Expected/average message length

The expected length of message is

$$E_p[|\mathrm{Enc}(M)|] = \sum_{m \in \{\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}\}} p(M = m)|\mathrm{Enc}(m)|$$

Since $|Enc(m)| = 2$ for all m, on average for each message you will use ... 2 bits

Entropy

You can show that the codeword length assignment that minimizes the expected message length is $-\log_2 p(m)$.

Given a random variable M distributed according to p entropy is

$$H(M) = \sum_m p(M = m) \left[-\log_2 p(M = m)\right]$$

The entropy can be interpreted as number bits spent in communication using the optimal codebook.

For simplicity, we may write $H(p)$ when it is clear which random variable we are considering.

Conditional entropy

$$H(X) = \sum_m p(X = m) \left[-\log_2 p(M = m) \right]$$

If we have access to extra information Y then we can compute conditional entropy

$$H(X|Y) = \sum_y \sum_x p(X = x|Y = y) \left[-\log_2 p(X = x|Y = y) \right]$$

Entropy – characteristics

$$H(p) = \sum_m p(m) \left[-\log_2 p(m) \right]$$

Entropy is always non-negative

The more uniform the distribution the higher the entropy (I need 2 bits for 4 messages with prob. $1/4$).

The less uniform the distribution the lower the entropy (I need 0 bits if I always send the same message).

Entropy recap

- Entropy is evaluated on probability distributions

$$H(p) = -\sum_m p(M = m) \log_2 p(M = m)$$

- Given a distribution of messages, entropy tells us the least number of bits we would need to encode messages to communicate efficiently.
- Length of a code for a message should be $-\log_2 p(m)$. Sometimes this can not be achieved.[2]
- Entropy is maximized for uniform distributions

$$H([\frac{1}{K} \dots \frac{1}{K}]) = \log_2 K$$

[2] If $p(m) \neq 2^{-c}$ we get fractional code lengths.

Cross-entropy

Suppose we were given message probabilities q.

We build our codebook based on q and the optimal average message length is $H(q)$.

But it turns out the q is incorrect and the true message probabilities are p.

Cross-entropy is the average message length under these circumstances

$$H(p, q) = -\sum_{m} p(m) \log q(m)$$

Cross-entropy

Cross entropy

$$H(p, q) = -\sum_m p(m) \log q(m)$$

is always greater or equal than entropy

$$H(p) = -\sum_m p(m) \log p(m).$$

Using wrong codebook is always incurs cost in communication – using longer codes for less frequent messages.

Kulback Leibler divergence

Had we known the true distribution p our average message length would be

$$H(p) = -\sum_m p(m) \log p(m)$$

we didn't and we are now on average using a *longer* message

$$H(p, q) = -\sum_m p(m) \log q(m).$$

How much longer?

$$H(p, q) - H(p) = -\sum_m p(m) \log q(m) + \sum_m p(m) \log p(m)$$

This difference is called Kullback-Leibler divergence

$$\mathrm{KL}(p||q) = H(p, q) - H(p) = -\sum_m p(m) \log q(m) + \sum_m p(m) \log p(m)$$

KL divergence

For discrete pdfs:

$$\mathrm{KL}(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

For continuous pdfs:

$$\mathrm{KL}(p||q) = \int_x p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x$$

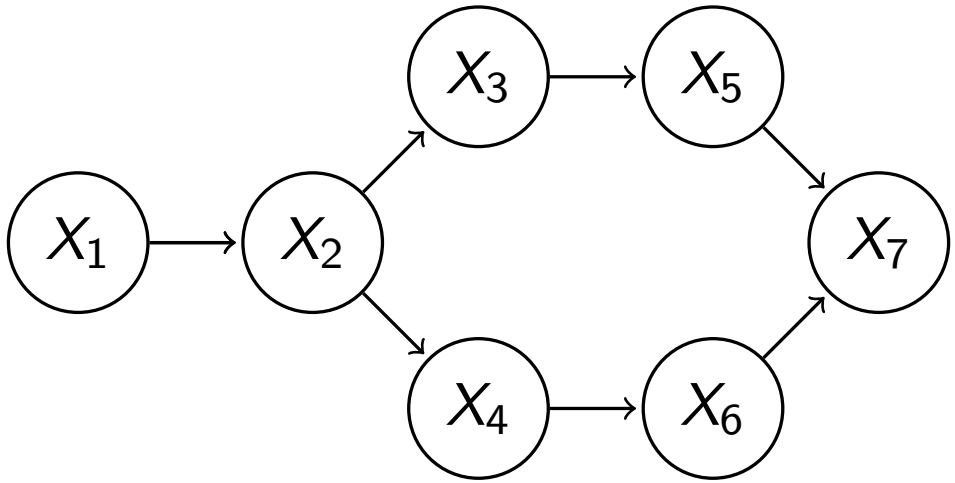
A couple of observations about KL-divergence

1. $\mathrm{KL}(p||q) \geq 0$
2. $\mathrm{KL}(p||q) = 0$ if and only if $p = q$
3. it is not symmetric $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$

Summary

- Entropy as a measurement of the number of bits needed to communicate efficiently.
- KL-divergence as a number of bits that could be saved by using the right distribution
- KL-divergence as a distance between distributions.

Specifying a graphical model



This graphical model specifies a joint distribution

$$
\begin{aligned}
p(X_1, X_2, X_3, X_4, X_5, X_6, X_7) &= \prod_i p(X_i | X_{\mathbf{pa}(i)}) \\
&= p(X_1)p(X_2|X_1)p(X_3|X_2)p(X_4|X_2) \\
&\quad p(X_5|X_3)p(X_6|X_4)p(X_7|X_5, X_6)
\end{aligned}
$$

Determining conditional independencies from graphs

In the topological order of nodes of a DAG, parent nodes precede child nodes. There can be many topological orders.

Given an order O, let $\mathbf{pnp}_O(i)$ denote a set of nodes that precede node i in a topological order but are not its parents.

We can show that

$$X_i \perp X_{\mathbf{pnp}_O(i)} | X_{\mathbf{pa}(i)}$$

These are basic conditional independence relationships.

Verifying a conditional independence

We want to show

$$p(X_3|X_2, X_1) = \frac{p(X_3, X_2, X_1)}{p(X_2, X_1)} = p(X_3|X_1)$$

Marginals are

$$\begin{aligned} p(X_3, X_2, X_1) &= p(X_1)p(X_2|X_1)p(X_3|X_1) \\ p(X_2, X_1) &= p(X_1)p(X_2|X_1) \end{aligned}$$

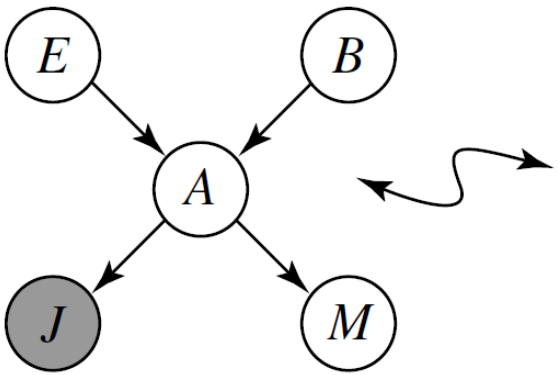
plugging them in we get

$$p(X_3|X_2, X_1) = \frac{p(X_3, X_2, X_1)}{p(X_2, X_1)} = \frac{p(X_1)p(X_2|X_1)p(X_3|X_1)}{p(X_1)p(X_2|X_1)} = p(X_3|X_1)$$

and this confirms $X_3 \perp X_2|X_1$

Representing evidence in Bayesian networks

- When we condition on some of the variables, the result is a conditional density that can have different independence properties.
- When we condition on a variable, we shade the corresponding node in the Bayes net.



$$p_{EBAM|J} = \frac{p_E \times p_B \times p_{A|EB} \times p_{J|A} \times p_{M|A}}{\sum_e \sum_b \sum_a \sum_m p_E \times p_B \times p_{A|EB} \times p_{J|A} \times p_{M|A}}$$

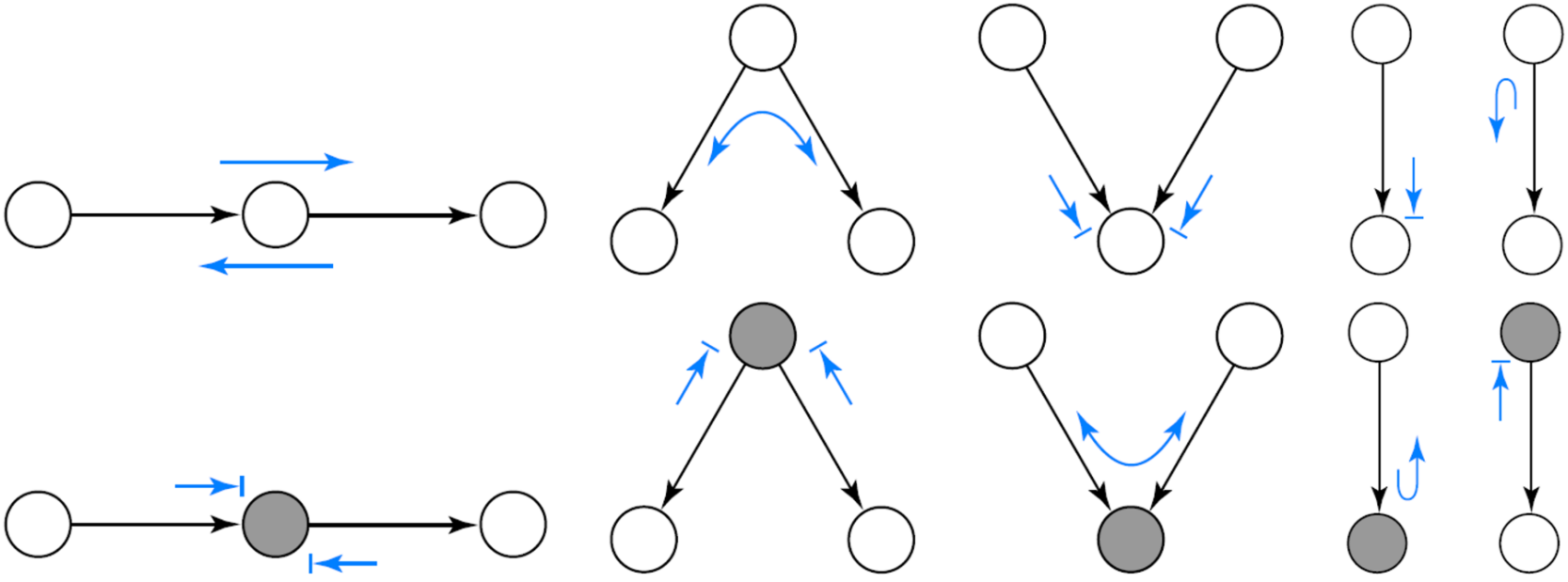
Encoding conditional independence via d-separation

- Bayesian networks encode the independence properties of a density.
- We can determine if a conditional independence $X \perp Y | \mathcal{Z}$ holds by appealing to a graph separation criterion called *d-separation* (*direction-dependent separation*).
- X and Y are d-separated if there is no active path between them.
- The formal definition of active paths is somewhat involved. The *Bayes Ball Algorithm* gives a nice graphical definition.

Bayes Ball Algorithm

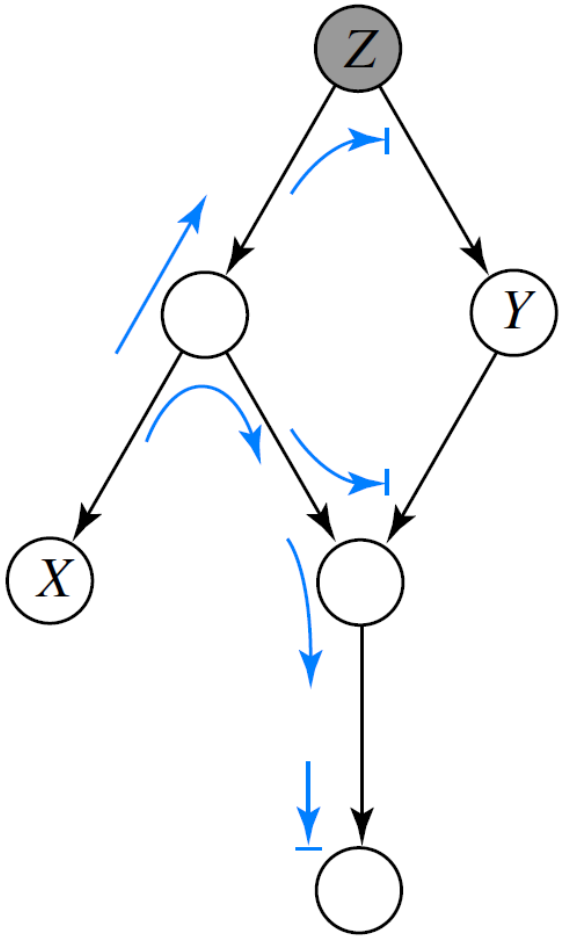
- You want to check if $X \perp Y | \mathcal{Z}$. Imagine passing a " from a node to a node, if the ball can make it from X to Y they are dependent. Shade the nodes in $\mathcal{Z}$ and apply following rules:

An Undirected path P is active if a Bayes ball travelling along it never encounters the "stop" symbol



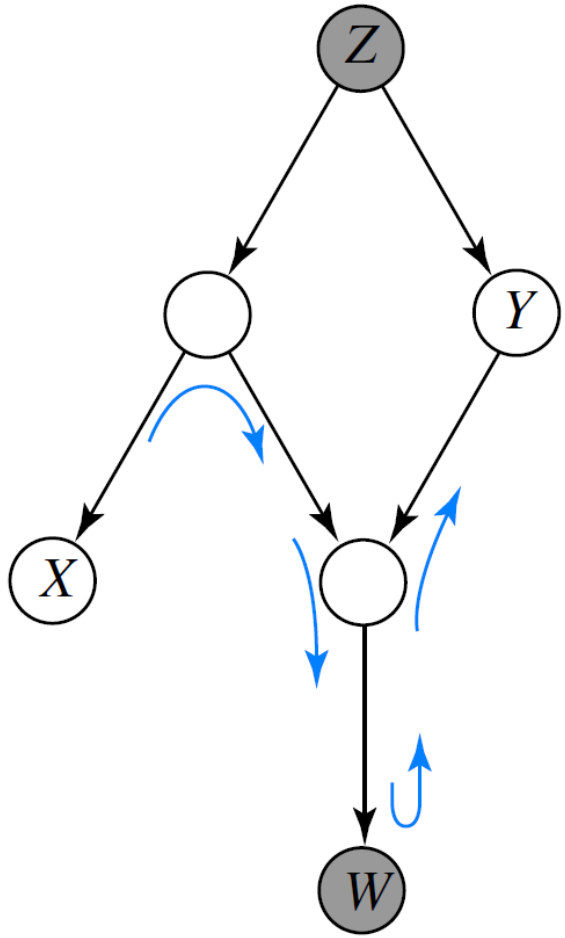
P Contains a Chain | **P Contains a fork** | **P Contains a V-structure** | **Boundary Conditions**

A double-header: two games of Bayes Ball



no active paths

$X \perp\!\!\!\perp Y \mid Z$



one active path

$X \not\perp\!\!\!\perp Y \mid \{W, Z\}$

Learning Bayesian Networks

Learning a Bayesian Network requires us to determine

- network's structure
- network's parameters

You will implement both of these in your HW2.

Learning a BayesNet

Probability of a state of all n variables $\mathbf{x}$

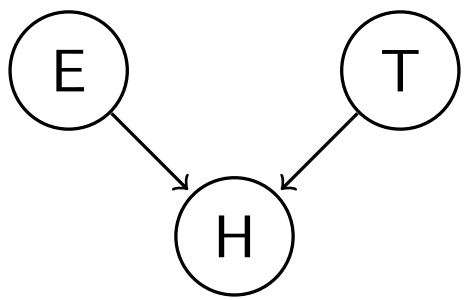
$$p(X_1 = x_1, ..., X_n = x_n) = \prod_{j=1}^{n} p(X_j = x_j | X_{\mathbf{pa}(j)} = x_{\mathbf{pa}(j)}, \theta_j)$$

Our goal is to learn parameters $\Theta = (\theta_1, ...\theta_n)$ from multiple samples of the state of the Bayes net.

Example

Data:

Sample \ Variable	Earthquake	Truck	House moved
1	1	0	1
2	0	0	0
⋮	⋮	⋮	⋮
S	0	1	0



Conditional distributions and their parameters

- $p(E = 1|\theta_E) = \theta_E$, where $\theta_E \in [0, 1]$
- $p(T = 1|\theta_T) = \theta_T$, where $\theta_R \in [0, 1]$
- $p(H = 1|E = e, T = t, \theta_H) = \theta_{H,e,t}$, where $\theta_{H,e,t} \in [0, 1]$

Learning a BayesNet – likelihood

Data S instances of Bayes net's state; $x_{i,j}$ state of variable X_j in i^{th} sample.

$$L(\Theta) = \underbrace{\prod_{i=1}^{S}}_{\text{samples}} \underbrace{\prod_{j=1}^{n}}_{\text{variables}} p(X_j = x_{i,j} | X_{\mathbf{pa}(j)} = x_{i,\mathbf{pa}(j)}, \theta_j)$$

Likelihood Example

Data:

Sample \ Variable	Earthquake	Truck	House moved
1	1	0	1
2	0	0	0
⋮	⋮	⋮	⋮
S	0	1	0

$$
\begin{aligned}
L(\Theta) \quad = \quad & p(E=1|\theta_E)p(T=0|\theta_T)p(H=1|E=1,T=0,\theta_H) \\
& p(E=0|\theta_E)p(T=0|\theta_T)p(H=0|E=0,T=0,\theta_H) \\
& \vdots \\
& p(E=0|\theta_E)p(T=1|\theta_T)p(H=0|E=0,T=1,\theta_H)
\end{aligned}
$$

Likelihood Example – continued

$$
\begin{aligned}
L(\Theta) &= p(E=1|\theta_E)p(T=0|\theta_T)p(H=1|E=1,T=0,\theta_H) \\
&\times p(E=0|\theta_E)p(T=0|\theta_T)p(H=0|E=0,T=0,\theta_H) \\
&\times \vdots \\
&\times p(E=0|\theta_E)p(T=1|\theta_T)p(H=0|E=0,T=1,\theta_H)
\end{aligned}
$$

In terms of parameters

$$
\begin{aligned}
L(\Theta) &= \theta_E & (1-\theta_T) & \theta_{H,1,0} \\
&\times (1-\theta_E) & (1-\theta_T) & (1-\theta_{H,0,0}) \\
&\times \vdots & & \\
&\times (1-\theta_E) & \theta_T & (1-\theta_{H,1,0})
\end{aligned}
$$

Learning a BayesNet – log-likelihood

Log-likelihood is given by

$$\mathrm{LL}(\Theta) = \underbrace{\sum_{i=1}^{S}}_{\text{samples}} \underbrace{\sum_{j=1}^{n}}_{\text{variables}} \log p(X_j = x_{i,j} | X_{\mathbf{pa}(j)} = x_{i,\mathbf{pa}(j)}, \theta_j)$$

Crucial observation:

$$\mathrm{LL}(\Theta) = \underbrace{\sum_{j=1}^{n}}_{\text{variables}} \underbrace{\sum_{i=1}^{S}}_{\text{samples}} \log p(X_j = x_{i,j} | X_{\mathbf{pa}(j)} = x_{i,\mathbf{pa}(j)}, \theta_j)$$

Learning a BayesNet – maximizing log-likelihood

$$\begin{aligned}
\underset{\theta_j}{\operatorname{argmax}}\, \mathrm{LL}(\Theta) &= \underset{\theta_j}{\operatorname{argmax}} \sum_{j=1}^{n} \sum_{i=1}^{S} \log p(X_j = x_{i,j} | X_{\mathbf{pa}(j)} = x_{i,\mathbf{pa}(j)}, \theta_j) \\
&= \underset{\theta_j}{\operatorname{argmax}} \sum_{i=1}^{S} \log p(X_j = x_{i,j} | X_{\mathbf{pa}(j)} = x_{i,\mathbf{pa}(j)}, \theta_j)
\end{aligned}$$

To learn parameters θ_j we only need states of node j and its parents.

Further, we do not need to concern ourselves with the rest of the graph.

Learning BayesNets – structure learning

If we assume that BayesNet is a tree – each variable has a single parent – we can derive an algorithm to discover the optimal structure.
Given an empirical distribution of the data f we wish to find optimal distribution with factorization

$$p(X_1, ..., X_n) = \prod_{j=1}^{n} p(X_j | X_{\mathbf{pa}(j)})$$

where $\mathbf{pa}(j)$ has at most one element for all j.
Under this assumption, we can define edge set

$$\text{Edges} = \{(j, \mathbf{pa}(j)) | j = 1, ..., n\}$$

Learning BayesNets – structure learning

Writing out log-likelihood

$$
\begin{aligned}
\mathrm{LL}(\Theta) &= \sum_{i=1}^{S} \sum_{j} \log p(x_{i,j} | x_{i,\mathbf{pa}(j)}) \\
&= \sum_{i=1}^{S} \sum_{j=1}^{n} \sum_{k=1}^{n} [(j,k) \in \mathrm{Edges}] \log p(x_{i,j} | x_{i,k}) \\
&= \sum_{j=1}^{n} \sum_{k=1}^{n} [(j,k) \in \mathrm{Edges}] \left(\sum_{i=1}^{S} \log p(x_{i,j} | x_{i,k}) \right) \\
&= \sum_{(j,k) \in \mathrm{Edges}} \left(\sum_{i=1}^{S} \log p(x_{i,j} | x_{i,k}) \right)
\end{aligned}
$$

Learning BayesNets – structure learning

$$\mathrm{LL}(\Theta) = \sum_{(j,k)\in \mathrm{Edges}} \left(\sum_{i=1}^{S} \log p(x_{i,j}|x_{i,k}) \right)$$

We observed that we can learn optimal $p(X_j|X_k, \theta_j)$ independently of the graph structure.

$$\theta_j^* = \operatorname*{argmax}_{\theta_j} \sum_{i=1}^{S} \log p(x_{i,j}|x_{i,k}, \theta_j)$$

Learning BayesNets – structure learning

$$\mathrm{LL}(\Theta) \sum_{(j,k)\in \mathrm{Edges}} \underbrace{\sum_{i=1}^{S} \log p(x_{i,j}|x_{i,k},\theta_j^*)}_{\text{weight of edge (j,k)}}$$

Hence, we wish to find a tree with maximum weight, where each edge's weight is

$$w_{j,k} = \sum_{i=1}^{S} \log p(X_j = x_{i,j}|X_k = x_{i,k},\theta_j^*)$$

Chow-Liu tree learning algorithm

Observe that

$$\sum_{i=1}^{S} \log p(X_j = x_{i,j} | X_k = x_{i,k}) = I(X_j, X_k) - H(X_j)$$

and since $H(X_j)$ does not depend on the edges we can rewrite the problem as

$$\underset{\text{Edges}}{\operatorname{argmax}} \sum_{(j,k) \in Edges} I(X_j, X_k) - H(X_j) = \underset{\text{Edges}}{\operatorname{argmax}} \sum_{(j,k) \in Edges} I(X_j, X_k)$$

Note that this is an example of **maximum spanning tree** problem which can be solved efficiently – HW2.

Chow-Liu tree learning algorithm – details

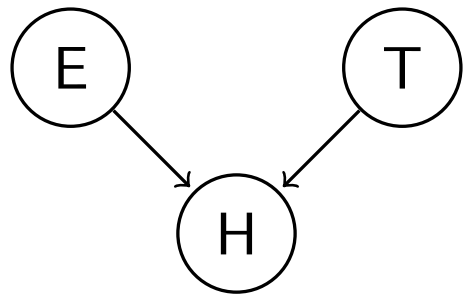
Mutual information $I(X_j, X_k)$ is computed on the empirical distribution

$$
\begin{aligned}
f_{j,k}(a,b) &= \frac{\sum_{i=1}^{S}[X_j = a, X_k = b]}{S} \\
f_l(a) &= \frac{\sum_{i=1}^{S}[X_l = a]}{S} \\
I(X_j, X_k) &= \sum_a \sum_b f_{j,k}(a,b) \log \frac{f_{j,k}(a,b)}{f_j(a) f_k(b)}
\end{aligned}
$$

Chow-Liu tree – downsides

Single parent assumptions can be restrictive.

The tree models do not permit explaining away.



Tree structured Bayes nets are step above independent models.

On the upside, they are extremely easy to learn.

Maximum likelihood and KL minimization

Delta function or indicator function:

$$[x] = \begin{cases} 1, \text{if } x \text{ is true} \\ 0, \textit{otherwise} \end{cases}$$

Data/Empirical distribution:

$$f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} [\mathbf{x}_i = \mathbf{x}]$$

Importantly:

$$\int_{\mathbf{x}} f(x) g(x) \mathrm{d}\mathbf{x} = \int_{\mathbf{x}} \left(\sum_{i=1}^{N} [\mathbf{x}_i = \mathbf{x}] \right) g(x) \mathrm{d}\mathbf{x} = \frac{1}{N} \sum_{i=1}^{N} g(x_i)$$

Maximizing likelihood is equivalent to minimizing KL

$$\mathrm{ALL}(\theta) = \sum_{i=1}^{N} \frac{1}{N} \log p(\mathbf{x}_i|\theta)$$

KL divergence between empirical distribution $f(\mathbf{x})$ and $p(\mathbf{x}|\theta)$:

$$\begin{aligned}
\mathrm{KL}(f||p) &= \int_{\mathbf{x}} f(\mathbf{x}) \log \frac{f(\mathbf{x})}{p(\mathbf{x})} \mathrm{d}\mathbf{x} \\
&= \int_{\mathbf{x}} \left(\frac{1}{N} \sum_{i=1}^{N} [\mathbf{x}_i = \mathbf{x}] \right) \log \frac{f(\mathbf{x})}{p(\mathbf{x})} \mathrm{d}\mathbf{x} \\
&= \sum_{i=1}^{N} \frac{1}{N} \log \frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)} = \underbrace{-\sum_{i=1}^{N} \frac{1}{N} \log p(\mathbf{x}_i)}_{-\mathrm{ALL}(\theta)} + \underbrace{\frac{1}{N} \log \frac{1}{N}}_{\text{const.}}
\end{aligned}$$

Maximizing likelihood is equivalent to minimizing KL

$$\mathrm{KL}(f||p) = -\mathrm{ALL}(\theta)$$

and hence

$$\operatorname*{argmin}_{\theta} \mathrm{KL}(f||p) = \operatorname*{argmax}_{\theta} \mathrm{ALL}(\theta) = \operatorname*{argmax}_{\theta} \mathrm{LL}(\theta)$$

Mutual information

Mutual information between random variables is defined as

$$I(X;Y) = \mathrm{KL}(p(X,Y)||p(X)p(Y)) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

Mutual information can be expressed as a difference of entropy and conditional entropy.

$$I(X;Y) = H(X) - H(X|Y)$$

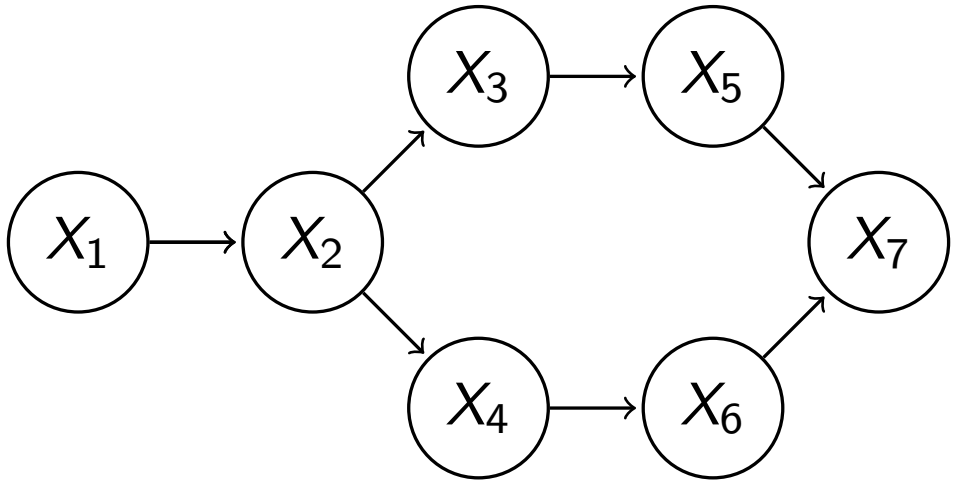
Note that mutual information is symmetric

$$I(X;Y) = I(Y;X) = H(Y) - H(Y|X)$$

We will use mutual information your homework to learn Bayesian networks.

Specifying a graphical model

Each of these nodes corresponds to a random variable.



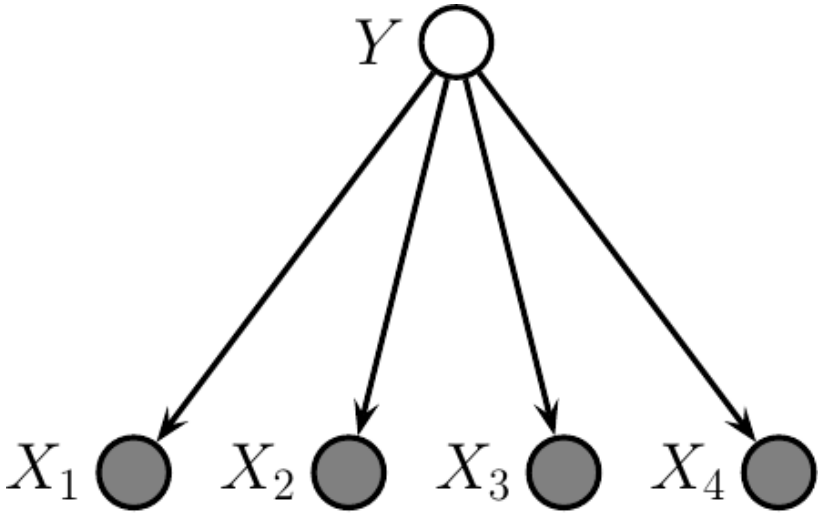
and each node has a conditional probability associated with it

$$p(X_i|X_{\mathbf{pa}(i)})$$

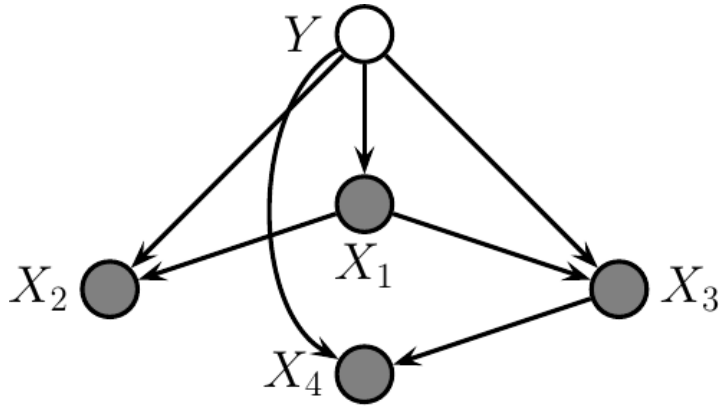
where $\mathbf{pa}(i)$ is a list of parent nodes of node i, e.g.

$$p(X_7|X_5, X_6)$$

Example: naive Bayes classifiers as Bayesian networks



A naive Bayes classifier represented as a directed graphical model. We assume there are D = 4 features, for simplicity. Shaded nodes are observed, unshaded nodes are hidden.



Tree-augmented naive Bayes (TAN) classifier for D = 4 features. In general, the tree topology can change depending on the value of y.

$$p(y, \mathbf{x}) = p(y) \prod_{j=1}^{D} p(x_j|y)$$