人工智能技术应用

概率模型

Review: probability (example)

Random variables: sunshine $S \in \{0, 1\}$, rain $R \in \{0, 1\}$

Joint distribution:

$P(S, R) =$

s	r	$P(S = s, R = r)$
0	0	0.20
0	1	0.08
1	0	0.70
1	1	0.02

Marginal distribution:

$P(S) =$

s	$P(S = s)$
0	0.28
1	0.72

(aggregate rows)

Conditional distribution:

$P(S \mid R = 1) =$

s	$P(S = s \mid R = 1)$
0	0.8
1	0.2

(select rows, normalize)

Review: probability (general)

Random variables:

$X = (X_1, \ldots, X_n)$ partitioned into (A, B)

Joint distribution:

$\mathrm{P}(X) = \mathrm{P}(X_1, \ldots, X_n)$

Marginal distribution:

$\mathrm{P}(A) = \sum_b \mathrm{P}(A, B = b)$

Conditional distribution:

$\mathrm{P}(A \mid B = b) \propto \mathrm{P}(A, B = b)$

Probabilistic inference task

Random variables: unknown quantities in the world

$$X = (S, R, T, A)$$

In words:

- Observe evidence (traffic in autumn): $T = 1, A = 1$
- Interested in query (rain?): R

In symbols:

$$\mathbb{P}(\underbrace{R}_{\text{query}} \mid \underbrace{T = 1, A = 1}_{\text{condition}})$$

(S is **marginalized out**)

Challenges

Modeling: How to specify a joint distribution $\mathrm{P}(X_1, \ldots, X_n)$ **compactly**?

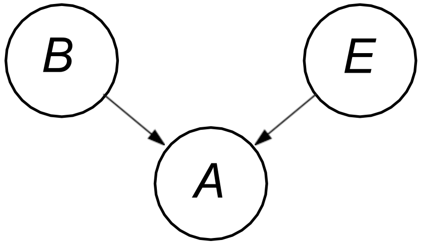
Bayesian networks (factor graphs for probability distributions)

Inference: How to compute queries $\mathrm{P}(R \mid T = 1, A = 1)$ **efficiently**?

Variable elimination, Gibbs sampling, particle filtering (analogue of algorithms for finding maximum weight assignment)

Bayesian network (alarm)

$$\mathrm{P}(B = b, E = e, A = a) = {\color{blue}p(b)}{\color{green}p(e)}{\color{red}p(a \mid b, e)}$$



b	$p(b)$
1	s
0	$1-s$

e	$p(e)$
1	s
0	$1-s$

b	e	a	$p(a \mid b, e)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$p(b) = s \cdot [b = 1] + (1 - s) \cdot [b = 0]$$

$$p(e) = s \cdot [e = 1] + (1 - s) \cdot [e = 0]$$

$$p(a \mid b, e) = [a = (b \vee e)]$$

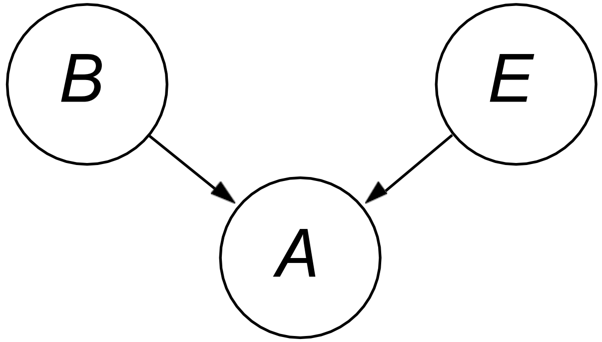
Probabilistic inference (alarm)

Joint distribution:

b	e	a	$P(B = b, E = e, A = a)$
0	0	0	$(1-s)^2$
0	0	1	0
0	1	0	0
0	1	1	$(1-s)s$
1	0	0	0
1	0	1	$s(1-s)$
1	1	0	0
1	1	1	s^2

Queries: $P(B)$? $P(B \mid A = 1)$? $P(B \mid A = 1, E = 1)$?

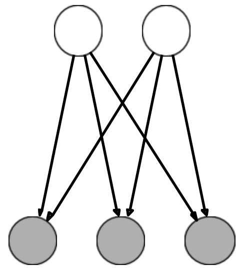
Explaining away



Key idea: explaining away

Suppose two causes positively influence an effect. Conditioned on the effect, conditioning on one cause reduces the probability of the other cause.

Definition



Definition: Bayesian network

Let $X = (X_1, \ldots, X_n)$ be random variables.

A **Bayesian network** is a directed acyclic graph (DAG) that specifies a joint distribution over X as a product of local conditional distributions, one for each node:

$$P(X_1 = x_1, \ldots, X_n = x_n) \stackrel{\text{def}}{=} \prod_{i=1}^{n} p(x_i \mid x_{\text{Parents}(i)})$$

Special properties

Key difference from general factor graphs:

Key idea: locally normalized

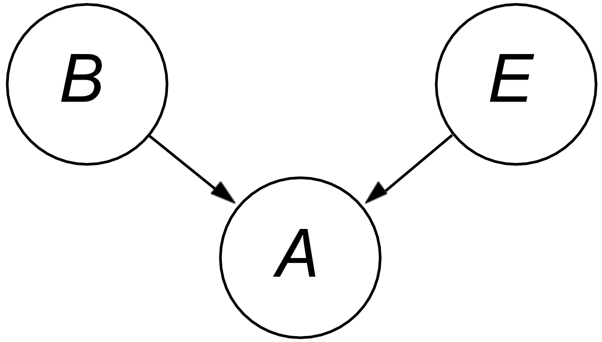
All factors (local conditional distributions) satisfy:

$$\sum_{x_i} p(x_i \mid x_{\text{Parents}(i)}) = 1 \text{ for each } x_{\text{Parents}(i)}$$

Implications:

- Consistency of sub-Bayesian networks
- Consistency of conditional distributions

Consistency of sub-Bayesian networks



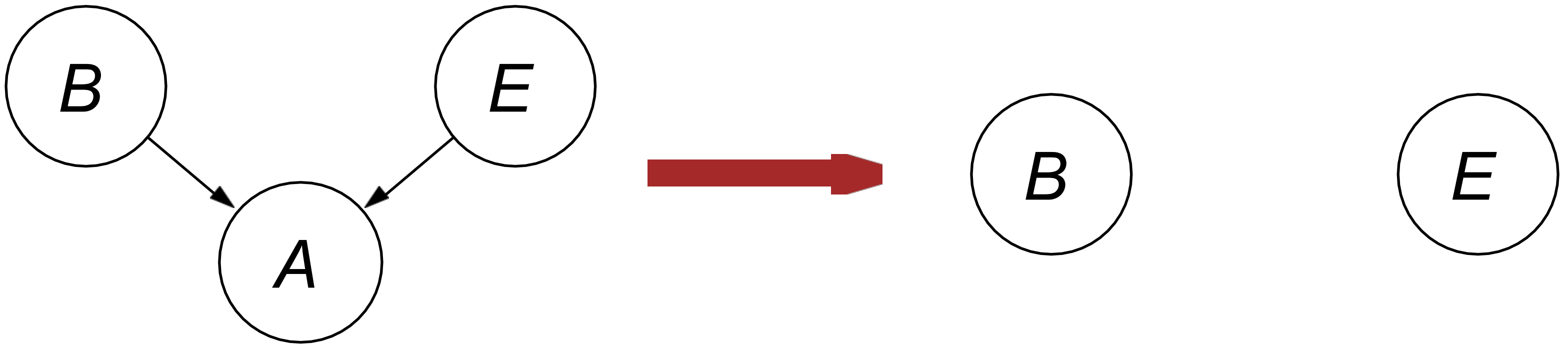
A short calculation:

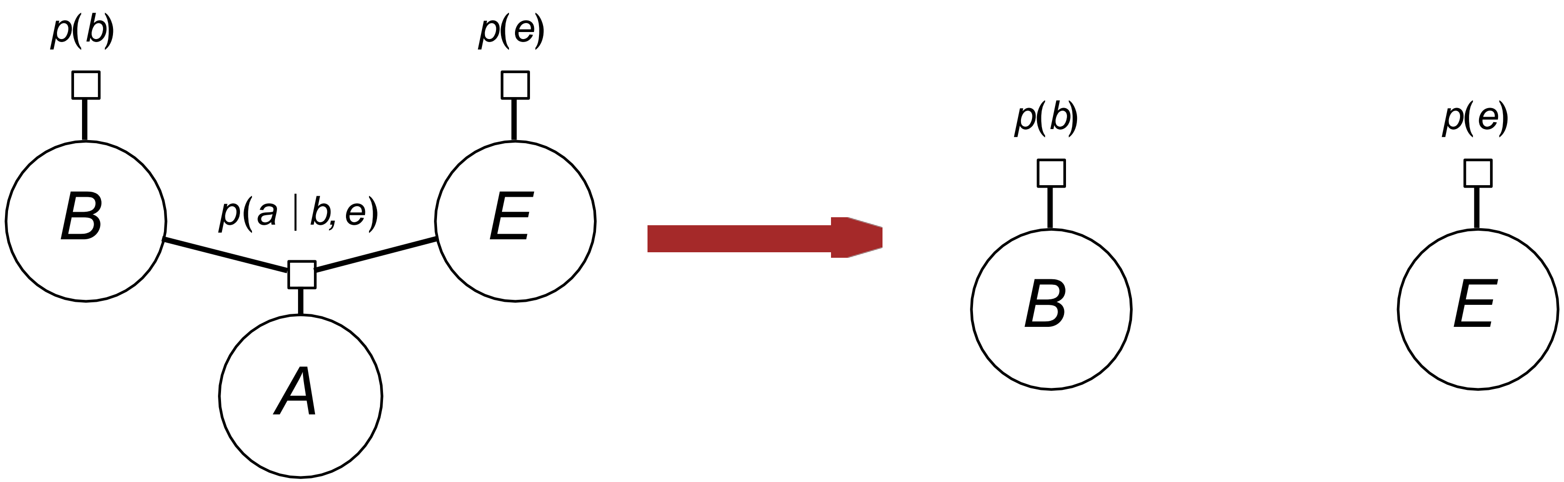
$$\begin{aligned}\mathbb{P}(B=b, E=e) &= \textstyle\sum_a \mathbb{P}(B=b, E=e, A=a) \\ &= \textstyle\sum_a p(b)p(e)p(a \mid b, e) \\ &= p(b)p(e) \textstyle\sum_a p(a \mid b, e) \\ &= p(b)p(e)\end{aligned}$$

Consistency of sub-Bayesian networks

Key idea: marginalization

Marginalization of a leaf node yields a Bayesian network without the node.

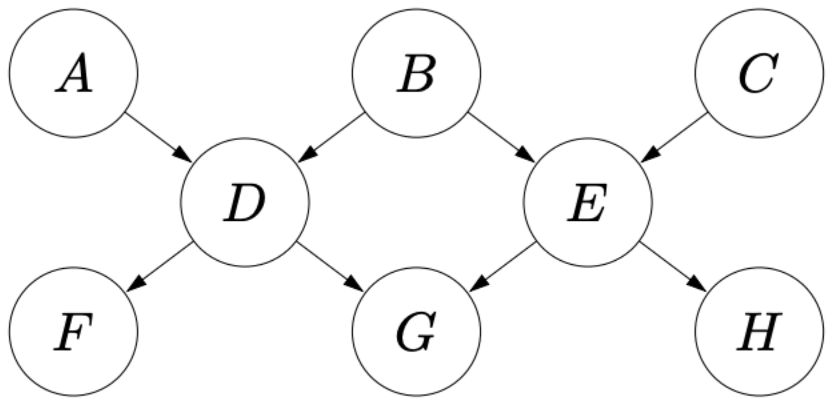




Consistency of local conditionals

Key idea: local conditional distributions

Local conditional distributions (factors) are the true conditional distributions.



$$\underbrace{\mathbb{P}(D = d \mid A = a, B = b)}_{\text{from probabilistic inference}} = \underbrace{p(d \mid a, b)}_{\text{by definition}}$$

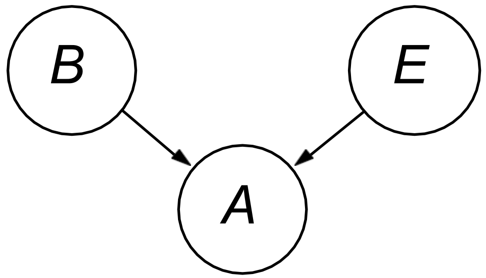
Probabilistic programs

Goal: make it easier to write down complex Bayesian networks

Key idea: probabilistic program

Write a program to generate an assignment (rather than specifying the probability of an assignment).

Probabilistic programs



Probabilistic program: alarm

$$B \sim \text{Bernoulli}(s)$$
$$E \sim \text{Bernoulli}(s)$$
$$A = B \vee E$$

Key idea: probabilistic program

A randomized program that sets the random variables.

```
def Bernoulli(epsilon):
    return random.random() < epsilon
```

Probabilistic program: example

Probabilistic program: object tracking

$X_0 = (0,0)$

For each time step $i = 1, \ldots, n$:

  With probability α:

    $X_i = X_{i-1} + (1,0)$ [go right]

  With probability $1 - \alpha$:

    $X_i = X_{i-1} + (0,1)$ [go down]

Bayesian network structure:

$$X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \rightarrow X_5$$

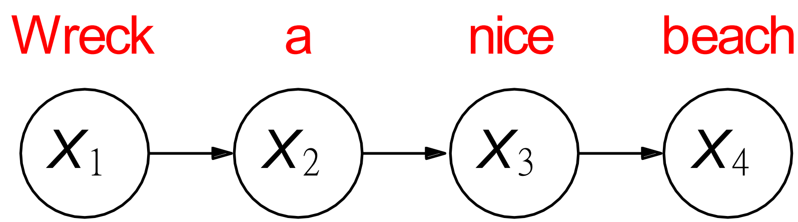
Markov model

Application: language modeling

Probabilistic program: Markov model

For each position $i = 1, 2, \ldots, n$:

Generate word $X_i \sim p(X_i \mid X_{i-1})$



higher-order Markov models

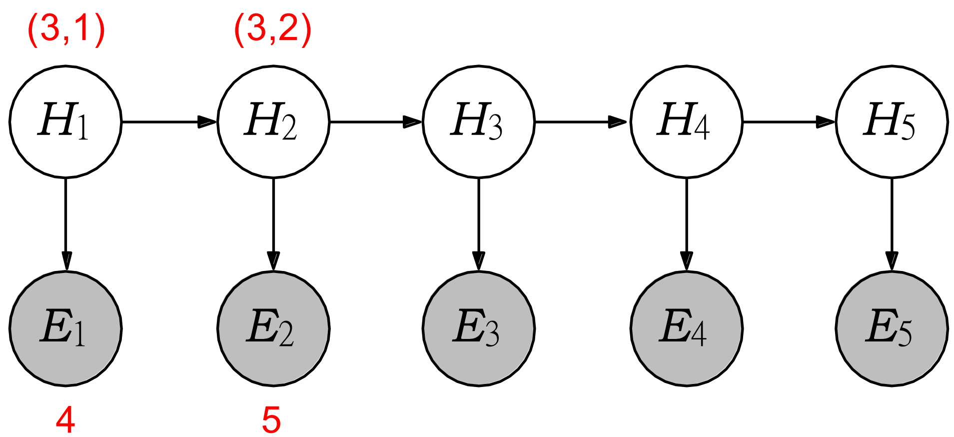
Application: object tracking

Probabilistic program: hidden Markov model (HMM)

For each time step $t = 1, \ldots, T$:

Generate object location $H_t \sim p(H_t \mid H_{t-1})$

Generate sensor reading $E_t \sim p(E_t \mid H_t)$



Applications: speech recognition, information extraction, gene finding

Application: multiple object tracking

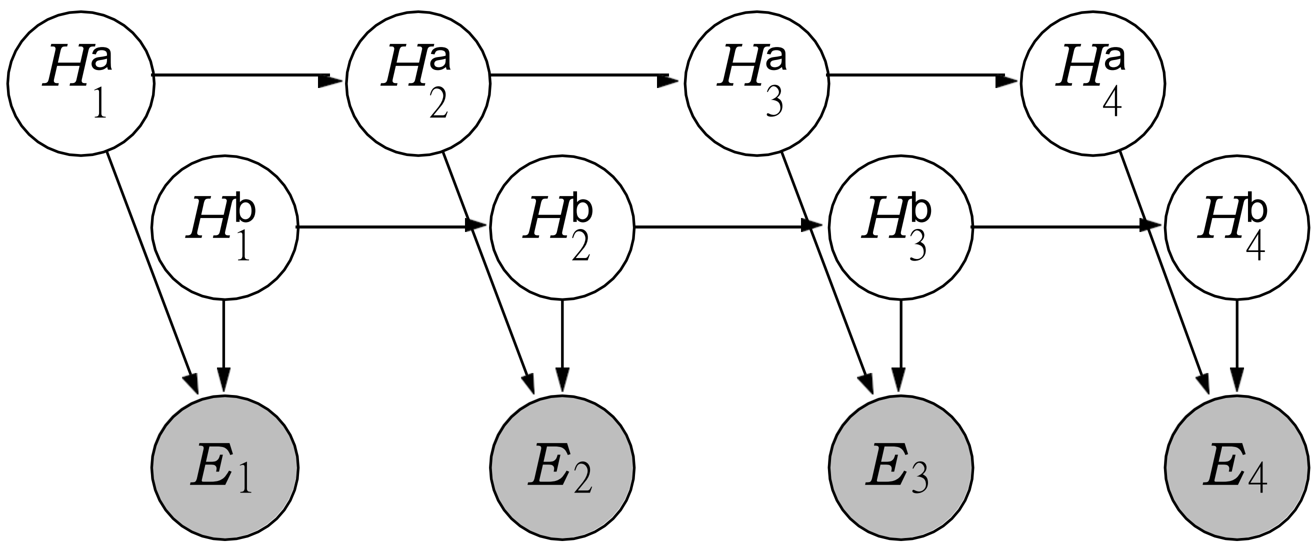
Probabilistic program: factorial HMM

For each time step $t = 1, \ldots, T$:

For each object $o \in \{a, b\}$:

Generate location $H_t^o \sim p(H_t^o \mid H_{t-1}^o)$

Generate sensor reading $E_t \sim p(E_t \mid H_t^a, H_t^b)$



Application: document classification

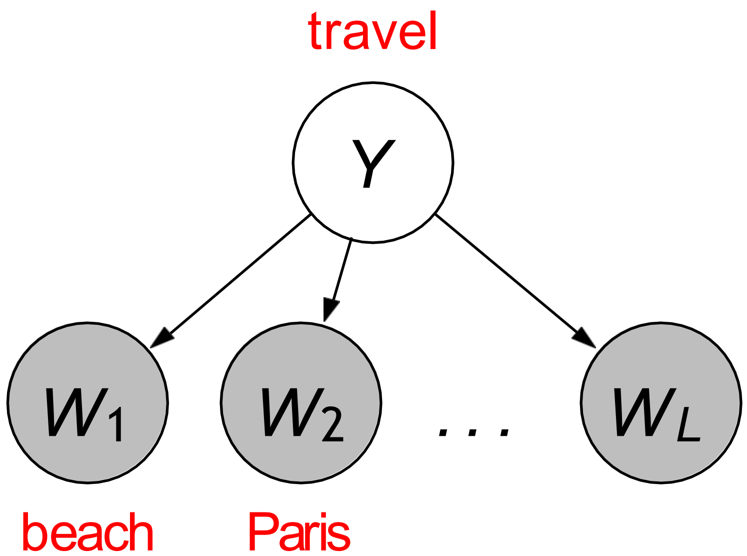
Question: given a text document, what is it about?

Probabilistic program: naive Bayes

Generate label $Y \sim p(Y)$

For each position $i = 1, \dots, L$:

  Generate word $W_i \sim p(W_i \mid Y)$



Application: topic modeling

Question: given a text document, what topics is it about?

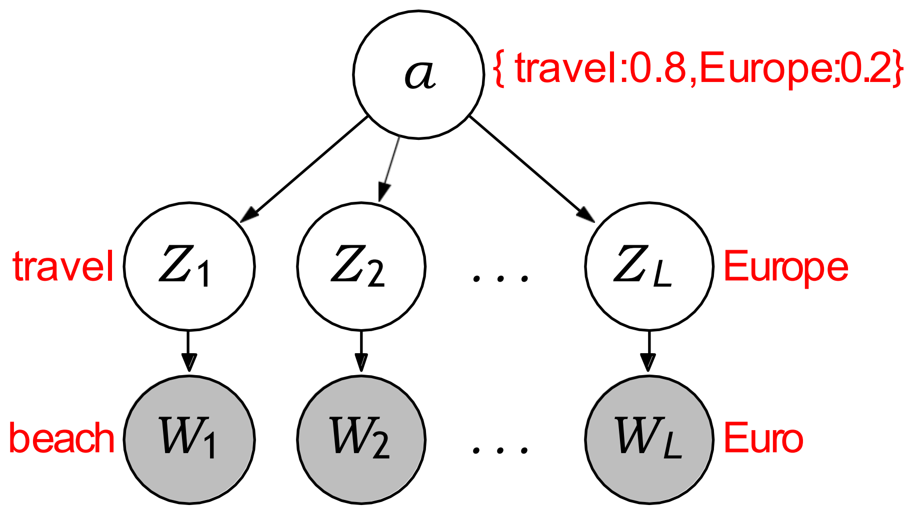
Probabilistic program: latent Dirichlet allocation

Generate a distribution over topics $a \in \mathbb{R}^K$

For each position $i = 1, \ldots, L$:

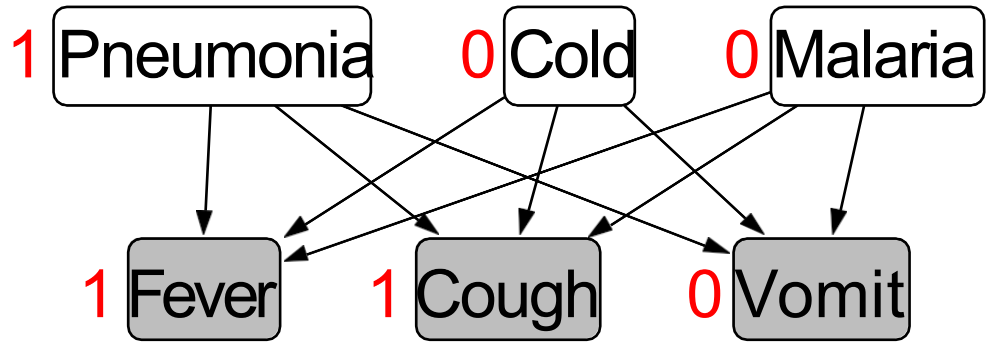
Generate a topic $Z_i \sim p(Z_i \mid a)$

Generate a word $W_i \sim p(W_i \mid Z_i)$



Application: medical diagnostics

Question: If patient has has a cough and fever, what disease(s) does he/she have?



Probabilistic program: diseases and symptoms

For each disease $i = 1, \ldots, m$:

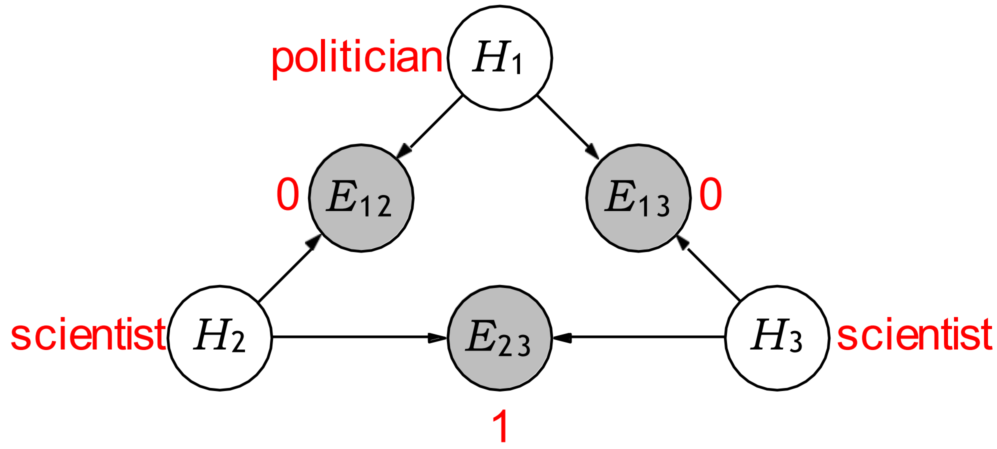
Generate activity of disease $D_i \sim p(D_i)$

For each symptom $j = 1, \ldots, n$:

Generate activity of symptom $S_j \sim p(S_j \mid D_{1:m})$

Application: social network analysis

Question: Given a social network (graph over n people), what types of people are there?



Probabilistic program: stochastic block model

For each person $i = 1, \ldots, n$:

  Generate person type $H_i \sim p(H_i)$

For each pair of people $i \neq j$:

  Generate connectedness $E_{ij} \sim p(E_{ij} \mid H_i, H_j)$

probabilistic inference

Input

Bayesian network: $P(X_1 = x_1, \ldots, X_n = x_n)$

Evidence: $E = e$ where $E \subseteq X$ is subset of variables

Query: $Q \subseteq X$ is subset of variables

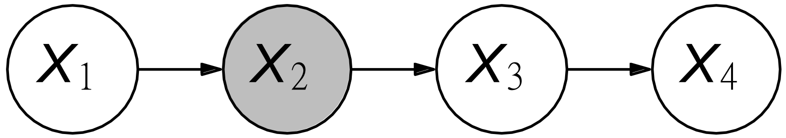
Output

$P(Q = q \mid E = e)$ for all values q

Example: if coughing and itchy eyes, have a cold?

$$P(C \mid H = 1, I = 1)$$

Example: Markov model



Query: $\mathbb{P}(X_3 = x_3 \mid X_2 = 5)$ for all x_3

Tedious way:

$$\propto \sum_{x_1, x_4} p(x_1) p(x_2 = 5 \mid x_1) p(x_3 \mid x_2 = 5) p(x_4 \mid x_3)$$

$$\propto \left(\sum_{x_1} p(x_1) p(x_2 = 5 \mid x_1) \right) p(x_3 \mid x_2 = 5)$$

$$\propto p(x_3 \mid x_2 = 5)$$

General strategy

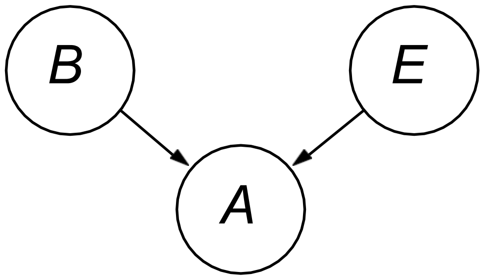
Query:

$$\mathrm{P}(Q \mid E = e)$$

Algorithm: general probabilistic inference strategy

- Remove (marginalize) variables that are not ancestors of Q or E.
- Convert Bayesian network to factor graph.
- Condition on $E = e$ (shade nodes + disconnect).
- Remove (marginalize) nodes disconnected from Q.
- Run probabilistic inference algorithm (manual, variable elimination, Gibbs sampling, particle filtering).

Example: alarm



b	$p(b)$
1	s
0	$1-s$

e	$p(e)$
1	s
0	$1-s$

b	e	a	$p(a \mid b, e)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

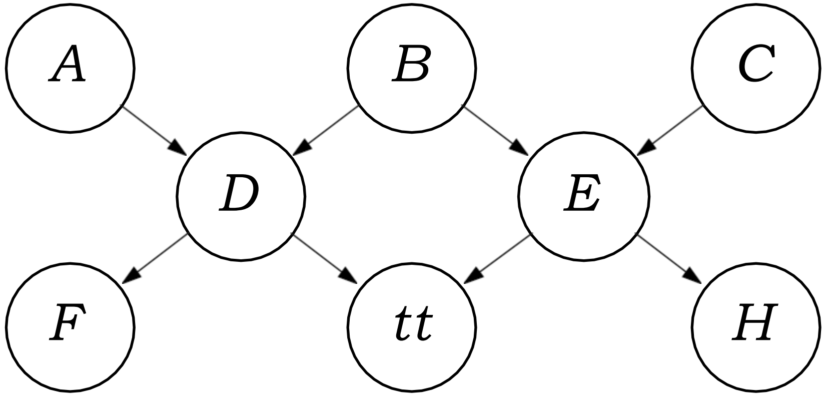
Query: $P(B)$

- Marginalize out A, E

Query: $P(B \mid A = 1)$

- Condition on $A = 1$

Example: A-H (section)



[whiteboard]

Query: $\mathrm{P}(C \mid B = b)$

- Marginalize out everything else, note $C \perp B$

Query: $\mathrm{P}(C, H \mid E = e)$

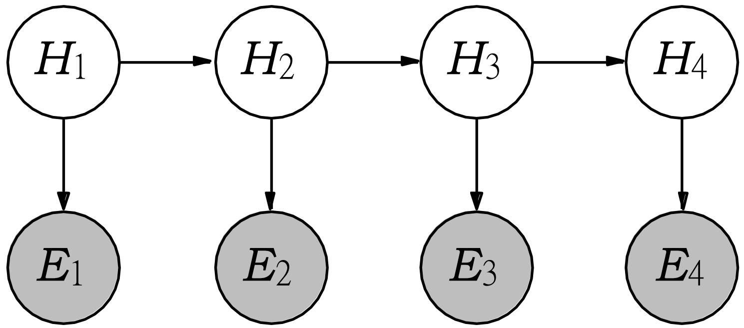
- Marginalize out A, D, F, tt, note $C \perp H \mid E$

Roadmap

Forward-backward

Gibbs sampling

Object tracking



Problem: object tracking

$H_i \in \{1, \ldots, K\}$: location of object at time step i

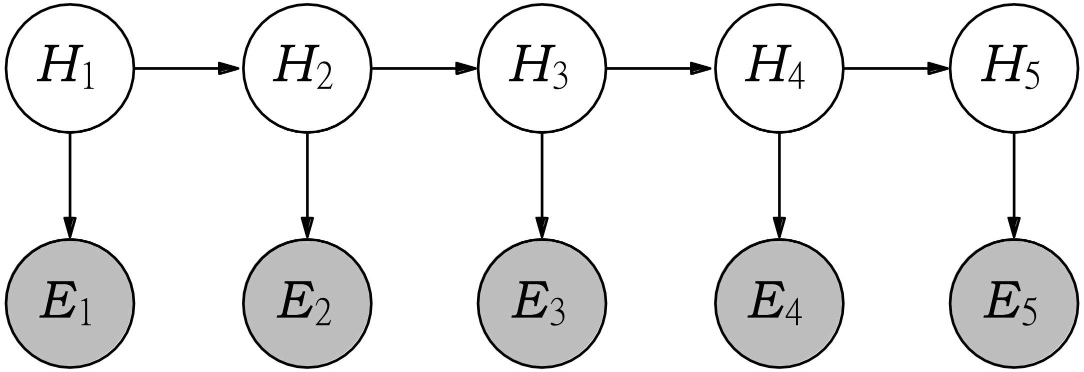
$E_i \in \{1, \ldots, K\}$: sensor reading at time step i

Start: $p(h_1)$: uniform over all locations

Transition $p(h_i \mid h_{i-1})$: uniform over adjacent loc.

Emission $p(e_i \mid h_i)$: uniform over adjacent loc.

Hidden Markov model



$$\mathbb{P}(H = h, E = e) = \underbrace{p(h_1)}_{\text{start}} \prod_{i=2}^{n} \underbrace{p(h_i \mid h_{i-1})}_{\text{transition}} \prod_{i=1}^{n} \underbrace{p(e_i \mid h_i)}_{\text{emission}}$$

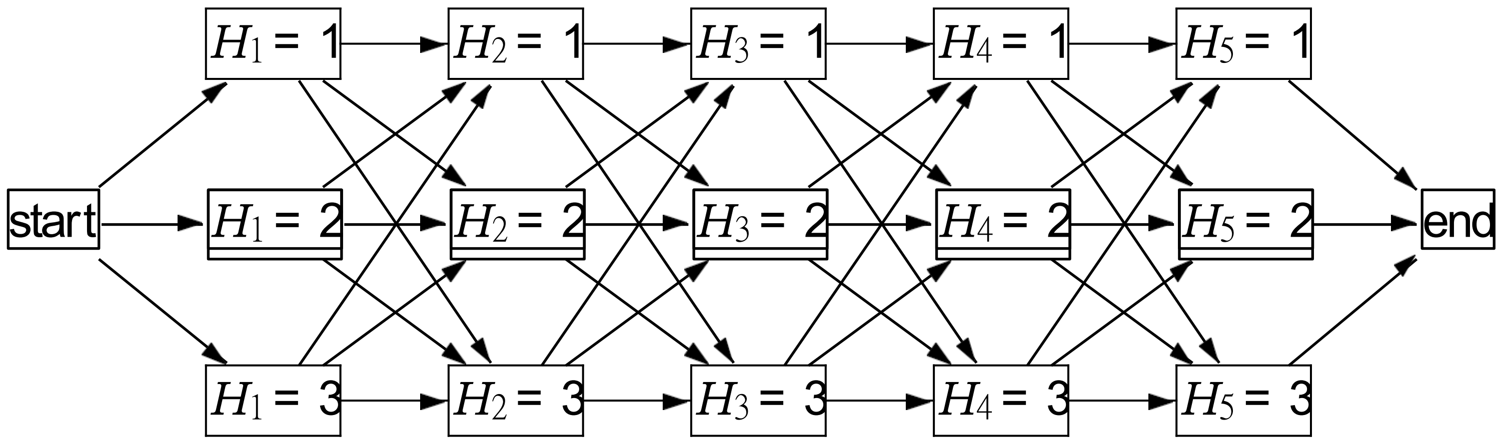
Query (**filtering**):

$$\mathsf{P}(H_3 \mid E_1 = e_1, E_2 = e_2, E_3 = e_3)$$

Query (**smoothing**):

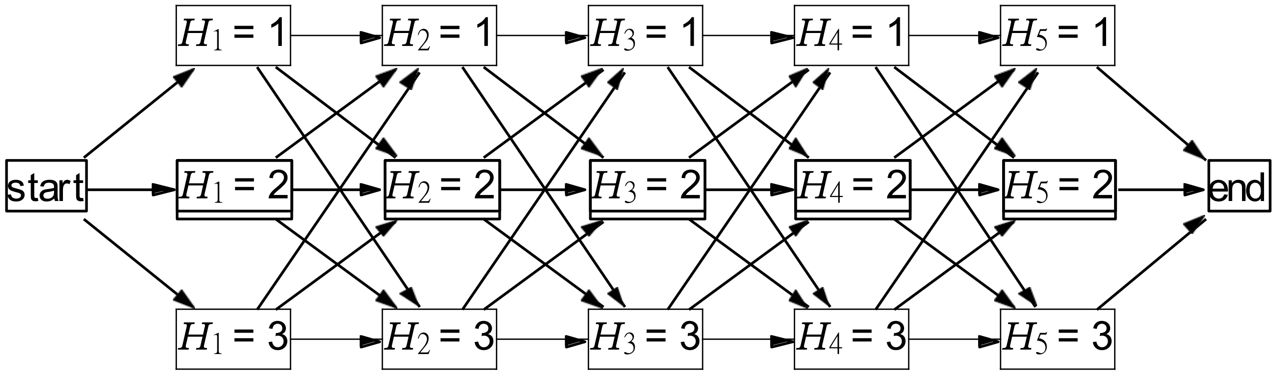
$$\mathsf{P}(H_3 \mid E_1 = e_1, E_2 = e_2, E_3 = e_3, E_4 = e_4, E_5 = e_5)$$

Lattice representation



- Edge $\boxed{\text{start} \Rightarrow}$ $\boxed{H_1 = h_1}$ has weight $p(h_1)p(e_1 \mid h_1)$
- Edge $\boxed{H_{i-1} = h_{i-1}}$ $\Rightarrow$ $\boxed{H_i = h_i}$ has weight $p(h_i \mid h_{i-1})p(e_i \mid h_i)$
- Each path from $\boxed{\text{start}}$ to $\boxed{\text{end}}$ is an assignment with weight equal to the product of node/edge weights

Lattice representation



Forward: $F_i(h_i) = \sum_{h_{i-1}} F_{i-1}(h_{i-1}) w(h_{i-1}, h_i)$

sum of weights of paths from $\boxed{\text{start}}$ to $\boxed{H_i = h_i}$

Backward: $B_i(h_i) = \sum_{h_{i+1}} B_{i+1}(h_{i+1}) w(h_i, h_{i+1})$

sum of weights of paths from $\boxed{H_i = h_i}$ to $\boxed{\text{end}}$

Define $S_i(h_i) = F_i(h_i) B_i(h_i)$:

sum of weights of paths from $\boxed{\text{start}}$ to $\boxed{\text{end}}$ through $\boxed{H_i = h_i}$

Lattice representation

Smoothing queries (marginals):

$$\mathsf{P}(H_i = h_i \mid E = e) \propto S_i(h_i)$$

Algorithm: forward-backward algorithm

Compute $F_1, F_2, \ldots, F_L$

Compute $B_L, B_{L-1}, \ldots, B_1$

Compute S_i for each i and normalize

Running time: $O(LK^2)$

Roadmap

Forward-backward

Gibbs sampling

Particle-based approximation

$$P(X_1, X_2, X_3)$$

Key idea: particles

Use a small set of assignments (particles) to represent a large probability distribution.

x_1	x_2	x_3	$P(X_1 = x_1, X_2 = x_2, X_3 = x_3)$
0	0	0	0.18
0	0	1	0.02
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0.08
1	1	1	0.72
0.8	**0.8**	**0.74**	**(true marginals)**

	x_1	x_2	x_3
Sample 1	0	0	0
Sample 2	1	1	1
Sample 3	1	1	1
Estimated marginals	**0.67**	**0.67**	**0.67**

Gibbs sampling

Algorithm: Gibbs sampling

Initialize x to a random complete assignment

Loop through $i = 1, \ldots, n$ until convergence:

Compute weight of $x \cup \{X_i : v\}$ for each v

Choose $x \cup \{X_i : v\}$ with probability prop. to weight

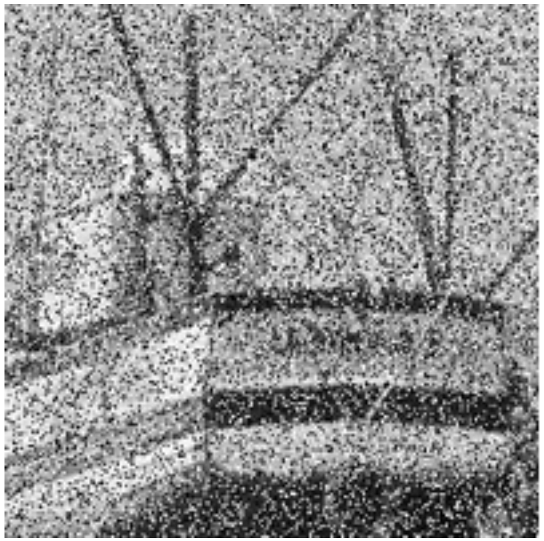
Gibbs sampling (probabilistic interpretation)

Loop through $i = 1, \ldots, n$ until convergence:

Set $X_i = v$ with prob. $\mathbb{P}(X_i = v \mid X_{-i} = x_{-i})$

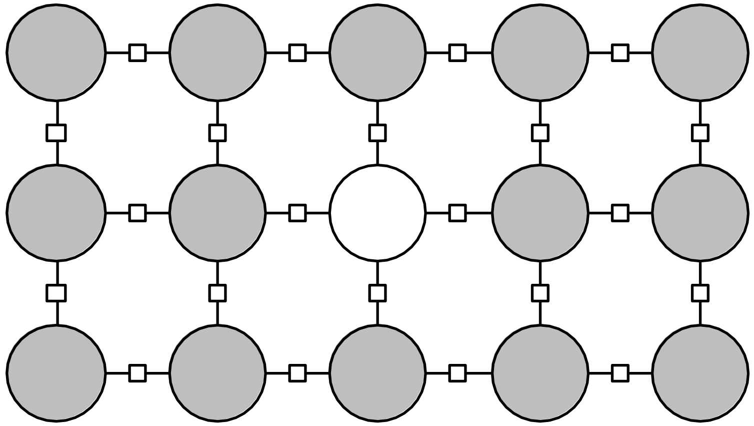
(notation: $X_{-i} = X \backslash \{X_i\}$)

Application: image denoising



Application: image denoising

Example: image denoising



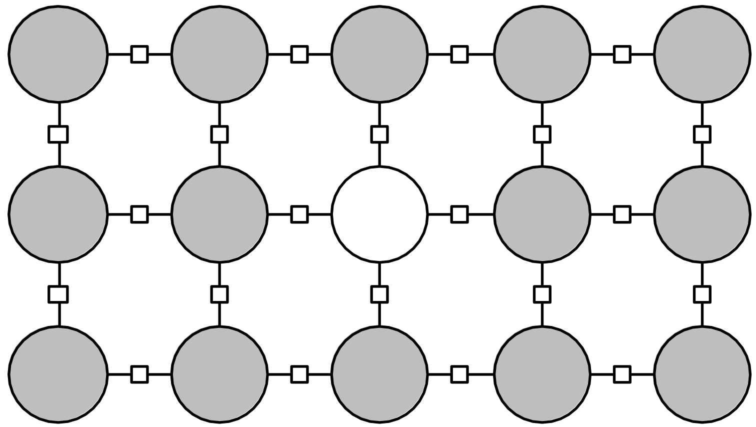
- $X_i \in \{0, 1\}$ is pixel value in location i
- Subset of pixels are observed

$o_i(x_i) = [x_i = \text{ observed value at } i]$

- Neighboring pixels more likely to be same than different

$t_{ij}(x_i, x_j) = [x_i = x_j] + 1$

Application: image denoising



Example: image denoising

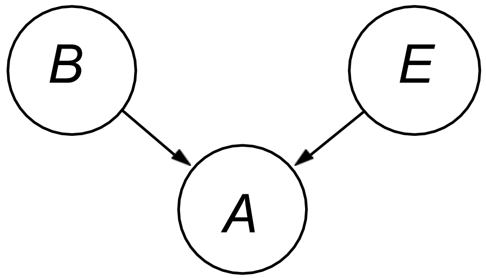
If neighbors are 1, 1, 1, 0 and X_i not observed:

$$P(X_i = 1 \mid X_{-i} = x_{-i}) = \frac{2\cdot2\cdot2\cdot1}{2\cdot2\cdot2\cdot1+1\cdot1\cdot1\cdot2} = 0.8$$

If neighbors are 0, 1, 0, 1 and X_i not observed:

$$P(X_i = 1 \mid X_{-i} = x_{-i}) = \frac{1\cdot2\cdot1\cdot2}{1\cdot2\cdot1\cdot2+2\cdot1\cdot2\cdot1} = 0.5$$

Where do parameters come from?



b	$p(b)$
1	?
0	?

e	$p(e)$
1	?
0	?

b	e	a	$p(a \mid b, e)$
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Learning task

Training data

D_{train} (an example is an assignment to X)

Parameters

θ (local conditional probabilities)

Example: one variable

Setup:

- One variable R representing the rating of a movie $\{1, 2, 3, 4, 5\}$

$$\text{P}(R = r) = p(r)$$

Parameters:

$$\theta = (p(1), p(2), p(3), p(4), p(5))$$

Training data:

$$D_{\text{train}} = \{1, 3, 4, 4, 4, 4, 4, 5, 5, 5\}$$

Example: one variable

Learning:

$$D_{train} \quad \Rightarrow \quad \theta$$

Intuition: $p(r) \propto$ number of occurrences of r in D_{train}

Example:

$$D_{train} = \{1, 3, 4, 4, 4, 4, 4, 5, 5, 5\}$$

$\downarrow$

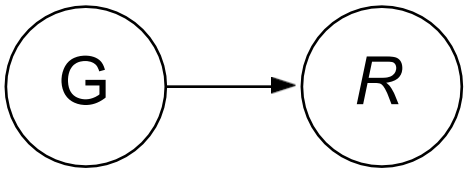
θ:

r	$p(r)$
1	0.1
2	0.0
3	0.1
4	0.5
5	0.3

Example: two variables

Variables:

- Genre $G \in \{\text{drama, comedy}\}$
- Rating $R \in \{1, 2, 3, 4, 5\}$

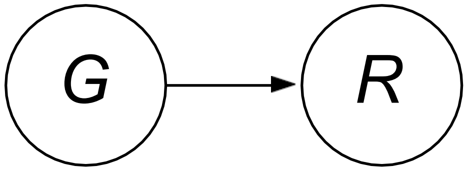


$$P(G = g, R = r) = p_G(g)\, p_R(r \mid g)$$

$D_{\text{train}} = \{(d, 4), (d, 4), (d, 5), (c, 1), (c, 5)\}$

Parameters: $\theta = (p_G, p_R)$

Example: two variables



$$\mathrm{P}(G = g, R = r) = p_G(g)\,p_R(r \mid g)$$

$$\mathrm{D}_{\mathrm{train}} = \{(\mathrm{d}, 4), (\mathrm{d}, 4), (\mathrm{d}, 5), (\mathrm{c}, 1), (\mathrm{c}, 5)\}$$

Intuitive strategy: Estimate each local conditional distribution (p_G and p_R) separately

θ:

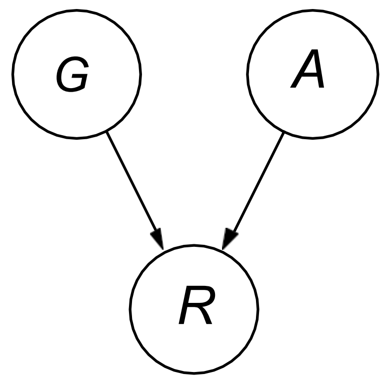
g	$p_G(g)$
d	3/5
c	2/5

g	r	$p_R(r \mid g)$
d	4	2/3
d	5	1/3
c	1	1/2
c	5	1/2

Example: v-structure

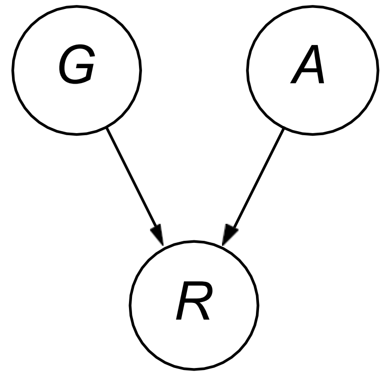
Variables:

- Genre $G \in$ {drama, comedy}
- Won award $A \in \{0, 1\}$
- Rating $R \in \{1, 2, 3, 4, 5\}$



$$P(G = g, A = a, R = r) = p_G(g)p_A(a)p_R(r \mid g, a)$$

Example: v-structure



D_{train} = {(d, 0, 3), (d, 1, 5), (c, 0, 1), (c, 0, 5), (c, 1, 4)}

Parameters: $\theta = (p_G, p_A, p_R)$

θ:

g	$p_G(g)$
d	3/5
c	2/5

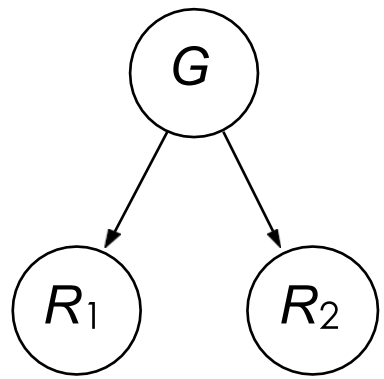
a	$p_A(a)$
0	3/5
1	2/5

g	a	r	$p_R(r \mid g, a)$
d	0	3	1
d	1	5	1
c	0	1	1/2
c	0	5	1/2
c	1	4	1

Example: inverted-v structure

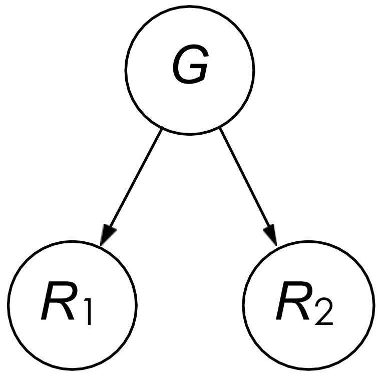
Variables:

- Genre $G \in \{\text{drama}, \text{comedy}\}$
- Jim's rating $R_1 \in \{1, 2, 3, 4, 5\}$
- Martha's rating $R_2 \in \{1, 2, 3, 4, 5\}$



$$P(G = g, R_1 = r_1, R_2 = r_2) = p_G(g) p_{R_1}(r_1 \mid g) p_{R_2}(r_2 \mid g)$$

Example: inverted-v structure



D_{train} = {(d, 4, 5), (d, 4, 4), (d, 5, 3), (c, 1, 2), (c, 5, 4)}

Parameters: $\theta = (p_G, p_{R_1}, p_{R_2})$

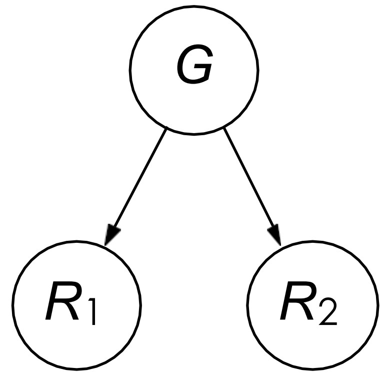
θ:

g	$p_G(g)$
d	3/5
c	2/5

g	r_1	$p_{R_1}(r \mid g)$
d	4	2/3
d	5	1/3
c	1	1/2
c	5	1/2

g	r_2	$p_{R_2}(r \mid g)$
d	3	1/3
d	4	1/3
d	5	1/3
c	2	1/2
c	4	1/2

Example: inverted-v structure



$D_{\text{train}} = \{(d, 4, 5), (d, 4, 4), (d, 5, 3), (c, 1, 2), (c, 5, 4)\}$

Parameters: $\theta = (p_G, p_R)$

θ:

g	$p_G(g)$
d	3/5
c	2/5

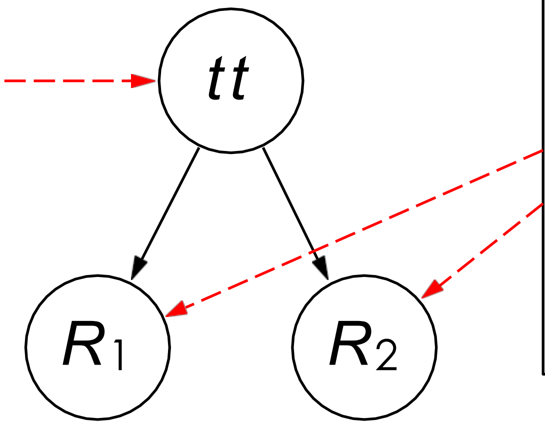
g	r	$p_R(r \mid g)$
d	3	1/6
d	4	3/6
d	5	2/6
c	1	1/4
c	2	1/4
c	4	1/4
c	5	1/4

Parameter sharing

Key idea: parameter sharing

The local conditional distributions of different variables use the same parameters.

g	$p_G(g)$
c	2/5
d	3/5



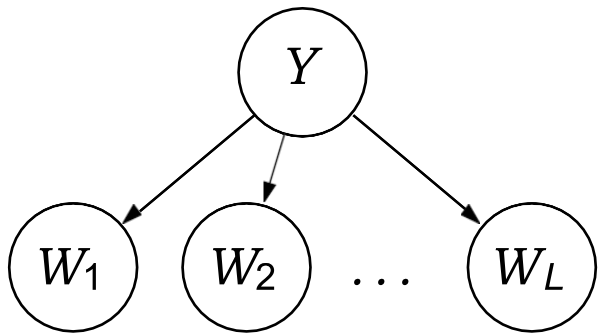
g	r	$p_R(r \mid g)$
d	3	1/6
d	4	3/6
d	5	2/6
c	1	1/4
c	2	1/4
c	4	1/4
c	5	1/4

Result: more reliable estimates, less expressive

Example: Naive Bayes

Variables:

- Genre $Y \in$ {comedy, drama}
- Movie review (sequence of words): $W_1, \ldots, W_L$



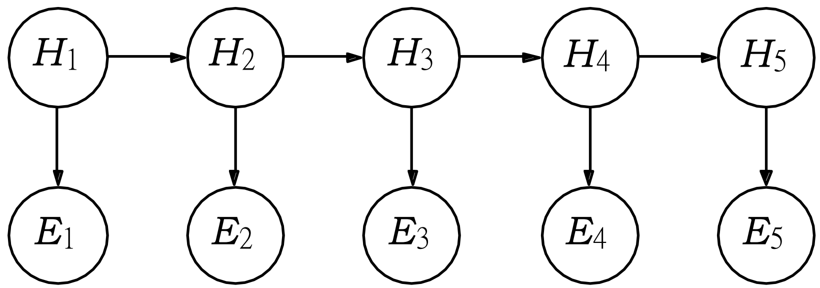
$$\mathbb{P}(Y = y, W_1 = w_1, \ldots, W_L = w_L) = p_{\text{genre}}(y) \prod_{j=1}^{L} p_{\text{word}}(w_j \mid y)$$

Parameters: $\theta = (p_{\text{genre}}, p_{\text{word}})$

Example: HMMs

Variables:

- $H_1, \ldots, H_n$ (e.g., actual positions)
- $E_1, \ldots, E_n$ (e.g., sensor readings)



$$\mathbb{P}(H = h, E = e) = p_{\text{start}}(h_1) \prod_{i=2}^{n} p_{\text{trans}}(h_i \mid h_{i-1}) \prod_{i=1}^{n} p_{\text{emit}}(e_i \mid h_i)$$

Parameters: $\theta = (p_{\text{start}}, p_{\text{trans}}, p_{\text{emit}})$

D_{train} is a set of full assignments to (H, E)

General case

Bayesian network: variables $X_1, \ldots, X_n$

Parameters: collection of distributions $\theta = \{p_d : d \in D\}$ (e.g., $D = \{\text{start}, \text{trans}, \text{emit}\}$)

Each variable X_i is generated from distribution p_{d_i}:

$$\mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} p_{d_i}(x_i \mid x_{\text{Parents}(i)})$$

Parameter sharing: d_i could be same for multiple i

General case: learning algorithm

Input: training examples D_{train} of full assignments

Output: parameters $\theta = \{p_d : d \in D\}$

Algorithm: maximum likelihood for Bayesian networks

Count:

For each $x \in D_{train}$:

For each variable x_i:

Increment $\text{count}_{d_i}(x_{\text{Parents}(i)}, x_i)$

Normalize:

For each d and local assignment $x_{\text{Parents}(i)}$:

Set $p_d(x_i \mid x_{\text{Parents}(i)}) \propto \text{count}_d(x_{\text{Parents}(i)}, x_i)$

Maximum likelihood

Maximum likelihood objective:

$$\max_{\theta} \prod_{x \in \mathcal{D}_{\mathsf{train}}} \mathbb{P}(X = x; \theta)$$

Algorithm on previous slide exactly computes maximum likelihood parameters (closed form solution).

Maximum likelihood

$D_{\text{train}} = \{(\text{d}, 4), (\text{d}, 5), (\text{c}, 5)\}$

$$\max_{p_G(\cdot)} (p_G(\text{d})p_G(\text{d})p_G(\text{c})) \max_{p_R(\cdot|\text{c})} p_R(5 \mid \text{c}) \max_{p_R(\cdot|\text{d})} (p_R(4 \mid \text{d})p_R(5 \mid \text{d}))$$

- Key: decomposes into subproblems, one for each distribution d and assignment x_{Parents}
- For each subproblem, solve in closed form (Lagrange multipliers for sum-to-1 constraint)

Roadmap

Supervised learning

Laplace smoothing

Unsupervised learning with EM

Scenario 1

Setup:

- You have a coin with an unknown probability of heads $p(\mathrm{H})$.
- You flip it 100 times, resulting in 23 heads, 77 tails.
- What is estimate of $p(\mathrm{H})$?

Maximum likelihood estimate:

$$p(\mathrm{H}) = 0.23 \quad p(\mathrm{T}) = 0.77$$

Scenario 2

Setup:

- You flip a coin once and get heads.
- What is estimate of $p(\text{H})$?

Maximum likelihood estimate:

$$p(\text{H}) = 1 \quad p(\text{T}) = 0$$

Intuition: This is a bad estimate; real $p(\text{H})$ should be closer to half

When have less data, maximum likelihood overfits, want a more reasonable estimate...

Regularization: Laplace smoothing

Maximum likelihood:

$$p(\mathrm{H}) = \frac{1}{1} \quad p(\mathrm{T}) = \frac{0}{1}$$

Maximum likelihood with Laplace smoothing:

$$p(\mathrm{H}) = \frac{1+1}{1+2} = \frac{2}{3} \quad p(\mathrm{T}) = \frac{0+1}{1+2} = \frac{1}{3}$$

Example: two variables

$D_{train} = \{(d, 4), (d, 5), (c, 5)\}$

Amount of smoothing: $\lambda = 1$

θ:

g	$p_G(g)$
d	3/5
c	2/5

g	r	$p_R(r \mid g)$
d	1	1/7
d	2	1/7
d	3	1/7
d	4	2/7
d	5	2/7
c	1	1/6
c	2	1/6
c	3	1/6
c	4	1/6
c	5	2/6

Regularization: Laplace smoothing

Key idea: Laplace smoothing

For each distribution d and partial assignment $(x_{\text{Parents}(i)}, x_i)$, add λ to $\text{count}_d(x_{\text{Parents}(i)}, x_i)$.

Then normalize to get probability estimates.

Interpretation: hallucinate λ occurrences of each local assignment

Larger λ $\Rightarrow$ more smoothing $\Rightarrow$ probabilities closer to uniform.

Data wins out in the end:

$$p(\text{H}) = \frac{1+1}{1+2} = \frac{2}{3} \qquad p(\text{H}) = \frac{998+1}{998+2} = 0.999$$

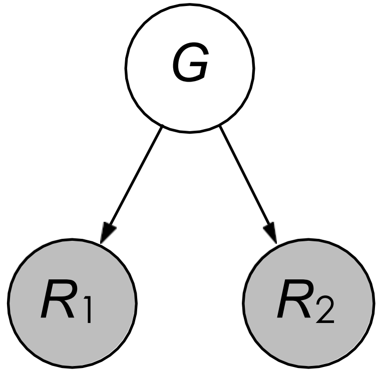
Roadmap

Supervised learning

Laplace smoothing

Unsupervised learning with EM

Motivation



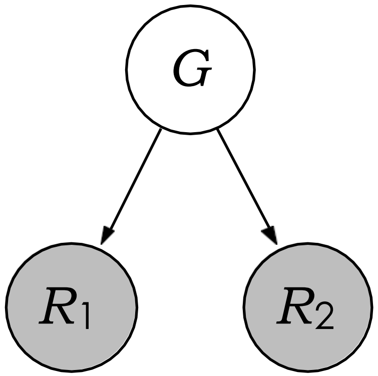
What if we **don't observe** some of the variables?

D_{train} = {(?, 4, 5), (?, 4, 4), (?, 5, 3), (?, 1, 2), (?, 5, 4)}

Maximum marginal likelihood

Variables: H is hidden, $E = e$ is observed

Example:



$H = G \quad E = (R_1, R_2) \quad e = (4, 5)$

$\theta = (p_G, p_R)$

Maximum marginal likelihood objective:

$$\max_{\theta} \prod_{e \in \mathcal{D}_{\text{train}}} \mathbb{P}(E = e; \theta)$$

$$= \max_{\theta} \prod_{e \in \mathcal{D}_{\text{train}}} \sum_{h} \mathbb{P}(H = h, E = e; \theta)$$

Expectation Maximization (EM)

Inspiration: K-means

Variables: H is hidden, E is observed (to be e)

Algorithm: Expectation Maximization (EM)

E-step:

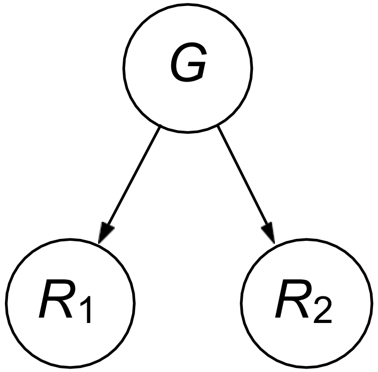
- Compute $q(h) = \mathrm{P}(H = h \mid E = e; \theta)$ for each h (use any probabilistic inference algorithm)
- Create weighted points: (h, e) with weight $q(h)$

M-step:

- Compute maximum likelihood (just count and normalize) to get θ

Repeat until convergence.

Example: one iteration of EM



$D_{train} = \{(?, 2, 2), (?, 1, 2)\}$

θ:

g	$p_G(g)$
c	0.5
d	0.5

g	r	$p_R(r \mid g)$
c	1	0.4
c	2	0.6
d	1	0.6
d	2	0.4

E-step

(r_1, r_2)	g	$P(G = g, R_1 = r_1, R_2 = r_2)$	$q(g)$
(2, 2)	c	$0.5 \cdot 0.6 \cdot 0.6$	$\frac{0.18}{0.18+0.08} = 0.69$
(2, 2)	d	$0.5 \cdot 0.4 \cdot 0.4$	$\frac{0.08}{0.18+0.08} = 0.31$
(1, 2)	c	$0.5 \cdot 0.4 \cdot 0.6$	$\frac{0.12}{0.12+0.12} = 0.5$
(1, 2)	d	$0.5 \cdot 0.6 \cdot 0.4$	$\frac{0.12}{0.12+0.12} = 0.5$

M-step

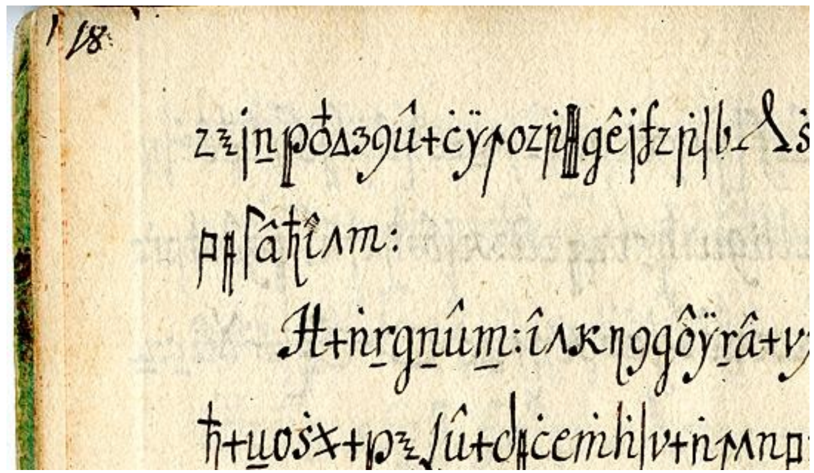
θ:

g	count	$p_G(g)$
c	0.69 + 0.5	0.59
d	0.31 + 0.5	0.41

g	r	count	$p_R(r \mid g)$
c	1	0.5	0.21
c	2	0.5 + 0.69 + 0.69	0.79
d	1	0.5	0.31
d	2	0.5 + 0.31 + 0.31	0.69

Copiale cipher (105-page encrypted volume from 1730s):

Application: decipherment



Substitution ciphers

Letter substitution table (unknown):

Plain:	a b c d e f g h i j k l m n o p q r s t u v w x y z
Cipher:	p l o k m i j n u h b y g v t f c r d x e s z a q w

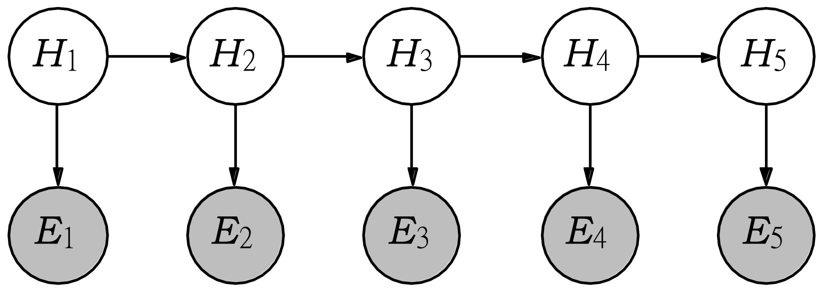
Plaintext (unknown): h e l l o w o r l d

Ciphertext (known): n m y y t z t r y k

Application: decipherment as an HMM

Variables:

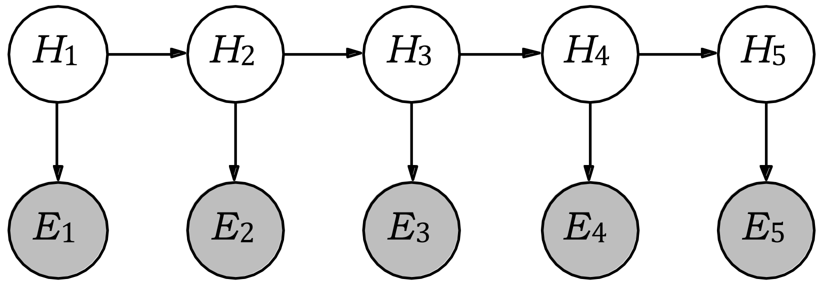
- $H_1, \ldots, H_n$ (e.g., characters of plaintext)
- $E_1, \ldots, E_n$ (e.g., characters of ciphertext)



$$\mathbb{P}(H = h, E = e) = p_{\text{start}}(h_1) \prod_{i=2}^{n} p_{\text{trans}}(h_i \mid h_{i-1}) \prod_{i=1}^{n} p_{\text{emit}}(e_i \mid h_i)$$

Parameters: $\theta = (p_{\text{start}}, p_{\text{trans}}, p_{\text{emit}})$

Application: decipherment as an HMM

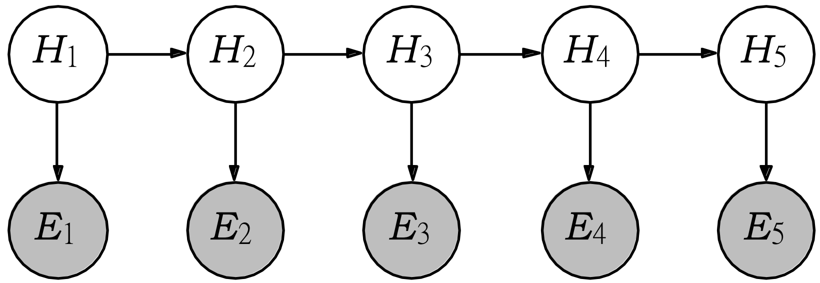


Strategy:

- p_{start}: set to uniform
- p_{trans}: estimate on tons of English text
- p_{emit}: **substitution table**, from EM

Intuition: rely on language model (p_{trans}) to favor plaintexts h that look like English

Application: decipherment as an HMM



E-step: forward-backward algorithm computes

$$q_i(h) \stackrel{\text{def}}{=} \mathbb{P}(H_i = h \mid E_1 = e_1, \dots E_n = e_n)$$

M-step: count (fractional) and normalize

$$\text{count}_{\text{emit}}(h, e) = \textstyle\sum_{i=1}^{n} q_i(h) \cdot [e_i = e]$$

$$p_{\text{emit}}(e \mid h) \propto \text{count}_{\text{emit}}(h, e)$$

Summary

- 概率模型
- 贝叶斯网络
- 概率程序
- 参数学习