Introduction to information theory and coding - Lecture 1 on Graphical models

Louis Wehenkel

Department of Electrical Engineering and Computer Science
University of Liège

Montefiore - Liège - October, 2011

Find slides: http://montefiore.ulg.ac.be/∼lwh/Info/

Motivations and structure of course

- Objective:
  Introduce and motivate graphical representation of qualitative and quantitative probabilistic knowledge
  - Qualitative notion of dependence
  - Characterization of desired properties of independence relations
  - Probability calculus as a model of Independence relations
- Two graphical representations of Independence relations
  - Undirected graphs: Markov networks
  - Directed graphs: Bayesian networks
  - Relations between these two types of representations
  - Quantitative aspects/questions
- In depth analysis of tree-structured graphical models
  - Undirected trees and the Chow-Liu algorithm
  - Directed trees and polytrees

Why do we need a qualitative notion of dependence?

- Making statements about independence (or relevance) is a profound feature of common-sense reasoning, while probability calculus gives a formalization and a safe procedure for testing any (conditional) Independence statements.
- However, this procedure relies on the computation of the probabilities of all combinations of statements, and is essentially intractable in large domains.
- In short, the probability calculus procedure is in itself not an operational model of reasoning about Independence relations, specially when we don't (yet) have the numbers.
- We would like to dispose of a kind of 'logic' of Independence, in which we can derive easily new Independence statements from previously established or postulated ones, without resorting to number crunching.

Desired properties of Independence relations

Consider a domain characterised by a finite set $\mathbb{U}$ of discrete variables, and let $\mathbb{A}, \mathbb{B}, \mathbb{C}$ denote three disjoint subsets of $\mathbb{U}$.

Let us denote by $\mathbb{A} \perp \mathbb{B} | \mathbb{C}$ the statement that "$\mathbb{A}$ is independent of $\mathbb{B}$, given that we know $\mathbb{C}$", i.e. when we already know the values of the variables in $\mathbb{C}$, we consider that the knowledge of the values of the variables in $\mathbb{B}$ is irrelevant to our beliefs about the values in $\mathbb{A}$.

We want to derive rules which are characteristic of independence relations, and which allow us to infer in a sound way new independence relations from established ones.

We will first propose a set of four rules and then verify that they are valid inference rules for probabilistic independence relations.

Semi-graphoids

What are the desired properties of an independence relation ?

- Symmetry:

$$(\mathbb{X} \perp \mathbb{Y}|\mathbb{Z}) \Leftrightarrow (\mathbb{Y} \perp \mathbb{X}|\mathbb{Z}). \quad (1)$$

- Decomposition:

$$(\mathbb{X} \perp (\mathbb{Y} \cup \mathbb{W})|\mathbb{Z}) \Rightarrow (\mathbb{X} \perp \mathbb{Y}|\mathbb{Z})\&(\mathbb{X} \perp \mathbb{W}|\mathbb{Z}). \quad (2)$$

- Weak union: NB: "strong" union will be defined later.

$$(\mathbb{X} \perp (\mathbb{Y} \cup \mathbb{W})|\mathbb{Z}) \Rightarrow (\mathbb{X} \perp \mathbb{Y}|(\mathbb{Z} \cup \mathbb{W})). \quad (3)$$

- Contraction:

$$(\mathbb{X} \perp \mathbb{Y}|\mathbb{Z})\&(\mathbb{X} \perp \mathbb{W}|(\mathbb{Z} \cup \mathbb{Y})) \Rightarrow (\mathbb{X} \perp (\mathbb{Y} \cup \mathbb{W})|\mathbb{Z}). \quad (4)$$

In other words, what are the desired properties of such a relation ?

- Symmetry:
  If $\mathbb{Y}$ tells us nothing about $\mathbb{X}$ (in some context $\mathbb{Z}$), then $\mathbb{X}$ tells us nothing about $\mathbb{Y}$.
- Decomposition:
  If two combined items of information are judged irrelevant to $\mathbb{X}$, then each separate item is irrelevant as well.
- Weak union:
  Learning some irrelevant information $\mathbb{W}$ cannot help the other irrelevant information $\mathbb{Y}$ become relevant.
- Contraction:
  If we judge $\mathbb{W}$ irrelevant to $\mathbb{X}$ after learning some irrelevant information $\mathbb{Y}$, then $\mathbb{W}$ must also have been irrelevant before we learned $\mathbb{Y}$.

Properties of the probabilistic independence relation

NB: if P is a probability distribution defined over the variables in $\mathbb{U}$, we write: $(\mathbb{A} \perp_P \mathbb{B}|\mathbb{C}) \Leftrightarrow (\forall a, b, c : P(b, c) > 0 \Rightarrow P(a|b, c) = P(a|c))$.

Theorem (Probabilistic independence)

The probabilistic independence relationship $(\cdot \perp_P \cdot | \cdot)$ *induced by any probabilistic model* P *satisfies the four properties (1)-(4) (symmetry, decomposition, weak union and contraction).*

Theorem (Intersection property)

The probabilistic independence relationship induced by any strictly positive probabilistic model P *also satisfies*

$$(\mathbb{X} \perp_p \mathbb{Y}|(\mathbb{Z} \cup \mathbb{W}))\&(\mathbb{X} \perp_P \mathbb{W}|(\mathbb{Z} \cup \mathbb{Y})) \Rightarrow (\mathbb{X} \perp_P (\mathbb{Y} \cup \mathbb{W})|\mathbb{Z}). \quad (5)$$

Induced inference rules

- Chaining rule:

$$(\mathbb{X} \perp \mathbb{Z}|\mathbb{Y})\&((\mathbb{X} \cup \mathbb{Y}) \perp \mathbb{W}|\mathbb{Z}) \Rightarrow \mathbb{X} \perp \mathbb{W}|\mathbb{Y}.$$

- Mixing rule:

$$(\mathbb{X} \perp (\mathbb{Y} \cup \mathbb{W})|\mathbb{Z})\&(\mathbb{Y} \perp \mathbb{W}|\mathbb{Z}) \Rightarrow (\mathbb{X} \cup \mathbb{W}) \perp \mathbb{Y}|\mathbb{Z}.$$

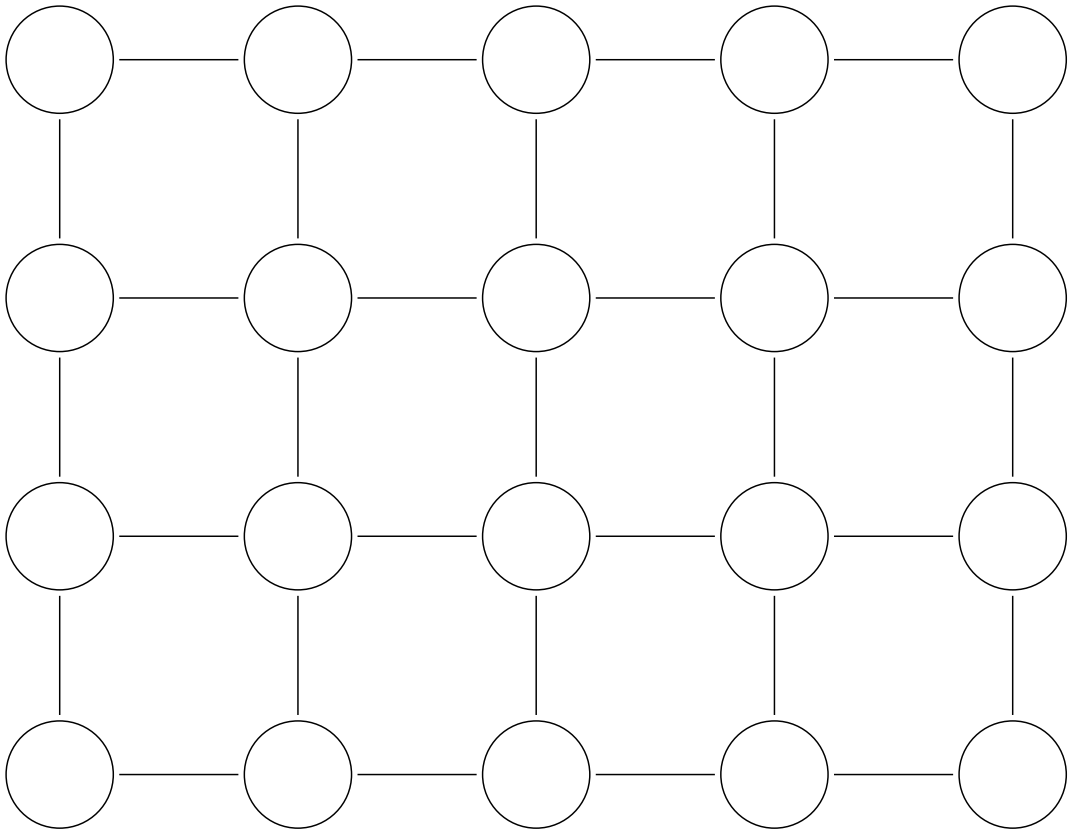
Exercise: Show that these two rules follow logically from (1) to (4), and hence are valid inference rules for any probabilistic independence relation.

Summary

- We have abstracted from the quantitative notion of conditional independence defined by probability theory.
- This abstraction is necessary for efficient manipulation of the notion of independence/irrelevance.
- We have shown, to some extent, that one can axiomatize the notion of independence in a way which remains logically coherent with the same notion defined by probability calculus.
- We have illustrated that such an axiomatization is useful to derive new independencies from postulated ones, and even new inference rules from postulated ones.
- However, we are still lacking an intuitive and efficient way to reason ourselves coherently in this framework.

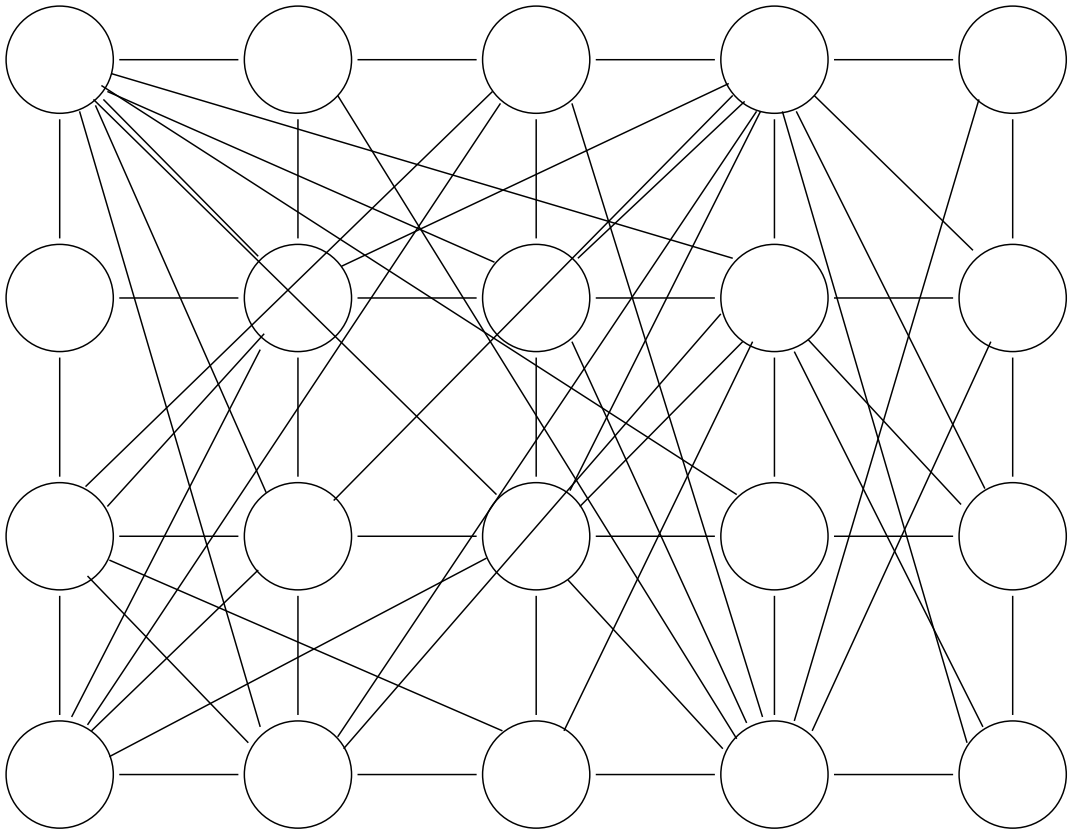
Why graphical (independence) models?

- A picture is worth a thousand words...



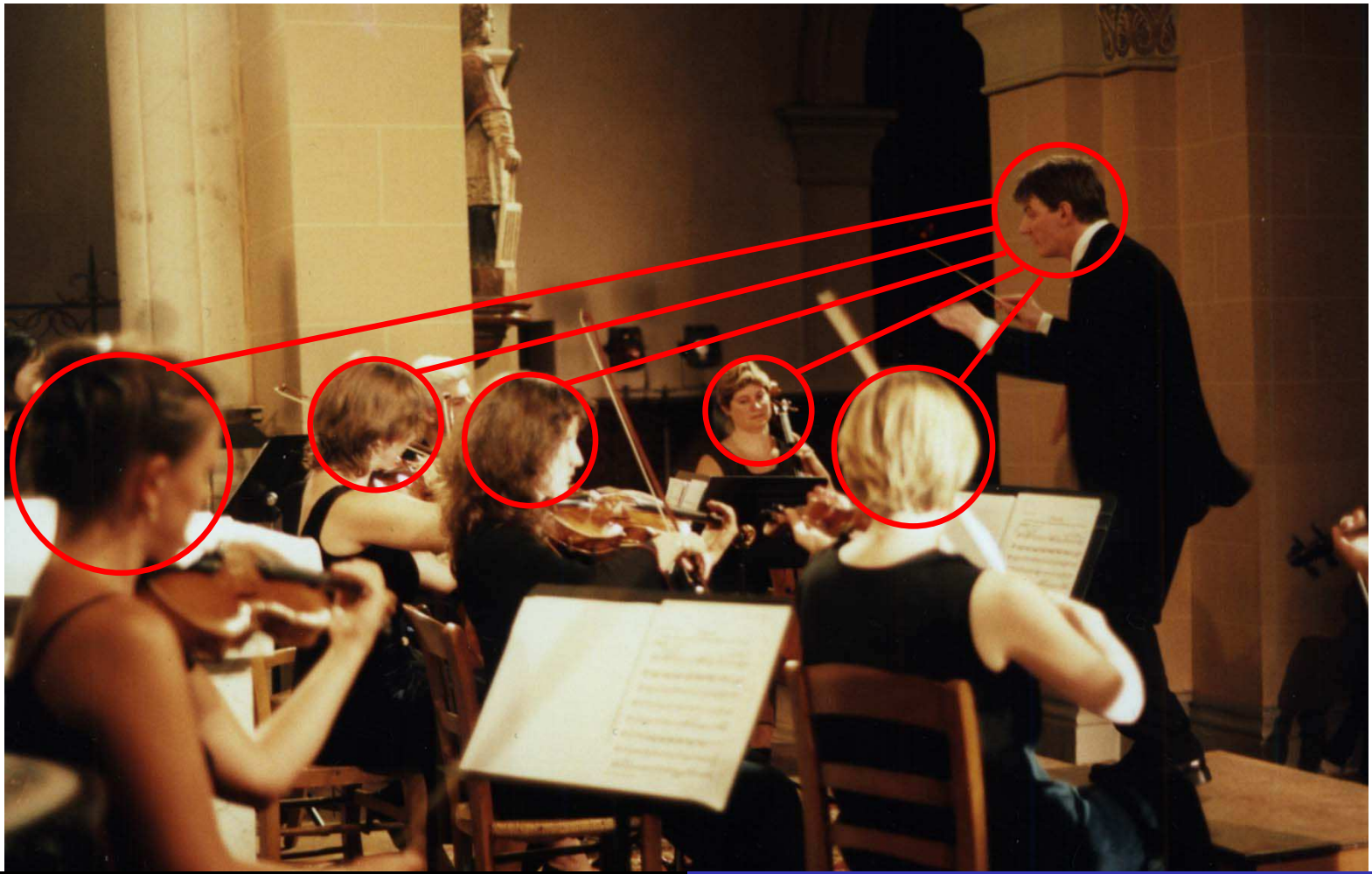
Why graphical models?

- A picture is worth a thousand words...



Why graphical models?

- A picture is worth a thousand words...



Undirected graphs as independence models

Notion of undirected graph:

- A (general) graph is denoted by $G = (V, E)$ where V is a finite set of vertices, and $E \subset V \times V$ is the set of edges.
- A path (of length $n > 0$) in G, is a sequence of different vertices $v_1, v_2, \ldots, v_{n+1}$ such that $(v_i, v_{i+1}) \in E, i = 1, \ldots, n$.
- An edge $(v, v') \in E$ such that $v = v'$ is called a loop.
- An edge $(v, v') \in E$ such that $v \neq v'$ and $(v', v) \in E$ is called a line.
- An edge which not a line nor a loop is called an arrow.
- We say that G is undirected if G has no loops and no arrows (i.e. G has only lines).

Vertex separation in undirected graphs

From local to global vertex separation:

- Consider an undirected $G = (\mathbb{U}, E)$.
- The absence of a line between two variables represents the absence of a direct interaction between them.
- All other relations are induced by the notion of separation:
  We say that in a graph G the sets $\mathbb{A}$ and $\mathbb{B}$ are separated by $\mathbb{C}$ if all paths from $\mathbb{A}$ to $\mathbb{B}$ traverse $\mathbb{C}$.
  We denote this by $(\mathbb{A}; \mathbb{B} | \mathbb{C})_G$.
- In particular, we say that the sets $\mathbb{A}$ and $\mathbb{B}$ are separated if there is no path from $\mathbb{A}$ to $\mathbb{B}$.
- In particular, if there is no line connecting $\mathbb{A}$ to $\mathbb{B}$, then $(\mathbb{A}; \mathbb{B} | \mathbb{U} \setminus (\mathbb{A} \cup \mathbb{B}))_G$

Undirected graphs as independence models

- The good news:
  - The vertex separation relation satisfies properties (1) to (5) (please check this yourself)
  - Vertex separation is easy to check (polynomial time).
- Questions:
  - Is vertex separation compatible with probabilistic independence?
  - How general is vertex separation wrt probabilistic independence?
  - What kind of independence relations can be exactly represented by vertex separation?

Notions of dependency, independency and perfect maps

Consider a distribution P and an undirected graph G over $\mathbb{U}$.

Definition (D-map (independent subsets are indeed separated))

G is a D-map of P if for any three disjoint $\mathbb{A}, \mathbb{B}, \mathbb{C} \subset \mathbb{U}$ we have

$$(\mathbb{A} \perp_P \mathbb{B}|\mathbb{C}) \Rightarrow (\mathbb{A}; \mathbb{B}|\mathbb{C})_G. \tag{6}$$

Definition (I-map (separated subsets are indeed independent))

G is a I-map of P if for any three disjoint $\mathbb{A}, \mathbb{B}, \mathbb{C} \subset \mathbb{U}$ we have

$$(\mathbb{A} \perp_P \mathbb{B}|\mathbb{C}) \Leftarrow (\mathbb{A}; \mathbb{B}|\mathbb{C})_G. \tag{7}$$

Definition (Perfect map (equivalence between "$\perp_P$" and ";"))

G is a perfect map of P if it is a D-map and an I-map of P.

Representation power of UGs

- Preliminary comments:
  - Any P has at least a D-map (e.g. the empty graph)
  - Any P has at least an I-map (e.g. the complete graph)
  - Some P have no perfect-map (e.g. two coins and a bell)
- There is thus a need to delineate more precisely
  - those dependency models that have perfect maps, and
  - those graphical models which are perfect maps of a dependency model
  - provide constructive algorithms to switch between P and G.
- We say that a dependency model M (i.e. a rule that assigns truth values to a three-place relation $(\mathbb{A} \perp_M \mathbb{B}|\mathbb{C})$ over disjoint subsets of some $\mathbb{U}$) is graph-isomorph if there exists an undirected graph $(\mathbb{U}, E)$ which is a perfect map of M.
- Goal: characterize graph-isomorph probabilistic models.

On the structure of the set of UGs over some $\mathbb{U}$

- Lattice structure:
  - For a fixed $\mathbb{U}$, we can identify an undirected graph $G = (\mathbb{U}, E)$ with its set of edges E.
  - The set of edges can itself be identified with a subset of the set of pairs $\{v, v'\} \in \mathbb{U}$.
  - For any $G = (\mathbb{U}, E)$ and $G' = (\mathbb{U}, E')$, let us write $G \subset G'$ if $E \subset E'$.
- Monotonicity wrt addition or removal of edges:
  - if G is a D-map of P, any $G' \subset G$ is also a D-map of P,
  - if G is an I-map of P, any $G' \supset G$ is also an I-map of P.
- Extreme maps:
  - G is a minimal I-map, if there is no $G' \subset G$ (other than G itself) which is also an I-map.
  - G is a maximal D-map, if there is no $G' \supset G$ (other than G itself) which is also a D-map.

Characterization of graph-isomorph dependency model

Theorem (Graph isomorph dependency model M)

A necessary and sufficient condition for a dependency model M over some $\mathbb{U}$ to be graph-isomorph, is that is satisfies Symmetry, Decomposition, Intersection, Strong union and Transitivity,

where Strong union means that:

$$(\mathbb{X} \perp_M \mathbb{Y}|\mathbb{Z}) \Rightarrow (\mathbb{X} \perp_M \mathbb{Y}|(\mathbb{Z} \cup \mathbb{W})), \tag{8}$$

and Transitivity means that:

$$(\mathbb{X} \perp_M \mathbb{Y}|\mathbb{Z}) \Rightarrow \forall \gamma \in \mathbb{U} : (\mathbb{X} \perp_M \{\gamma\}|\mathbb{Z}) \text{ or } (\{\gamma\} \perp_M \mathbb{Y}|\mathbb{Z}). \tag{9}$$

(NB: $\{\gamma\}$ denotes a singleton subset of $\mathbb{U}$)

Question 1: Given P construct minimal I-map G of P

Theorem (Existence, unicity, construction of minimal I-map)

Every dependency model M which satisfies symmetry, decomposition and intersection, has a unique minimal I-map $G_0 = (\mathbb{U}, E_0)$ produced by connecting only those pairs (v, v') for which $(\{v\} \perp_M \{v'\} | \mathbb{U} \setminus \{v, v'\})$ is FALSE.

- Motivation: a minimal I-map is a graph displaying a maximal number of independencies without false-positives.
- Notice that the property holds for independencies displayed by strictly positive P.
- NB:
  - existence of an I-map guarantees existence of a minimal one, but not unicity.
  - NB: unicity of minimal I-map implies that any I-map is a superset of the unique minimal one.

Question 2: Check whether G is an I-map of P

- If P is strictly positive, we can check whether G is an I-map, by constructing first a minimal I-mapG_0 of P and then checking whether $G_0 \subset G$.
- Assuming that we can establish $(\{v\} \perp_P \{v'\} | \mathbb{U} \setminus \{v, v'\})$ for any (v, v') in constant time:
  - building a minimal I-map G_0 may be done in polynomial time (quadratic in the number of variables).
  - checking $G_0 \subset G$ may also be done in polynomial time (quadratic).
  - Hence these problems are solved "efficiently" if we have an oracle providing in constant time answers to "elementary" queries of conditional independence of two variables given all others.

Markov networks, blankets and boundaries

Definitions:

- Given P (resp. M), we say that G is a Markov network of P (resp. M) is it is a minimal I-map of P (resp. M).
- A Markov blanket $BL_M(v)$ of $v \in \mathbb{U}$ is any subset $\mathbb{S} \subset \mathbb{U}$ for which $(\{v\} \perp_M U \setminus (\{v\} \cup \mathbb{S}) | \mathbb{S})$.
- A Markov boundary $B_M(v)$ of $v \in \mathbb{U}$ is a minimal Markov blanket.

Theorem (Unicity and construction of Markov boundaries)

Every element v of a dependency model M which satisfies symmetry, decomposition, intersection, and weak union, has a unique Markov boundary, and this corresponds with the set of vertices adjacent to v in the minimal I-map G_0 of M.

Summary of UG models

- We have seen that we can make use of undirected models to represent useful independencies, in a way compatible with the definition of probabilistic conditional independence.
- Not all independence structures may be represented by UGs.
- But we can commit with the idea of building the most refined model of them in the form of a minimal I-map.
- Further topics of relevance:
  - For any G is there a P such that its G is a perfect map ? (answer is yes, with a few hypotheses).
  - Once we have a minimal I-map (or simply an I-map) how do we "decorate" this structure with numerical information to represent a particular probability distribution P ? (later)
  - How to carry out "numerical computations" with such structures ? (later)

Directed graphical models: motivations

- Undirected graphical models lack of representation power:
  - unable to represent induced and non-transitive dependencies.
  - no representation of causality.
- To overcome these deficiencies, we will use the language of directed graphs.
  - Arrows (instead of lines) may allow to distinguish genuine dependencies from spurious ones (see two-coins + bell example).
  - Arrows also allow to impose (asymmetric) causal relations on top of (symmetric) dependencies.
- Menu:
  - Define Bayesian networks and their semantics as independence maps.
  - Answer questions 1, 2 and 3 along the treatment we have done for Markov models.

Directed acyclic graphs (DAGs)

- A directed graph $D = (V, E)$ is a graph with only arrows:
  - i.e. no loops and no lines,
  - or, in other words, $(v, v') \in E \Rightarrow v \neq v' \& (v', v) \notin E$.
- A cycle of length $n > 0$, in a graph $G = (V, E)$, is a sequence $v_1, \ldots, v_{n+1}$ such that $(v_i, v_{i+1}) \in E \,\&\, v_1 = v_{n+1}$.
- The cycle is said to be simple (or proper) if all nodes except v_1 and v_{n+1} are different.
- A DAG is a directed graph without any cycle.
- Note that this is equivalent to saying:
  - that a DAG is a directed graph without any simple cycle.
  - or that a DAG is a graph without any cycle.

D-separation in DAGs

Definition (D-separation)

If $\mathbb{X}, \mathbb{Y}, \mathbb{Z}$ are three disjoint sets of vertices in a DAG D, then $\mathbb{Z}$ is said to d-separate $\mathbb{X}$ from $\mathbb{Y}$, denoted by $(\mathbb{X}; \mathbb{Y}|\mathbb{Z})_D$, if there is no path between a vertex in $\mathbb{X}$ and a vertex in $\mathbb{Y}$ along which the following two conditions hold: (i) every node with converging arrows is in $\mathbb{Z}$ or has a descendant in $\mathbb{Z}$ and (ii) every other node is outside of $\mathbb{Z}$.

- If a path satisfies the above condition, it is said to be active; otherwise it is said to be blocked.
- A DAG is an I-map of P is all its d-separations correspond to conditional independencies satisfied in P.
- It is a minimal I-map, or a Bayesian network of P, if none of its arrows can be deleted without destroying its I-mapness.

Construction of Bayesian networks for a distribution P

Definition (Boundary DAG of M relative to a vertex ordering)

Let M be a dependency model over $\mathbb{U}$ and $d = (v_1, \ldots, v_n)$ any ordering of the elements of $\mathbb{U}$, and define the sequence of nested sets $\mathbb{U}_i^d$ by $\mathbb{U}_1^d = \emptyset$ and $\mathbb{U}_i^d = \{v_1, \ldots, v_{i-1}\}$, for $i = 2, \ldots, n$.

The boundary strata of M relative to d is an ordered set of subsets of $\mathbb{U}, (\mathbb{B}_1^d, \mathbb{B}_2^d, \ldots, \mathbb{B}_i^d, \ldots)$ such that $\mathbb{B}_i^d$ is a Markov boundary[1] of $\{v_i\}$ wrt. $\mathbb{U}_i^d$.

The DAG created by designating each $\mathbb{B}_i^d$ as parent set of v_i is called the boundary DAG of M relative to the ordering d.

[1] ie. is a minimal subset of $\mathbb{U}_i^d$ satisfying $(\{v_i\} \perp_M (\mathbb{U}_i^d \setminus \mathbb{B}_i^d) | \mathbb{B}_i^d)$

DAGs as minimal I-maps of Semi-graphoids

- Nota Bene:
  - given any M, the existence (not necessarily uniquely) of a boundary strata is established once d is given.
  - hence, there is a at least one boundary DAG defined by each ordering d; they may all be different or all identical or diverse (cf examples).
  - if M is induced by a strictly positive P, then for each ordering d the boundary strata is unique as well as its induced boundary DAG.
  - Still, different orderings may yield identical boundary DAGs.
- Main result:
  - For any semi-graphoid M (in particular, for any P induced independence model) and any ordering d, any corresponding boundary DAG is a minimal I-map of M.

Corollaries of the main result

- Given P over $\mathbb{U}$ and any ordering $d = (X_1, \ldots, X_n)$ of the variables, the DAG created by designating as parents of X_i any minimal subset of Π_{X_i} of predecessors of X_i in d satisfying

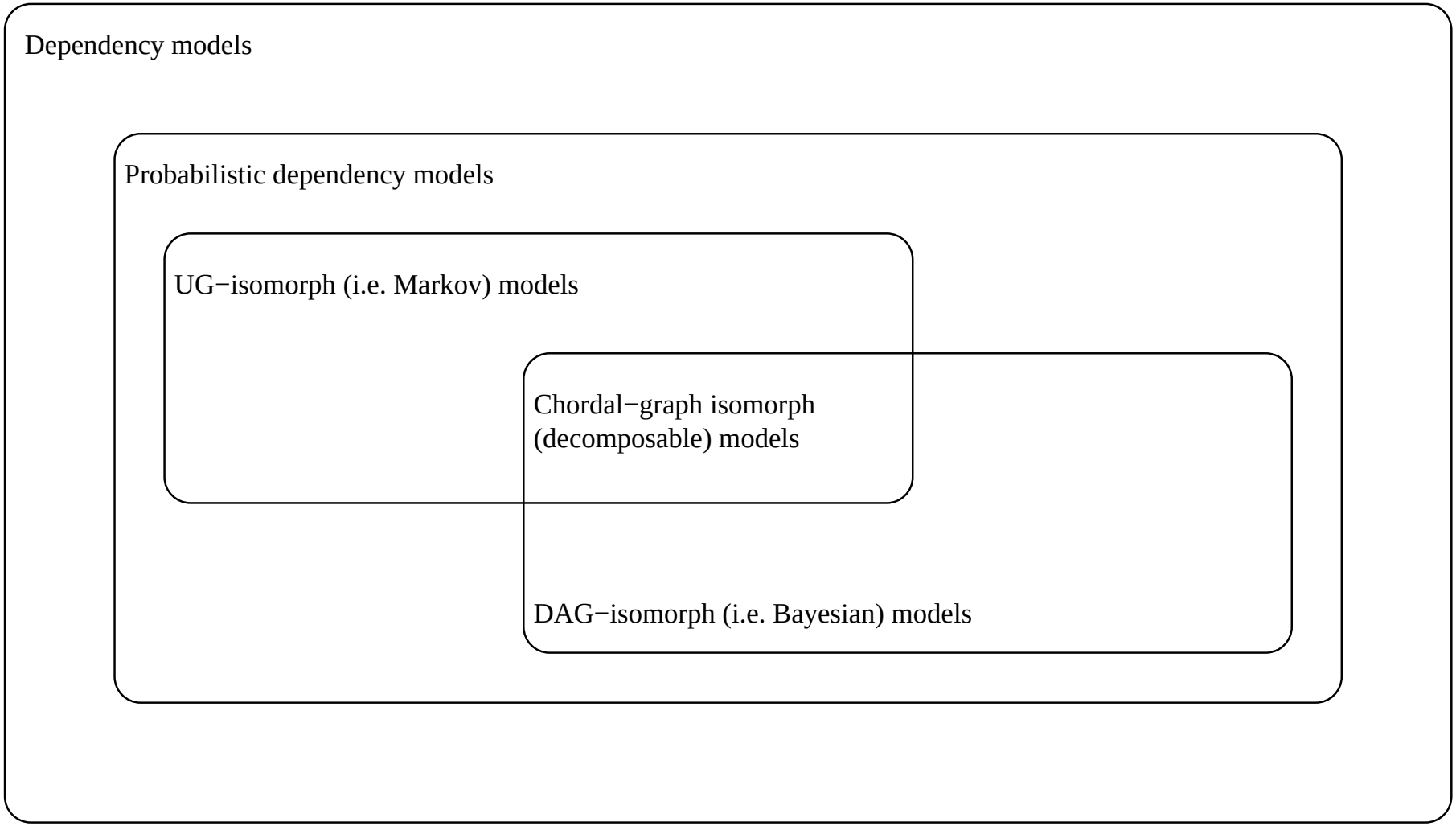
$$P(X_i|X_1, \ldots, X_{i-1}) = P(X_i|\Pi_{X_i})$$

  is a Bayesian network of P. If P is strictly positive then this is uniquely defined.
- A necessary and sufficient condition for a DAG D to be a Bayesian network of P is that each variable X be conditionally independent of all its non-descendants, given its parents Π_X, and that no proper subset of Π_X satisfies this condition.
- If any Bayesian network D is constructed by the boundary strata method in some ordering d, then any ordering d' consistent with the direction of arrows in D will lead to the same D.

Summary of DAG models

- We have seen that we can make use of directed acyclic graphical models to represent useful independencies, in a way compatible with the definition of probabilistic conditional independence.
- Not all independence structures may be represented exactly by DAGs (examples will be given).
- But we can commit with the idea of building the most refined model of them in the form of a minimal I-map.
- As with UGs, we can infer independencies by inspection of the graph, in polynomial time.

The global picture



Quantitative aspects

- What do we need to add to a minimal I-map graphical structure to describe fully a given P?
  - UGs: parameterization via potential (or compatibility) functions over cliques.
  - DAGs: parameterization via conditional distributions over families.
- How can we compute with parameterized DAG or UG P-models?
  - Exact computations: reduce du CG and use (generalized) forward-backward algorithm.
  - Approximations: turn problem into a tractable optimization problem (subject of current research).
- How can we infer UG or DAG models from data ?

Tree structured graphical models

Motivations

- Tree structured models offer simple interpretations
- Efficient inference algorithms
- Efficient learning algorithms

Two classes

- Undirected trees and their equivalent directed versions
- Polytrees : DAGs whose skeleton is a tree

A few additional definitions from graph theory

Skeleton of a DAG: UG obtained by replacing all arrows by lines.

Directing an UG: DAG obtained by replacing every line by an arrow, under the constraint of producing a DAG.

Induced subraph: (of $G = (V, E)$) by $V' \subset V$ is the graph $G(V') = (V', E \cap V' \times V')$. (I.e; induced subgraphs of UGs (resp. DAGs) are UGs (resp. DAGs).)

Clique of an UG: a clique of $G = (V, E)$ is an induced subgraph $G(V')$ such that $\forall v, v' \in V' : v \neq v' \Rightarrow (v, v') \in E$.

Maximal clique: a clique which can not be augmented while maintaining the property of being a clique, i.e. a maximal subgraph whose vertices are all adjacent to each other in G.

Parameterizing UGs

In order to create a P from an UG we proceed as follows (see Pearl for additional details):

1. Identify the set $\mathcal{C} = \{C_1, \ldots\}$ of all maximal cliques of G.
2. For each $i \leq \#\mathcal{C}$, assign a compatibility function $g_i(\cdot)$ which maps configurations of the subset of variables of the clique to a non-negative real number.
3. Write

$$P(x_1, \ldots, x_n) = \frac{\prod_i g_i(c_i(x_1, \ldots, x_n))}{\sum_{x_1,\ldots,x_n} \prod_i g_i(c_i(x_1, \ldots, x_n))},$$

where $c_i(x_1, \ldots, x_n)$ extracts from a configuration of all the variables of the UG the corresponding configuration of the subset of variables of the clique C_i.

But: How to derive the functions g_i from evidence, and how to compute the normalization constant are practical problems...

Example: Markov chain $X - Y - Z$

- Cliques: $X - Y$ and $Y - Z$
- Compatibility functions: $g_1(x, y)$ and $g_2(y, z)$
- Suppose we know $P(X, Y, Z)$: how to derive the g_i from P?
- Idea 1:
  - let us take $g_1(x, y) = P(x, y)$ and $g_2(y, z) = P(y, z)$
  - we get $P(x, y, z) = (P(x, y)P(y, z))/K$ where $K = \sum_{x \in X} \sum_{y \in Y} \sum_{z \in Z} P(x, y)P(y, z)$
  - Impossible in general (Why ?)
- Idea 2:
  - Let us, instead, take $g_1(x, y) = P(x, y)/\sqrt{P(y)}$ and $g_2(y, z) = P(y, z)/\sqrt{P(y)}$: we get $K = 1$! (Why ?)
  - Possible in general (Why ?)

Example: Markov chain $X - Y - Z$ (continued)

- We could as well take $g_1(x, y) = P(x, y) = P(x)P(y|x)$ and $g_2(y, z) = P(z|y)$, which corresponds to the parameterization of the DAG $X \to Y \to Z$;
- or $g_2(y, z) = P(y, z) = P(z|y)P(y)$ and $g_1 = P(x|y)$ which corresponds to $X \leftarrow Y \to Z$;
- or $g_2(y, z) = P(z)P(y|z)$ and $g_1 = P(x|y)$ which corresponds to $X \leftarrow Y \leftarrow Z$.
- But, we could not take the parameterization of the DAG $X \to Y \leftarrow Z$.
- The three first parameterizations correspond to directed versions of the UG which do not introduce a v-structure, while the fourth one introduces a v-structure.

Markov trees

- We use indifferently the term Markov tree, tree, or tree-structured UG, to denote UGs whithout any cycles.
- Typically, we assume in addition that these trees are singly connected, i.e. such that there is a path from any vertex to anyother vertex, and use the term 'forest' to denote the case where not all nodes are connected.
- In a singly connected tree over n vertices, we always have exactly $n-1$ edges.
- In a forest over n vertices, we have $n-c$ edges, where c is the number of connected components.

General procedure for parameterizing Markov trees:

1. For each edge (i.e. maximal clique C_i) $e = X - Y$ in a Markov tree use a function $g_e(x_1, \ldots, x_n) = P(x(e), y(e))$ derived from $P(\mathbb{Z})$
2. For each vertex X_i in a Markov tree, use a function $g_{X_i}(x_i) = (P(x_i))^{d(X_i)-1}$, where $d(X_i)$ denotes the number of neighbors of the vertex X_i in G.
3. Write:

$$P_G(x_1, \ldots, x_n) = \frac{\prod_{e \in E(G)} g_e(x(e), y(e))}{\prod_{X_i \in V(G)} g_{X_i}(x_i)}.$$

4. Now, let us consider what we get if we first direct the Markov tree, and then use the DAG parameterization procedure to infer a probability distribution from it... (from examples)

I-map preserving direction of tree-structured UGs

- Theorem A: any directed version of a tree-structured UG which has no *v*-structure produces a DAG which represents exactly the same set of independencies as the original undirected tree. (also true for chordal graphs)
- Corollary A1: tree-structured UGs may be parameterized by first directing them without introducing any *v*-structure, and then parameterizing the resulting DAG.
- Algorithm: to direct a tree-structured UG in such a way that no *v*-structures are introduced
  1. Choose first a root of the tree: any node of the UG
  2. Direct its arcs 'away' from the root
  3. Proceed recursively by directing the yet not directed arcs of the successors 'away' from them.

Summary

- Like DAGs, tree structured UGs may be parameterized 'easily' to represent a P which satisfies the independencies encoded by the UG, by first directing the tree structured UG without introducing v-structures (which maintains the encoded independencies), and by then using the DAG parameterization procedure to attach conditional distributions to nodes.
- The generalization of these ideas carries over to "chordal graphs", i.e. UGs such that every cycle of length 4 or more has a chord (a line between non-consecutive vertices of the cycle), with some modifications (see Pearl, chapter 3...)
- See examples to understand why chordality is essential to yield such a decomposable representation of a probability distribution...

Learning structure from data (chapter 8 of Pearl)

- Main question: how to infer the graph structure from the information at hand
- We will limit ourselves to tree structures
- We will decompose the question in this context into three successive questions:
  - Given a $P(x)$ known to factorize according to a tree structured graph, how to efficiently recover its tree-structured perfect map.
  - Given a general $P(x)$, can we recover the best approximation of $P(x)$ in the form of a parameterization of a tree structured graph.
  - Given only a sample from a generative distribution, how to answer the two preceding questions.
- These questions will be declined successively with tree structured UGs and general tree structured DAGs (polytrees).

Intuition about structure inference:

- Consider the case of three variables X, Y, Z, and suppose that we know that they form a Markov chain, but that we don't know in which order.
- In other words, we hesitate between the three following structures: $X - Y - Z$, $Y - X - Z$, $X - Z - Y$.
- Suppose that we are able to compute $I(X;Y)$, $I(Y;Z)$ and $I(X;Z)$:
  - Can we infer from these three quantities a correct structure ?
  - The answer is YES.
  - Sort the quantities $I(X;Y)$, $I(Y;Z)$ and $I(X;Z)$ by decreasing order of numerical value, take the two first and create an UG with lines among the corresponding two pairs of variables.
  - Explanation: data processing inequality !

Generalization to distributions which factorize according to a tree-structured UG

- We want to represent graphically the independencies of a distribution $P(X_1, \ldots, X_n)$ known to be Markov w.r.t. to a tree-structured UG (but we do not know the structure).
- Algorithm (Chow and Liu, 1968):
  1. Compute the pairwise mutual informations $I(X_i; X_j), \forall i \neq j$.
  2. Assign a line between the variables corresponding to the largest mutual information.
  3. Examine the next largest information and assign a line, unless it creates a cycle in the graph.
  4. Repeat step 3, until $n - 1$ branches have been assigned.
- Select an arbitrary node as root, direct the UG from it (without introducing v-structures) and to each X_i assign $P(X_i|X_{p(i)})$ (where $p(i)$ addresses the (sole) parent of X_i).

Comments about the Chow and Liu algorithm

- When we want to infer a tree-structured UG (or a directed version of it without v-structures) for a target distribution, and dispose of means to compute pairwise quantities from the target distribution, in the form of mutual informations among variables and conditional distributions of one variable given another, we dispose of an 'efficient' algorithm for generating a Markov network (order n^2, roughly).
- The Chow Liu algorithm is an instance of the 'maximum weight spanning tree' algorithm of graph theory (MWST).
- NB: In the algorithm, we may in principle be led to situations where the $I(X_i; X_j)$ of the next line to assign is equal to zero; if this is the case we can immediately stop the procedure (leading to a 'forest' model, i.e. a model where some subsets of variables are disconnected).

Approximation of probability distributions

- In many practical situations, we do not dispose of precise information about de probability distribution at hand.
- In particular, in such contexts, we are not able to verify in a definite way independencies such as $(X_i \perp X_j | X_k)$.
- In other words, we can only estimate/approximate quantities such as $I(X_i; X_j)$ or $P(X_i | X_{p(i)})$.
- Then the following question arises:
  How to infer precise probabilistic models from imprecise data ?
- Approach:
  - Define a space of target probability distributions (model).
  - Define a measure of discrepancy between distributions.
  - Choose the probability distribution in the target space which is as 'compatible as possible' with the information at hand.

Measuring the compatibility among two distributions

- Kullback-Leibler divergence:

$$D(P, P') = \sum_x P(x) \log \frac{P(x)}{P'(x)}.$$

- tends to zero when $P \to P'$.
- has the likelihood interpretation, when P is inferred from a sample (...explained on the blackboard).
- Given a space $\mathcal{P}$ of distributions and a target distribution P, we thus may seek to compute

$$\hat{P}(\mathcal{P}) = \arg \min_{P' \in \mathcal{P}} D(P, P'),$$

which we call the D-projection of P onto $\mathcal{P}$.

Some first results

- Let us consider the space $\mathcal{P}^t$ of probability distributions that may be represented by an undirected tree t and a target probability distribution P.
- Then the D-projection of P onto the space $\mathcal{P}^t$ is obtained simply by directing t without introducing v-structures and by assigning to each node i in t the conditional distribution $P(X_i|X_{p(i)})$ where $p(i)$ denotes the father of i in a directed version of t.
- (For the proofs see Pearl chapter 8).
- Furthermore, to minimize D over the space of trees, we can simply use Chow Liu based on information quantities derived from P.

Comments about the Chow-Liu algorithm

- Given any probability distribution P and the means to compute pairwise mutual informations and pairwise conditional distributions in P, this algorithm allows to infer (in quadratic time), a tree structured approximation of P.
- The resulting distribution P' is the one, among all that factorize along UG trees, that is closest according to the distance measure $D(P, P')$.
- In particular, if P is Markov wrt to an UG tree, then the resulting P' will be equal to P.

Learning from observations drawn from P

- Let us consider a sample of observations $S = (x^1, \ldots, x^N)$ drawn i.i.d. from a target distribution $P(x)$ (where each x^i is actually an n-tuple, having one element for each variable X_j.
- Given anyother distribution P' defined over the same set of variables, we define the sample log-likelihood, by

$$IL(S, P') = \sum_{i=1}^{N} \log P'(x^i) = \log \left(\prod_{i=1}^{N} P'(x^i) \right)$$

- Given a space $\mathcal{P}$ of candidate distributions, a classical criterion use in statistics, is to choose the one which maximizes the sample likelihood.

Learning from observations drawn from P

- Let us consider a configuration x, and denote by $N(x)$ the number of observations in our sample which correspond to that configuration and by $F(x) = N(x)/N$ their relative frequency among the N observations.
- We can rewrite the log-likelihood of the sample wrt to P' as

$$IL(S, P') = N \sum_x F(x) \log P'(x)$$

.

- We then immediately see that maximizing the log-likelihood of the sample by choosing P' is equivalent to choosing P' so as to minimize the KL-divergence $\sum_x F(x) \log \frac{F(x)}{P'(x)}$. Indeed,

$$\sum_x F(x) \log \frac{F(x)}{P'(x)} = -N^{-1} L(S, P') + \sum_x F(x) \log F(x).$$

Learning a Markov tree approximation from a sample

- Goal: find a tree structure and a parameterization such that the sample likelihood is maximal (over all possible trees and parameterizations of them).
- Solution: use sample to estimate mutual informations, by replacing probabilities by relatives frequencies derived from the sample (see example on the blackboard), then apply Chow-Liu to get MWST, then choose a root, then use again sample to estimate the conditional probabilities needed for each vertex.

Polytrees

Explain the main differences between polytrees and trees.
Explain very briefly learning of polytree structures (François takes care of the rest...)