Probabilistic Graphical Models

Andrea Passerini
passerini@disi.unitn.it

Artificial Intelligence for Bioinformatics

Discrete random variables

Probability mass function

Given a discrete random variable X taking values in $\mathcal{X} = \{v_1, \ldots, v_m\}$, its *probability mass function* $P : \mathcal{X} \rightarrow [0, 1]$ is defined as:

$$P(v_i) = \Pr[X = v_i]$$

and satisfies the following conditions:

- $P(x) \geq 0$
- $\sum_{x \in \mathcal{X}} P(x) = 1$

Probability distributions

Bernoulli distribution

- Two possible values (outcomes): 1 (success), 0 (failure).
- Parameters: p probability of success.
- Probability mass function:

$$P(x; p) = \begin{cases} p & \text{if } x = 1 \\ 1 - p & \text{if } x = 0 \end{cases}$$

Example: tossing a coin

- Head (success) and tail (failure) possible outcomes
- p is probability of head

Probability distributions

Multinomial distribution (one sample)

- Models the probability of a certain outcome for an event with m possible outcomes $\{v_1, \dots, v_m\}$
- Parameters: $p_1, \dots, p_m$ probability of each outcome
- Probability mass function:

$$P(v_i; p_1, \dots, p_m) = p_i$$

Tossing a dice

- m is the number of faces
- p_i is probability of obtaining face i

Continuouos random variables

Probability density function

Instead of the probability of a specific value of X, we model the probability that x falls in an interval (a, b):

$$\Pr[x \in (a, b)] = \int_a^b p(x)dx$$

Properties:

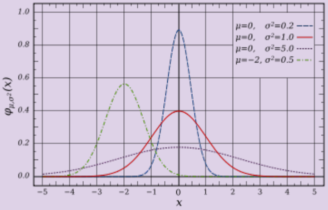
- $p(x) \geq 0$
- $\int_{-\infty}^{\infty} p(x)dx = 1$

Note

The probability of a specific value x_0 is given by:

$$p(x_0) = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \Pr[x \in [x_0, x_0 + \epsilon)]$$

Gaussian (or normal) distribution



- Bell-shaped curve.
- Parameters: μ mean, σ^2 variance.
- Probability density function:

$$p(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} \exp -\frac{(x-\mu)^2}{2\sigma^2}$$

- Standard normal distribution: $N(0,1)$
- Standardization of a normal distribution $N(\mu,\sigma^2)$

$$z = \frac{x-\mu}{\sigma}$$

Conditional probabilities

conditional probability probability of x once y is observed

$$P(x|y) = \frac{P(x,y)}{P(y)}$$

statistical independence variables X and Y are statistical independent iff

$$P(x,y) = P(x)P(y)$$

implying:

$$P(x|y) = P(x) \qquad P(y|x) = P(y)$$

Basic rules

law of total probability The *marginal distribution* of a variable is obtained from a joint distribution summing over all possible values of the other variable (*sum rule*)

$$P(x) = \sum_{y \in \mathcal{Y}} P(x, y) \qquad P(y) = \sum_{x \in \mathcal{X}} P(x, y)$$

product rule conditional probability definition implies that

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

Bayes' rule

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Playing with probabilities

Use rules!

- Basic rules allow to model a certain probability given knowledge of some related ones
- All our manipulations will be applications of the three basic rules
- Basic rules apply to any number of varables:

$$
\begin{aligned}
P(y) &= \sum_x \sum_z P(x,y,z) \quad \text{(sum rule)} \\
&= \sum_x \sum_z P(y|x,z)P(x,z) \quad \text{(product rule)} \\
&= \sum_x \sum_z \frac{P(x|y,z)P(y|z)P(x,z)}{P(x|z)} \quad \text{(Bayes rule)}
\end{aligned}
$$

Playing with probabilities

Example

$$
\begin{aligned}
P(y|x,z) &= \frac{P(x,z|y)P(y)}{P(x,z)} \quad \text{(Bayes rule)} \\
&= \frac{P(x,z|y)P(y)}{P(x|z)P(z)} \quad \text{(product rule)} \\
&= \frac{P(x|z,y)P(z|y)P(y)}{P(x|z)P(z)} \quad \text{(product rule)} \\
&= \frac{P(x|z,y)P(z,y)}{P(x|z)P(z)} \quad \text{(product rule)} \\
&= \frac{P(x|z,y)P(y|z)P(z)}{P(x|z)P(z)} \quad \text{(product rule)} \\
&= \frac{P(x|z,y)P(y|z)}{P(x|z)}
\end{aligned}
$$

Graphical models

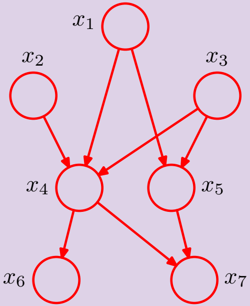
Why

- All probabilistic inference and learning amount at repeated applications of the sum and product rules
- *Probabilistic graphical models* are graphical representations of the *qualitative* aspects of probability distributions allowing to:
  - visualize the structure of a probabilistic model in a simple and intuitive way
  - discover properties of the model, such as conditional independencies, by inspecting the graph
  - express complex computations for inference and learning in terms of graphical manipulations
  - represent multiple probability distributions with the same graph, abstracting from their quantitative aspects (e.g. discrete vs continuous distributions)

Bayesian Networks (BN)

BN Semantics

- A BN structure ($\mathcal{G}$) is a *directed graphical model*
- Each node represents a random variable x_i
- Each edge represents a direct dependency between two variables



The structure encodes these independence assumptions:

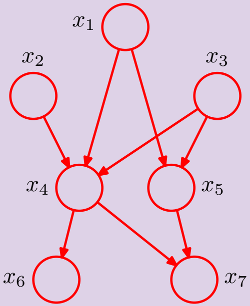
$$\mathcal{I}_\ell(\mathcal{G}) = \{\forall i \; x_i \perp NonDescendants_{x_i} | Parents_{x_i}\}$$

each variable is independent of its non-descendants given its parents

Graphs and Distributions

- Let *p* be a joint distribution over variables $\mathcal{X}$
- Let $\mathcal{I}(p)$ be the set of independence assertions holding in *p*
- $\mathcal{G}$ in as *independency map* (I-map) for *p* if *p* satisfies the local independences in $\mathcal{G}$:

$$\mathcal{I}_\ell(\mathcal{G}) \subseteq \mathcal{I}(p)$$



Note

The reverse is not necessarily true: there can be independences in *p* that are not modelled by $\mathcal{G}$.

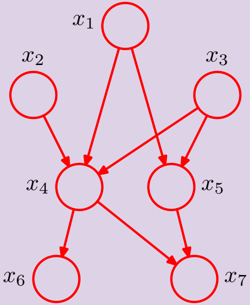
Bayesian Networks

Factorization

- We say that p *factorizes* according to $\mathcal{G}$ if:

$$p(x_1, \ldots, x_m) = \prod_{i=1}^{m} p(x_i | Pa_{x_i})$$

- If $\mathcal{G}$ is an I-map for p, then p factorizes according to $\mathcal{G}$
- If p factorizes according to $\mathcal{G}$, then $\mathcal{G}$ is an I-map for p



Example

$$\begin{aligned} p(x_1, \ldots, x_7) = & p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) \\ & p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5) \end{aligned}$$

Bayesian Networks

Definition

A Bayesian Network is a pair $(\mathcal{G}, p)$ where p factorizes over $\mathcal{G}$ and it is represented as a set of conditional probability distributions (cpd) associated with the nodes of $\mathcal{G}$.

Factorized Probability

$$p(x_1, \dots, x_m) = \prod_{i=1}^{m} p(x_i | Pa_{x_i})$$

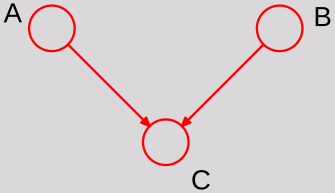
Bayesian Networks

Example: toy regulatory network

- Genes *A* and *B* have independent prior probabilities
- Gene *C* can be enhanced by both *A* and *B*

gene	value	P(value)
A	`active`	0.3
A	`inactive`	0.7

gene	value	P(value)
B	`active`	0.3
B	`inactive`	0.7



		A			
		`active`		`inactive`	
		B		B	
		`active`	`inactive`	`active`	`inactive`
C	`active`	0.9	0.6	0.7	0.1
C	`inactive`	0.1	0.4	0.3	0.9

Conditional independence

Introduction

- Two variables a, b are conditionally independent (written $a \perp\!\!\!\perp b \,|\, \emptyset$) if:

$$p(a, b) = p(a)p(b)$$

- Two variables a, b are conditionally independent given c (written $a \perp\!\!\!\perp b \,|\, c$) if:

$$p(a, b|c) = p(a|c)p(b|c)$$

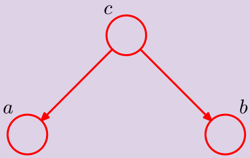
- Independency assumptions can be verified by repeated applications of sum and product rules
- Graphical models allow to directly verify them through the *d-separation* criterion

d-separation

Tail-to-tail

- Joint distribution:

$$p(a, b, c) = p(a|c)p(b|c)p(c)$$

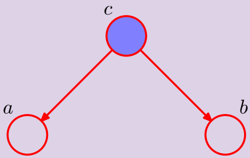


- a and b are **not conditionally independent** (written $a \not\perp\!\!\!\perp b \,|\, \emptyset$):

$$p(a, b) = \sum_c p(a|c)p(b|c)p(c) \neq p(a)p(b)$$

- a and b are **conditionally independent given** c:

$$p(a, b|c) = \frac{p(a, b, c)}{p(c)} = p(a|c)p(b|c)$$

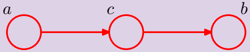


- c is *tail-to-tail* wrt to the path $a \to b$ as it is connected to the tails of the two arrows

d-separation

Head-to-tail

- Joint distribution:

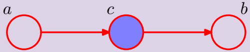


$$p(a, b, c) = p(b|c)p(c|a)p(a) = p(b|c)p(a|c)p(c)$$

- a and b are **not conditionally independent**:

$$p(a, b) = p(a) \sum_c p(b|c)p(c|a) \neq p(a)p(b)$$

- a and b are **conditionally independent given** c:

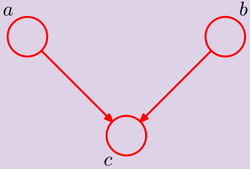


$$p(a, b|c) = \frac{p(b|c)p(a|c)p(c)}{p(c)} = p(b|c)p(a|c)$$

- c is *head-to-tail* wrt to the path $a \rightarrow b$ as it is connected to the head of an arrow and to the tail of the other one

d-separation

Head-to-head

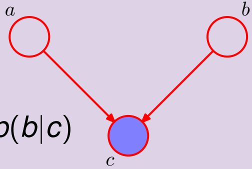


- Joint distribution:

$$p(a, b, c) = p(c|a, b)p(a)p(b)$$

- *a* and *b* are **conditionally independent**:

$$p(a, b) = \sum_c p(c|a, b)p(a)p(b) = p(a)p(b)$$

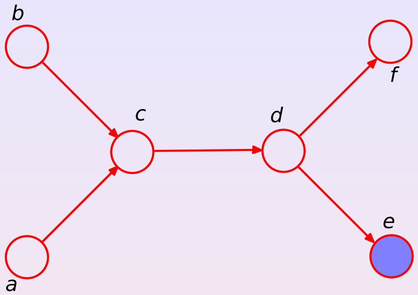


- *a* and *b* are **not conditionally independent given** *c*:

$$p(a, b|c) = \frac{p(c|a, b)p(a)p(b)}{p(c)} \neq p(a|c)p(b|c)$$

- *c* is *head-to-head* wrt to the path $a \to b$ as it is connected to the heads of the two arrows

d-separation



General Head-to-head

- Let a *descendant* of a node x be any node which can be reached from x with a path following the direction of the arrows
- A head-to-head node c unblocks the dependency path between its parents if either itself or *any of its descendants* receives evidence

General *d-separation* criterion

d-separation definition

- Given a generic Bayesian network
- Given A, B, C arbitrary nonintersecting sets of nodes
- The sets A and B are *d-separated* by C if:
  - All paths from any node in A to any node in B are *blocked*
- A path is blocked if it includes at least one node s.t. either:
  - the arrows on the path meet tail-to-tail or head-to-tail at the node and it is in C, or
  - the arrows on the path meet head-to-head at the node and neither it nor any of its descendants is in C

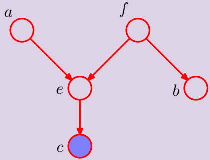
d-separation implies conditional independency

The sets A and B are independent given C ($A \perp\!\!\!\perp B \mid C$) if they are d-separated by C.

Example of general d-separation

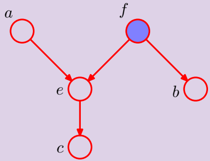
$a \not\perp\!\!\!\perp b | c$

- Nodes a and b are **not d-separated** by c:
  - Node f is tail-to-tail and not observed
  - Node e is head-to-head and its child c is observed



$a \perp\!\!\!\perp b | f$

- Nodes a and b are **d-separated** by f:
  - Node f is tail-to-tail and observed



Inference in graphical models

Description

- Assume we have evidence $\mathbf{e}$ on the state of a subset of variables in the model $\boldsymbol{E}$
- Inference amounts at computing the posterior probability of a subset $\boldsymbol{X}$ of the non-observed variables given the observations:

$$p(\boldsymbol{X}|\boldsymbol{E} = \mathbf{e})$$

Note

- When we need to distinguish between variables and their values, we will indicate random variables with uppercase letters, and their values with lowercase ones.

Inference in graphical models

Efficiency

- We can always compute the posterior probability as the ratio of two joint probabilities:

$$p(\boldsymbol{X}|\boldsymbol{E} = \mathbf{e}) = \frac{p(\boldsymbol{X}, \boldsymbol{E} = \mathbf{e})}{p(\boldsymbol{E} = \mathbf{e})}$$

- The problem consists of estimating such joint probabilities when dealing with a large number of variables
- Directly working on the full joint probabilities requires time exponential in the number of variables
- For instance, if all N variables are discrete and take one of K possible values, a joint probability table has K^N entries
- We would like to exploit the structure in graphical models to do inference more efficiently.

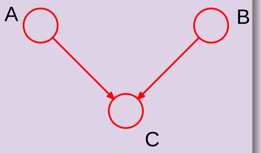
Example with head-to-head connection

A toy regulatory network

- Genes *A* and *B* have independent prior probabilities:

gene	value	P(value)
A	`active`	0.3
A	`inactive`	0.7

gene	value	P(value)
B	`active`	0.3
B	`inactive`	0.7

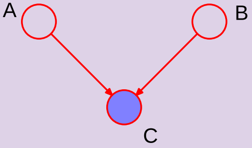


- Gene *C* can be enhanced by both *A* and *B*:

		A			
		`active`		`inactive`	
		B		B	
		`active`	`inactive`	`active`	`inactive`
C	`active`	0.9	0.6	0.7	0.1
C	`inactive`	0.1	0.4	0.3	0.9

Example with head-to-head connection

Probability of A active (1)



- Prior:

$$P(A = 1) = 1 - P(A = 0) = 0.3$$

- Posterior after observing active C:

$$P(A = 1|C = 1) = \frac{P(C = 1|A = 1)P(A = 1)}{P(C = 1)} \simeq 0.514$$

Note

The probability that A is active *increases* from observing that its regulated gene C is active

Example with head-to-head connection

Derivation

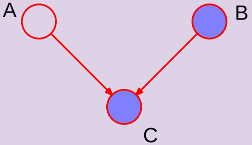
$$
\begin{aligned}
P(C=1|A=1) &= \sum_{B\in\{0,1\}} P(C=1,B|A=1) \\
&= \sum_{B\in\{0,1\}} P(C=1|B,A=1)P(B|A=1) \\
&= \sum_{B\in\{0,1\}} P(C=1|B,A=1)P(B)
\end{aligned}
$$

$$
\begin{aligned}
P(C=1) &= \sum_{B\in\{0,1\}} \sum_{A\in\{0,1\}} P(C=1,B,A) \\
&= \sum_{B\in\{0,1\}} \sum_{A\in\{0,1\}} P(C=1|B,A)P(B)P(A)
\end{aligned}
$$

Example with head-to-head connection

Probability of A active

- Posterior after observing that B is also active:



$$P(A=1|C=1,B=1) = \frac{P(C=1|A=1,B=1)P(A=1|B=1)}{P(C=1|B=1)} \simeq 0.355$$

Note

- The probability that A is active *decreases* after observing that B is also active
- The B condition *explains away* the observation that C is active
- The probability is still greater than the prior one (0.3), because the C active observation still gives some evidence in favour of an active A

Inference

Finding the most probable configuration

- Given a joint probability distribution $p(\mathbf{x})$
- We wish to find the configuration for variables $\mathbf{x}$ having the highest probability:

$$\mathbf{x}^{\max} = \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x})$$

for which the probability is:

$$p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} p(\mathbf{x})$$

Note

- We want the configuration which is *jointly* maximal for all variables
- We cannot simply compute $p(x_i)$ for each i and maximize it

Parameter estimation

- We assume the structure of the model is given
- We are given a dataset of examples $\mathcal{D} = \{\mathbf{x}(1), \dots, \mathbf{x}(N)\}$
- Each example $\mathbf{x}(i)$ is a configuration for *all* (complete data) or *some* (incomplete data) variables in the model
- We need to estimate the parameters of the model (conditional probability distributions) from the data

Learning Bayesian Networks

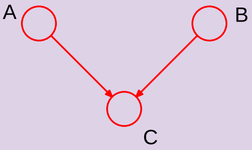
Simple case: complete data

- When training data are complete, we can estimate parameters simply by frequencies:
  1. Consider each conditional probability table (CPT) separately
  2. For each configuration of the variables, insert the number of times it occurred in the data
  3. Normalize each column to sum to one

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```

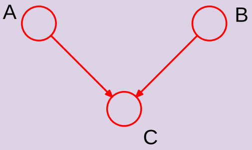


- Fill CPTs with counts

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```



- Fill CPTs with counts

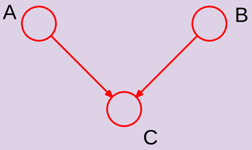
gene	value	counts
A	active	4
A	inactive	8

gene	value	counts
B	active	3
B	inactive	9

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```



- Normalize counts columnwise

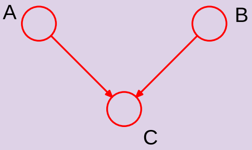
gene	value	counts
A	active	4/12
A	inactive	8/12

gene	value	counts
B	active	3/12
B	inactive	9/12

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```



- Normalize counts columnwise

gene	value	counts
A	active	0.33
A	inactive	0.67

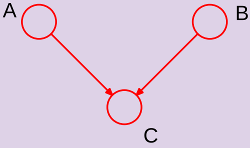
gene	value	counts
B	active	0.25
B	inactive	0.75

Learning Bayesian Networks

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```

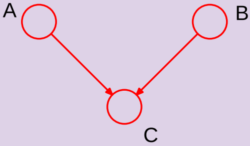


- Fill CPTs with counts

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```



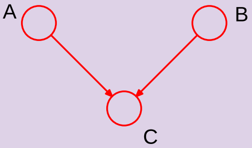
- Fill CPTs with counts

		A			
		active		inactive	
		B		B	
		active	inactive	active	inactive
C	active	1	2	1	0
C	inactive	0	1	1	6

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```



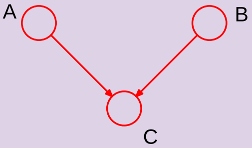
- Normalize counts columnwise

		A			
		active		inactive	
		B		B	
		active	inactive	active	inactive
C	active	1/1	2/3	1/2	0/6
C	inactive	0/1	1/3	1/2	6/6

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```



- Normalize counts columnwise

		A			
		active		inactive	
		B		B	
		active	inactive	active	inactive
C	active	1	0.67	0.5	0
C	inactive	0	0.33	0.5	1

Adding priors

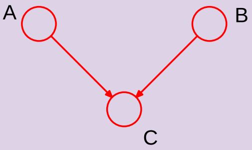
- The probability of configurations not occurring in training data is zero
- When few data available (always), this can be a too drastic choice
- Insert *prior* counts as imaginary configurations assumed to have been observed a-priori.
- E.g. one a-priori observation for each possible configuration

Learning Bayesian Networks

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```

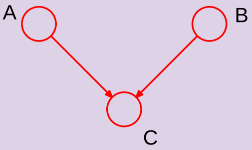


- Fill CPTs with priors as imaginary counts

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```



- Fill CPTs with priors as imaginary counts

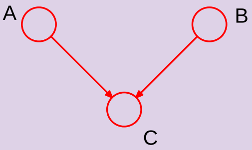
		A: active		A: inactive	
		B: active	B: inactive	B: active	B: inactive
C	active	1	1	1	1
C	inactive	1	1	1	1

Learning Bayesian Networks

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```

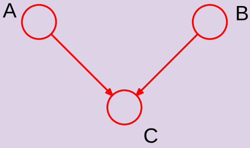


- Add observed counts

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```



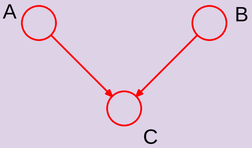
- Add observed counts

		A			
		active		inactive	
		B		B	
		active	inactive	active	inactive
C	active	1+1	1+2	1+1	1+0
C	inactive	1+0	1+1	1+1	1+6

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```



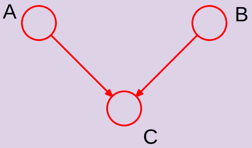
- Normalize counts columnwise

		A: active		A: inactive	
		B: active	B: inactive	B: active	B: inactive
C	active	2/3	3/5	2/4	1/8
C	inactive	1/3	2/5	2/4	7/8

Example

- Training examples as (A, B, C) tuples:

```
(act,act,act),(act,inact,act),
(act,inact,act),(act,inact,inact),
(inact,act,act),(inact,act,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact),
(inact,inact,inact),(inact,inact,inact).
```



- Normalize counts columnwise

		A			
		active		inactive	
		B		B	
		active	inactive	active	inactive
C	active	0.67	0.6	0.5	0.125
C	inactive	0.33	0.4	0.5	0.875

Incomplete data

- With incomplete data, some of the examples miss evidence on some of the variables
- Counts of occurrences of different configurations cannot be computed if not all data are observed
- We need approximate methods to deal with the problem

Learning with missing data: Expectation-Maximization

E-M for Bayesian nets in a nutshell

- Sufficient statistics (counts) cannot be computed (missing data)
- Fill-in missing data inferring them using current parameters (solve inference problem to get *expected* counts)
- Update parameters according to these expected counts
- Iterate until convergence to improve quality of parameters

Learning structure of graphical models

Approaches

constraint-based: test conditional independencies on the data and construct a model satisfying them

score-based: assign a score to each possible structure, define a search procedure looking for the structure maximizing the score

model-averaging: assign a prior probability to each structure, and average prediction over all possible structures weighted by their probabilities (full Bayesian, intractable)