine whether $X \perp Z \mid \mathbf{Y}$ by looking at graph d-separation
- Look to see if there is **active path** between X and Z when variables $\mathbf{Y}$ are observed:



D-separation ("directed separation") in Bayesian networks

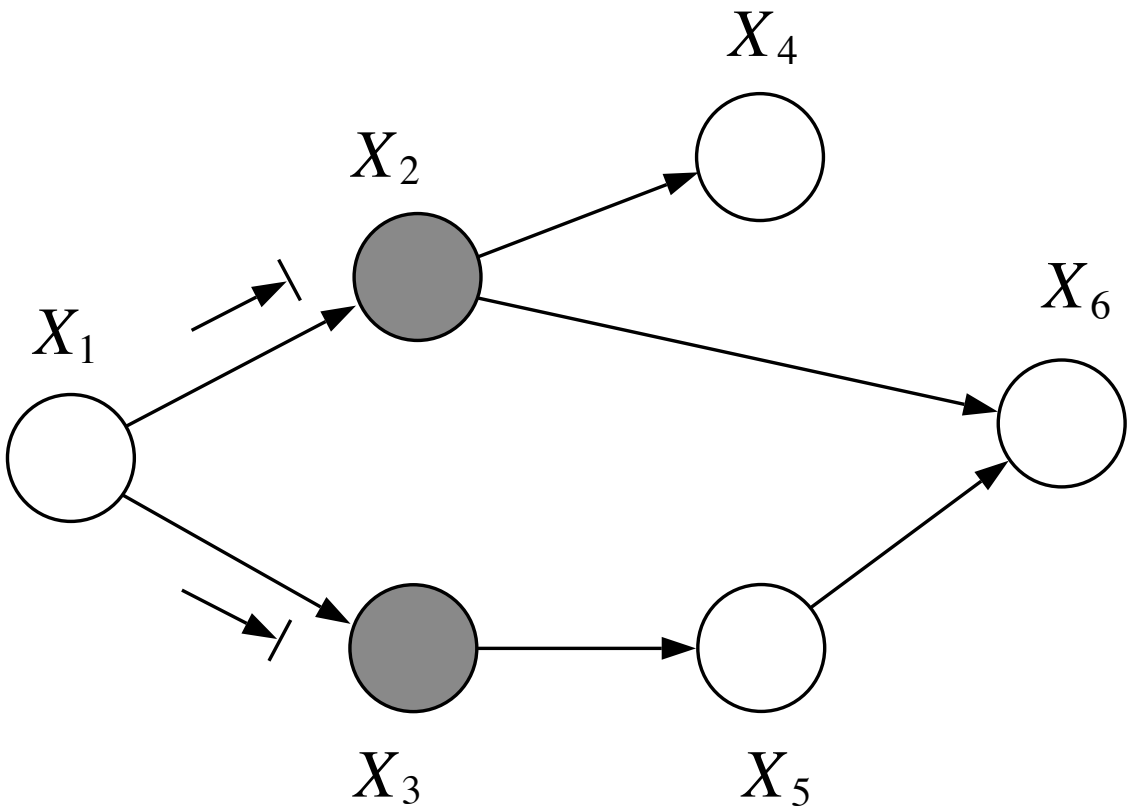
- Bayes Ball Algorithm to determine whether $X \perp Z \mid \mathbf{Y}$ by looking at graph d-separation
- Look to see if there is **active path** between X and Z when variables $\mathbf{Y}$ are observed:



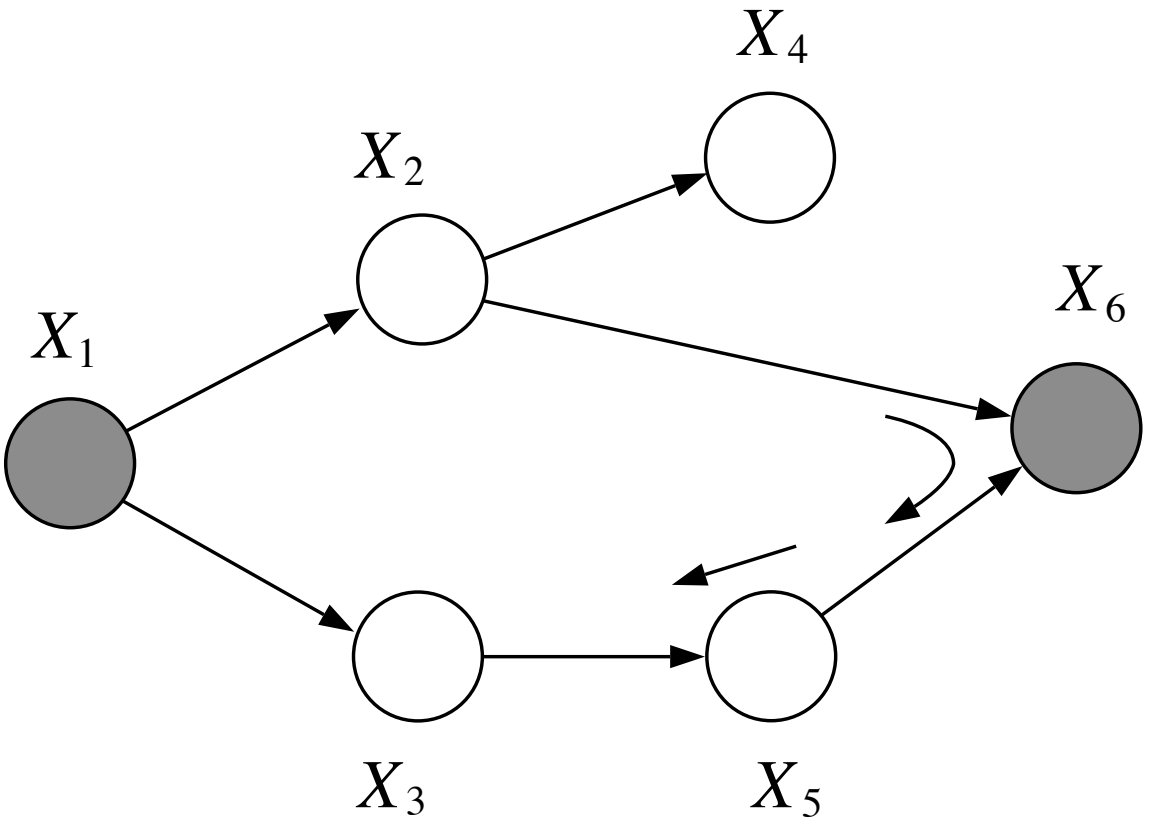
(a) (b)

- If no such path, then X and Z are **d-separated** with respect to $\mathbf{Y}$
- d-separation reduces statistical independencies (hard) to connectivity in graphs (easy)
- Important because it allows us to quickly prune the Bayesian network, finding just the relevant variables for answering a query

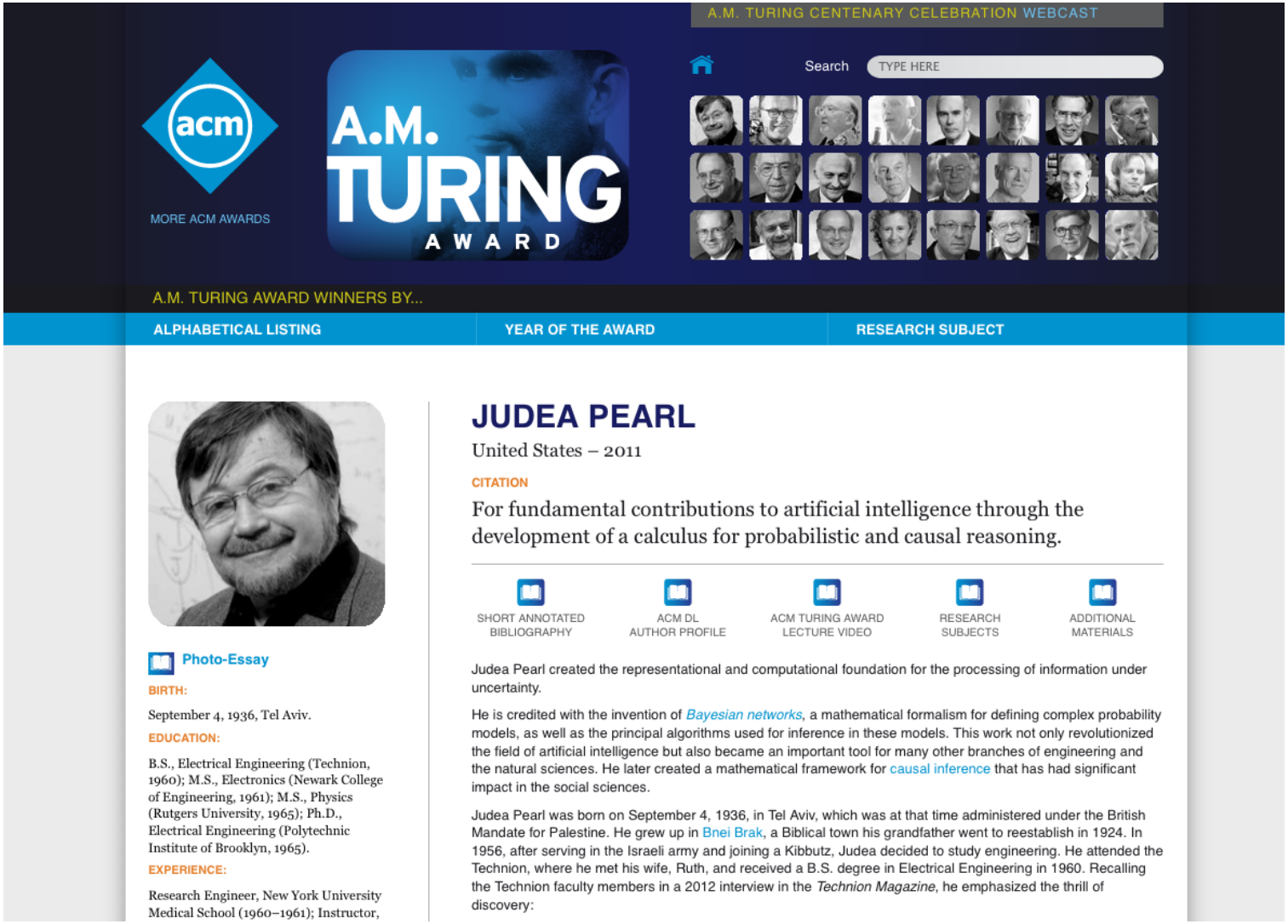
D-separation example 1



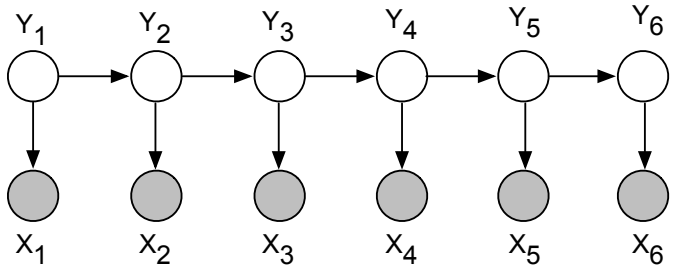
D-separation example 2



2011 Turing Award was for Bayesian networks



Example: hidden Markov model (HMM)

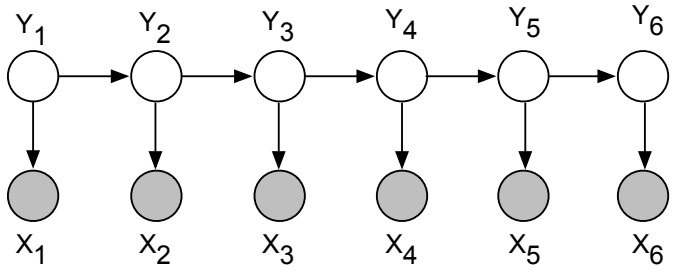


- Frequently used for speech recognition and part-of-speech tagging
- Joint distribution factors as:

$$p(\mathbf{y}, \mathbf{x}) = p(y_1)p(x_1 \mid y_1) \prod_{t=2}^{T} p(y_t \mid y_{t-1})p(x_t \mid y_t)$$

  - $p(y_1)$ is the initial distribution of the starting state
  - $p(y_t \mid y_{t-1})$ is the transition probability between hidden states
  - $p(x_t \mid y_t)$ is the emission probability
- What are the conditional independencies here?

Example: hidden Markov model (HMM)



- Frequently used for speech recognition and part-of-speech tagging
- Joint distribution factors as:

$$p(\mathbf{y}, \mathbf{x}) = p(y_1)p(x_1 \mid y_1) \prod_{t=2}^{T} p(y_t \mid y_{t-1})p(x_t \mid y_t)$$

  - $p(y_1)$ is the initial distribution of the starting state
  - $p(y_t \mid y_{t-1})$ is the transition probability between hidden states
  - $p(x_t \mid y_t)$ is the emission probability
- What are the conditional independencies here?
  Many, e.g. $Y_1 \perp \{Y_3, \ldots, Y_6\} \mid Y_2$

Summary

- A **Bayesian network** specifies the global distribution by a DAG and local conditional probability distributions (CPDs) for each node
- Can interpret as a generative model, where variables are sampled in topological order
- Examples: naive Bayes, hidden Markov models (HMMs), latent Dirichlet allocation
- Conditional independence via d-separation
- Compute the probability of any assignment by multiplying CPDs
- Maximum likelihood learning of CPDs is easy (decomposes, can estimate each CPD separately)

Undirected graphical models

- An alternative representation for a joint distribution is an **undirected graphical model**
- As for directed models, we have one node for each random variable
- Rather than CPDs, we specify (non-negative) **potential functions** over sets of variables associated with cliques C of the graph,

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

Z is the **partition function** and normalizes the distribution:

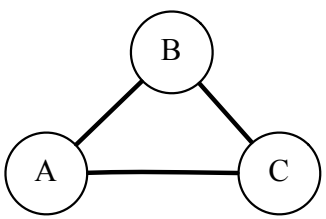
$$Z = \sum_{\hat{x}_1, \ldots, \hat{x}_n} \prod_{c \in C} \phi_c(\hat{\mathbf{x}}_c)$$

- Like a CPD, $\phi_c(\mathbf{x}_c)$ can be represented as a table, but it is not normalized
- Also known as **Markov random fields** (MRFs) or Markov networks

Undirected graphical models

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(\mathbf{x}_c), \qquad Z = \sum_{\hat{x}_1, \ldots, \hat{x}_n} \prod_{c \in C} \phi_c(\hat{\mathbf{x}}_c)$$

Simple example (each edge potential function encourages its variables to take the same value):



$\phi_{A,B}(a,b) =$

A \ B	0	1
0	10	1
1	1	10

$\phi_{B,C}(b,c) =$

B \ C	0	1
0	10	1
1	1	10

$\phi_{A,C}(a,c) =$

A \ C	0	1
0	10	1
1	1	10

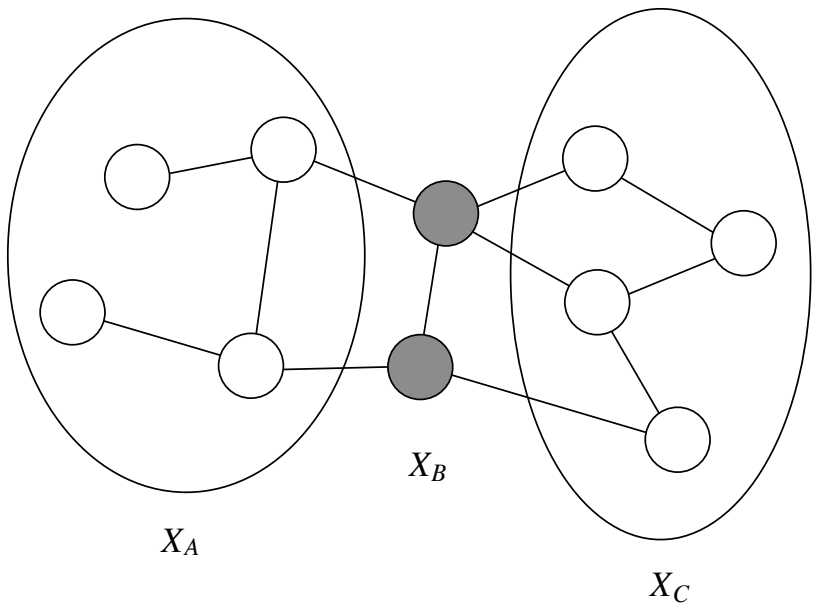
$$p(a,b,c) = \frac{1}{Z} \phi_{A,B}(a,b) \cdot \phi_{B,C}(b,c) \cdot \phi_{A,C}(a,c),$$

where

$$Z = \sum_{\hat{a},\hat{b},\hat{c} \in \{0,1\}^3} \phi_{A,B}(\hat{a},\hat{b}) \cdot \phi_{B,C}(\hat{b},\hat{c}) \cdot \phi_{A,C}(\hat{a},\hat{c}) = 2 \cdot 1000 + 6 \cdot 10 = 2060.$$

Markov network structure $\Rightarrow$ conditional independencies

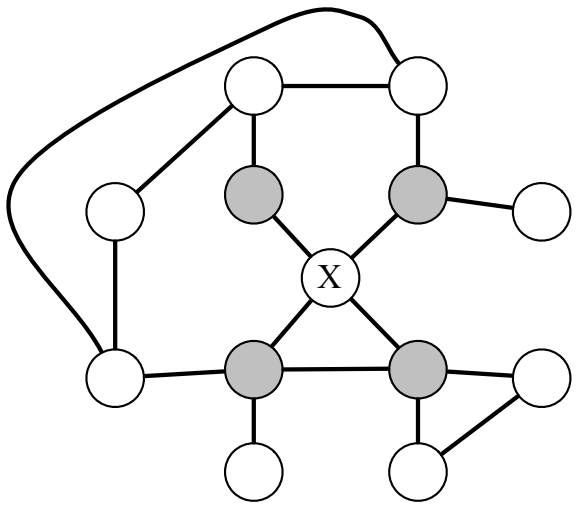
- Let G be the undirected graph where we have one edge for every pair of variables that appear together in a potential
- Conditional independence is given by **graph separation**



- $X_{\mathbf{A}} \perp X_{\mathbf{C}} \mid X_{\mathbf{B}}$ if there is no path from $a \in \mathbf{A}$ to $c \in \mathbf{C}$ after removing all variables in $\mathbf{B}$

Markov blanket

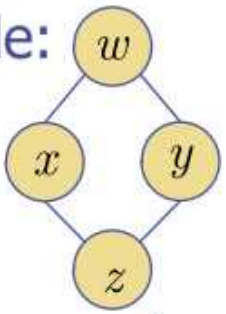
- A set **U** is a **Markov blanket** of X if $X \notin \mathbf{U}$ and if **U** is a minimal set of nodes such that $X \perp (\mathcal{X} - \{X\} - \mathbf{U}) \mid \mathbf{U}$
- In undirected graphical models, the Markov blanket of a variable is precisely its **neighbors** in the graph:



- In other words, X is independent of the rest of the nodes in the graph given its immediate neighbors

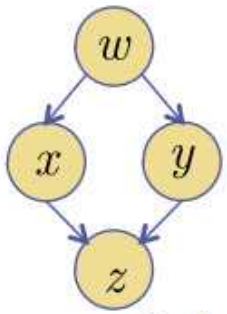
Directed and undirected models are different

- Example:



$$x \perp\!\!\!\perp y \mid \{w, z\}$$

$$w \perp\!\!\!\perp z \mid \{x, y\}$$

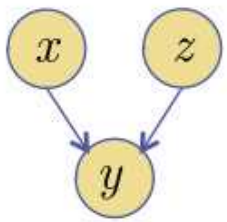


$$x \perp\!\!\!\perp y \mid \{w\}$$

$$x \not\perp\!\!\!\perp y \mid \{w, z\}$$

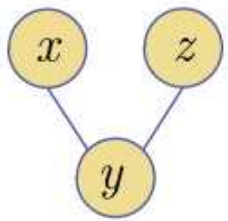
Directed can't do it!
Must be acyclic
Will have at least one V structure and ball goes through

- Example:



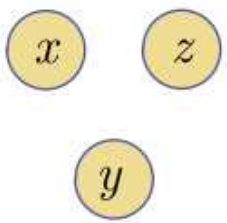
$$x \perp\!\!\!\perp z$$

$$x \not\perp\!\!\!\perp z \mid y$$



Undirected can't do it!

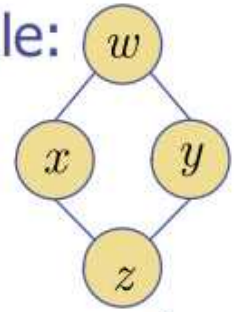
$$x \perp\!\!\!\perp z \mid y$$



$$x \perp\!\!\!\perp z$$

Directed and undirected models are different

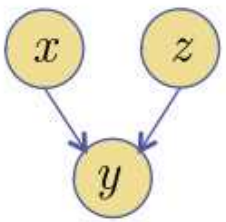
- Example:



$$x \perp\!\!\!\perp y \mid \{w, z\}$$

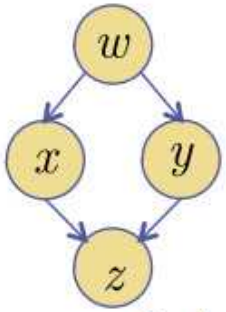
$$w \perp\!\!\!\perp z \mid \{x, y\}$$

- Example:



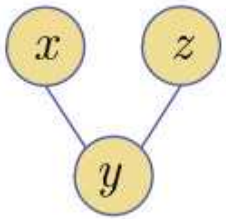
$$x \perp\!\!\!\perp z$$

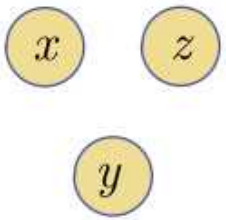
$$x \not\perp\!\!\!\perp z \mid y$$



$$x \perp\!\!\!\perp y \mid \{w\}$$

$$x \not\perp\!\!\!\perp y \mid \{w, z\}$$





Directed can't do it!
Must be acyclic
Will have at least one V structure and ball goes through

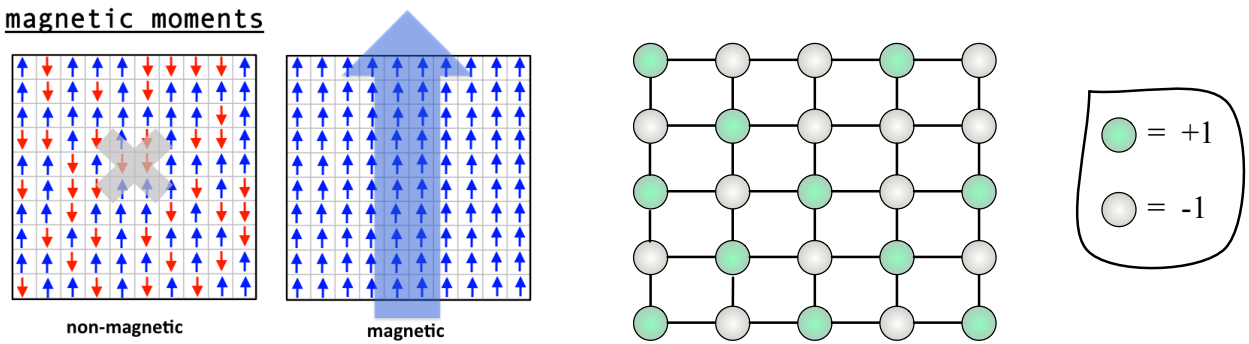
With <3 edges,
Undirected can't do it!

$$x \perp\!\!\!\perp z \mid y$$

$$x \perp\!\!\!\perp z$$

Example: Ising model

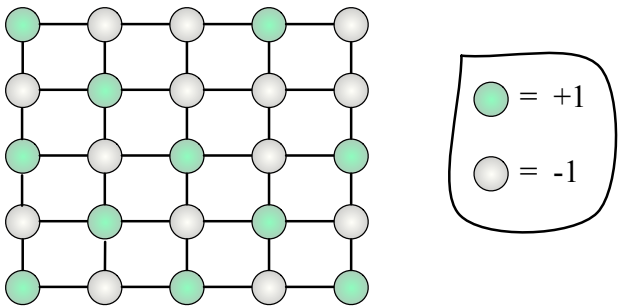
- Invented by the physicist Wilhelm Lenz (1920), who gave it as a problem to his student Ernst Ising
- Mathematical model of ferromagnetism in statistical mechanics
- The spin of an atom is influenced by the spins of atoms nearby on the material:



- Each atom $X_i \in \{-1, +1\}$, whose value is the direction of the atom spin
- If a spin at position i is $+1$, what is the probability that the spin at position j is also $+1$?
- Are there phase transitions where spins go from "disorder" to "order"?

Example: Ising model

- Each atom $X_i \in \{-1, +1\}$, whose value is the direction of the atom spin
- The spin of an atom is influenced by the spins of atoms nearby on the material:



$$p(x_1, \cdots, x_n) = \frac{1}{Z} \exp \frac{1}{T} \left(\sum_{i<j} w_{i,j} x_i x_j + \sum_i \theta_i x_i \right)$$

- When $w_{i,j} > 0$, adjacent atoms encouraged to have the same spin (**attractive** or **ferromagnetic**); $w_{i,j} < 0$ encourages $X_i \neq X_j$
- Node potentials θ_i encode the bias of the individual atoms
- Varying the temperature T makes the distribution more or less spiky

Supplementary material

Extra slides for questions or further explanation

Basic idea

- Suppose we have a simple chain $A \to B \to C \to D$, we want to compute $p(D)$
- $p(D)$ is a **set** of values, $\{p(D = d), d \in \mathrm{Val}(D)\}$. Algorithm computes sets of values at a time – an entire distribution
- The joint distribution factors as

$$p(A, B, C, D) = p(A)p(B \mid A)p(C \mid B)p(D \mid C)$$

- In order to compute $p(D)$, we have to marginalize over A, B, C:

$$p(D) = \sum_{a,b,c} p(A = a, B = b, C = c, D)$$

How can we perform the sum efficiently?

- Our goal is to compute

$$p(D) = \sum_{a,b,c} p(a,b,c,D) = \sum_{a,b,c} p(a)p(b \mid a)p(c \mid b)p(D \mid c)$$
$$= \sum_{c}\sum_{b}\sum_{a} p(D \mid c)p(c \mid b)p(b \mid a)p(a)$$

- We can push the summations inside to obtain:

$$p(D) = \sum_{c} p(D \mid c) \sum_{b} p(c \mid b) \underbrace{\sum_{a} \underbrace{p(b \mid a)p(a)}_{\psi_1(a,b)}}_{\tau_1(b)}$$

- Let's call $\psi_1(A,B) = P(A)P(B|A)$. Then, $\tau_1(B) = \sum_a \psi_1(a,B)$
- Similarly, let $\psi_2(B,C) = \tau_1(B)P(C|B)$. Then, $\tau_2(C) = \sum_b \psi_1(b,C)$
- This procedure is dynamic programming: computation is inside out instead of outside in

Inference in a chain

- Generalizing the previous example, suppose we have a chain $X_1 \to X_2 \to \cdots \to X_n$, where each variable has k states
- For $i = 1$ up to $n - 1$, compute (and cache)

$$p(X_{i+1}) = \sum_{x_i} p(X_{i+1} \mid x_i) p(x_i)$$

- Each update takes k^2 time (why?)
- The total running time is $\mathcal{O}(nk^2)$
- In comparison, naively marginalizing over all latent variables has complexity $\mathcal{O}(k^n)$
- We did inference over the joint without ever explicitly constructing it!

ML learning in Bayesian networks

- Maximum likelihood learning: $\max_\theta \ell(\theta; \mathcal{D})$, where

$$\begin{aligned}\ell(\theta; \mathcal{D}) = \log p(\mathcal{D}; \theta) &= \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x}; \theta) \\ &= \sum_i \sum_{\hat{\mathbf{x}}_{pa(i)}} \sum_{\substack{\mathbf{x} \in \mathcal{D}: \\ \mathbf{x}_{pa(i)} = \hat{\mathbf{x}}_{pa(i)}}} \log p(x_i \mid \hat{\mathbf{x}}_{pa(i)})\end{aligned}$$

- In Bayesian networks, we have the closed form ML solution:

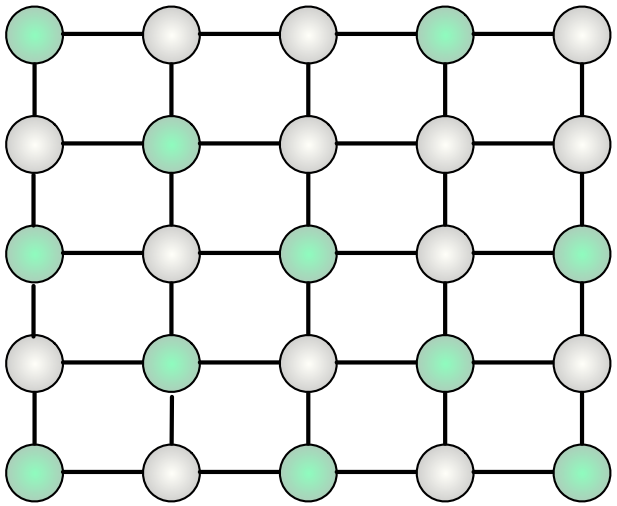
$$\theta^{ML}_{x_i | \mathbf{x}_{pa(i)}} = \frac{N_{x_i, \mathbf{x}_{pa(i)}}}{\sum_{\hat{x}_i} N_{\hat{x}_i, \mathbf{x}_{pa(i)}}}$$

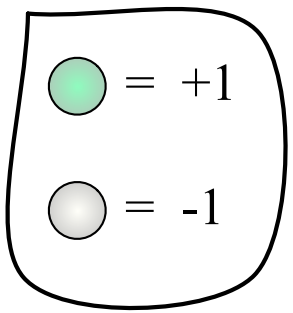
  where $N_{x_i, \mathbf{x}_{pa(i)}}$ is the number of times that the (partial) assignment $x_i, \mathbf{x}_{pa(i)}$ is observed in the training data

- We can estimate each CPD independently because the objective **decomposes** by variable and parent assignment

Parameter learning in Markov networks

- How do we learn the parameters of an Ising model?





$$p(x_1, \cdots, x_n) = \frac{1}{Z} \exp\left(\sum_{i<j} w_{i,j} x_i x_j + \sum_i \theta_i x_i\right)$$

Bad news for Markov networks

- The global normalization constant $Z(\theta)$ kills decomposability:

$$\begin{aligned}
\theta^{ML} &= \arg\max_{\theta} \log \prod_{\mathbf{x} \in \mathcal{D}} p(\mathbf{x}; \theta) \\
&= \arg\max_{\theta} \sum_{\mathbf{x} \in \mathcal{D}} \left(\sum_{c} \log \phi_c(\mathbf{x}_c; \theta) - \log Z(\theta) \right) \\
&= \arg\max_{\theta} \left(\sum_{\mathbf{x} \in \mathcal{D}} \sum_{c} \log \phi_c(\mathbf{x}_c; \theta) \right) - |\mathcal{D}| \log Z(\theta)
\end{aligned}$$

- The log-partition function prevents us from decomposing the objective into a sum over terms for each potential
- Solving for the parameters becomes much more complicated