Probabilistic Graphical Models

Master of Data Science

Alexandre Aussem

LIRIS UMR 5205 CNRS
Data Mining & Machine Learning Group (DM2L)
University of Lyon 1
Web: perso.univ-lyon1.fr/alexandre.aussem

November 3, 2017

Présentation du cours PGM

- Total de 32 heures de cours :
  - 16h CM (5 séances),
  - 8h TD (2 séances)
  - 8h de TP (2 séances)
- Travaux pratiques :
  - TP1 : 4h TP avec BNLearn, librairie R pour l'apprentisage de la structure et des paramètres des réseaux bayésiens et pour l'inférence.
  - TP2 : 4h TP deep learning avec Keras et Theano, des librairies Python pour optimiser et évaluer les expressions mathématiques avec des données multi-dimensionelles efficacement grâce au GPU.

Outline

Outline

Outline

Modeling complex data

- To model complex data, several questions have to be answered:
  - What is the task and the loss function?
  - What are the statistical properties and assumptions and underlying the data generating process?
  - What have to be captured from the probabilistic distribution to perform the task ?
  - How to learn the model parameters and perform inference in reasonable time?
- Once the model is chosen, two more issues:
  - Learning of the parameters of the model.
  - Inference of probabilistic queries

Examples

- **Image:** In a monochromatic image, each pixel is represented by a discrete random variable. The image may be modelled using a Markov network.
- **Bioinformatics:** Consider a long sequence of ADN bases. If each base of this sequence is modelled by a discrete random variable taking values in $\{A; C; G; T\}$, the sequence may be modeled by a Markov chain.

Examples

- **Speech processing:** consider the syllables of a word represented as a random signal. To retrieve the words from the signals, we may use a hidden Markov model.
- **Text:** The text may be modelled by a vector whose components are the keyword appearance frequency, i.e. “bag of words” model. A *naive Bayes classifier* works well for spam detection although the order of the words and the correlation between the keywords frequencies are not taken into account.

Complexity vs. tractability

- Poor models are usually based on simple independence assumptions among variables that are rarely met in practice but they are easy to learn.
- In contrast, rich models allow complex statistical interactions to be captured but are difficult to learn (lack of data) and computationally demanding.
- In practice, one has to achieve a trade off for the model to be able to *generalize* well (statistical point of view) while keeping the computational burden of training and inference as low as possible (tractable computations).

Basic properties

Fundamental rules of probability. Let X and Y be two random variables,

- Sum rule:

$$p(X) = \sum_{Y} p(X, Y).$$

- Product rule:

$$p(X, Y) = p(Y|X)p(X).$$

Independence. X and Y are independent *iff*

$$p(X, Y) = p(X)p(Y).$$

Basic properties

Conditional independence. Let X, Y, Z be random variables.

- We define X and Y to be conditionally independent given Z if and only if

$$p(X, Y|Z) = p(X|Z)p(Y|Z).$$

- *Property:* If X and Y are conditionally independent given Z, then

$$p(X|Y, Z) = p(X|Z).$$

Basic properties

Independent and identically distributed. A set of random variables is independent and identically distributed (*i.i.d.*) if each variable has the same probability distribution and they are jointly independent.

Bayes formula. For two random variables X, Y we have

$$p(X|Y) = \frac{p(Y|X)p(X)}{p(Y)}.$$

Conditional independence

- Let X, Y and Z denote 3 disjoint sets of random variables defined on $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$,

Definition

X and Y are **conditional independent** given Z, denoted $X \perp Y \mid Z$, iff $\forall (x, y, z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ such that $p(z) > 0$:

$$p(x, y|z) = p(x|z)p(y|z)$$

- The condition is equivalent to

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

Conditional independence

- An alternative definition of conditional independence is:

Theorem

$X \perp Y \mid Z$ *iff there exists two functions f and g such that*

$$p(x, y, z) = f(x, z)g(y, z).$$

- Proof: $\Rightarrow$ holds trivially. To show the converse:

$$p(x, y, z)p(z) = f(x, z)g(y, z) \sum_{x', y'} f(x', z)g(y', z),$$

$$p(x, z)p(y, z) = f(x, z)\left(\sum_{y'} g(y', z)\right) g(y, z) \left(\sum_{x'} f(x', z)\right),$$

Independence models

- Conditional independences inferred from data by means of statistical independence tests can be used to learn the structure of probabilistic graphical models.
- An independence model has an axiomatic characterization or properties that allows to build formal deductive system.

Definition

An independence model I over a set V consists in a set of triples $\langle X; Y \mid Z \rangle$, called independence relations, where X, Y and Z are disjoint subsets of V.

- Equivalently, I_1 is a dependence map for I_2 (D-map), if $I_2 \quad I_1$. Finally,
- I_1 is a **perfect map** for I_2 (P-map), if $I_1 = I_2$.

Independence models

Definition

A probability distribution p defined over V is said **faithful** to an independence model I when all and only the independence relations in I hold in p, that is,

$$\langle X, Y \mid Z \rangle \in I \iff X \perp Y \mid Z \quad \text{w.r.t. } p.$$

- An independence model I is said probabilistic, if there exists a probability distribution p which is faithful to it.

Outline

Semi-graphoids

- Consider four mutually disjoint random variables, W, X, Y and Z. The following properties hold for any probability distribution:
  - Symmetry: $\langle X, Y \mid Z \rangle \iff \langle Y, X \mid Z \rangle$.
  - Decomposition: $\langle X, Y \cup W \mid Z \rangle \implies \langle X, Y \mid Z \rangle$.
  - Weak Union: $\langle X, Y \cup W \mid Z \rangle \implies \langle X, Y \mid Z \cup W \rangle$.
  - Contraction: $\langle X, Y \mid Z \rangle \wedge \langle X, W \mid Z \cup Y \rangle \implies \langle X, Y \cup W \mid Z \rangle$.
- Any independence model that respects these four properties is called a **semi-graphoid**.

Graphoids

- Another property holds in strictly positive distributions, that is when $p > 0$:
  - Intersection:
    $\langle X; Y \mid Z \cup W \rangle \wedge \langle X; W \mid Z \cup Y \rangle \implies \langle X; Y \cup W \mid Z \rangle$.
- Any independence model that respects these five properties is called a **graphoid**.

The characterization problem

- It is possible to detect contradictory conditional independence relations, by checking if they respect the semi-graphoid properties.
- Do the semi-graphoid properties provide a sufficient condition to characterize a probabilistic independence model? No!
- **Uncompleteness**: the graphoid axioms are insufficient to characterize probabilistic independence models.

The characterization problem

- The following set of probabistic CI relations (and their symetric counterparts) satisfies the graphoid axioms, yet does not have any faithful probability distribution:

$$\langle A, B \mid \{C, D\} \rangle \wedge \langle C, D \mid A \rangle \wedge \langle C, D \mid B \rangle \wedge \langle A, B \mid \emptyset \rangle.$$

- In fact, **no finite set** of CI properties characterizes the probabilistic independence models.
- Probabilistic independencies have no finite complete axiomatic characterization.

The characterization problem

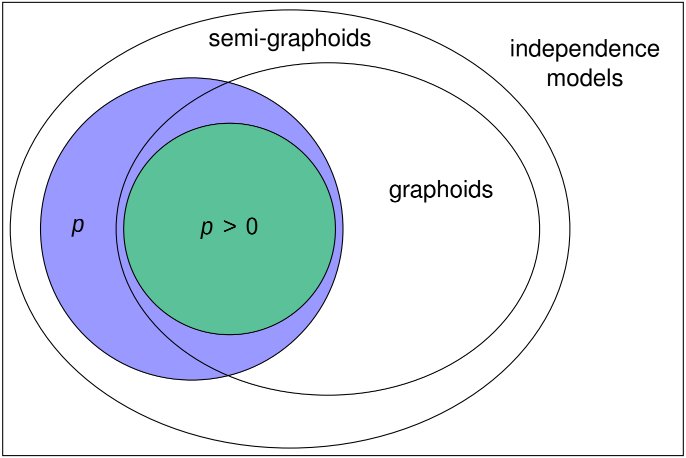


Figure: p denotes CI of probability distribution, and $p > 0$ stands for strictly positive probability distribution.

Outline

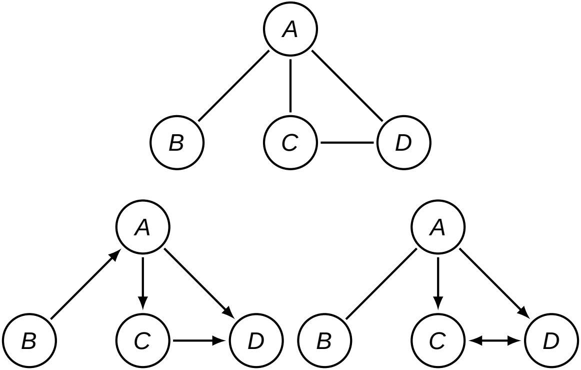
Probabilistic Graphical models

- A **probabilistic graphical model** (PGM) represents graphically a joint distribution.
- The nodes in the graph represent random variables, and the (lack of) edges represent conditional independence (CI) assumptions.
- Several useful properties:
  - Provide a simple way to visualize the probabilistic structure of a joint probability distribution.
  - Insights into the CI properties can be obtained by inspection of the graph.
  - Complex computations, required to perform inference and learning can be expressed in terms of graphical manipulations.
- Several kinds of PGMs: directed, undirected, mixed etc.

Graphs

- A graph G is an ordered pair of sets (V, E). $V = \{V_1, \dots, V_n\}$, are the *nodes* (or vertices), E represents the *edges*.
- A *clique* is a set of nodes such that each node is adjacent to every other node in the set. A *maximal clique* is a clique that does not accept any other clique as a proper superset.
- A *walk* between two nodes V_1 and V_k is a sequence of adjacent nodes in the form $V_1, \dots, V_k$. A walk with only distinct nodes is called a *path*. A path with $V_1 = V_k$ is called a *cycle*.
- A *complete graph* is a graph that has only one maximal clique.
- A *chordal graph* is a graph in which every cycle with more than 3 distinct nodes admits a smaller cycle as a proper subset.

Graphical models



- Directed, undirected and mixed graphs edges. The expressiveness of these models differ.

Probabilistic graphical models

- A PGM always consists in a set of parameters and a graphical structure G.
- G encodes a set of conditional independence relations between the variables and induces an independence model denoted $I(G)$. By definition, $I(G)$ is an I-map for p, that is,

$$\langle X; Y \mid Z \rangle \in I(G) \implies X \perp Y \mid Z \quad \text{w.r.t. } p.$$

- G allows explicit modelling of expert knowledge in the form of conditional independencies.
- And the edges provide a convenient way of communicating the investigator's beliefs of the causal influences among variables

Probabilistic graphical models

- A PGM is a compact graphical model for a joint distribution.
- The relationship between factorization, conditional independence, and graph structure comprises much of the power of the graphical modeling framework:
- The conditional independence viewpoint is most useful for designing models.
- The factorization viewpoint is most useful for designing inference algorithms.
- **Problems**: structure learning (G), parameter learning ( ), and inference using the model (e.g. $P(X \mid Y)$).

Outline

Markov Networks

- **Markov networks** (MNs), also called Markov random fields, are the most popular graphical models based on undirected graphs

Factorization

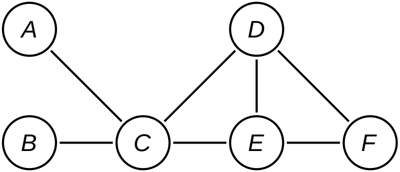
The probability distribution p factorizes as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{\mathbf{C} \in Cl_G} \psi_C(\mathbf{x}_C), \quad \text{with } Z = \sum_{\mathbf{x}} \prod_{\mathbf{C} \in Cl_G} \psi_C(\mathbf{x}_C)$$

- Cl_G is the set of all cliques in G. Z is called the *partition function*. Each ψ_i function is called a **factor**, a potential function, or a clique potential.
- $\psi_C(x_C) \geq 0$ ensures that $p(\mathbf{x}) \geq 0$.

u-separation

- $\langle \mathbf{X}, \mathbf{Y} \mid \mathbf{Z} \rangle$ belongs to $I(G)$ *iff* $\mathbf{Z}$ u-separates $\mathbf{X}$ and $\mathbf{Y}$ in G, that is, every path between a node in $\mathbf{X}$ and a node in $\mathbf{Y}$ contains a node in $\mathbf{Z}$.



- From the graph, we see that $\{A, B\} \perp \{D, E, F\} \mid C$ holds but not $\{A, B\} \perp F \mid E$.

Factorization

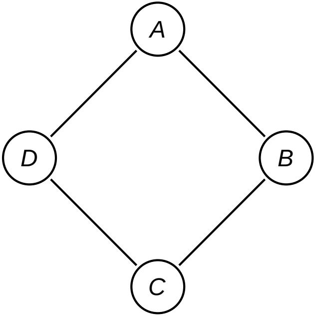
- Let G be an undirected graph over the random variables V, and p a probability distribution over V.
- G is an **I-map** for p if for all $X, Y, Z \in V$,

$$X \perp_G Y \mid Z \implies X \perp_P Y \mid Z.$$

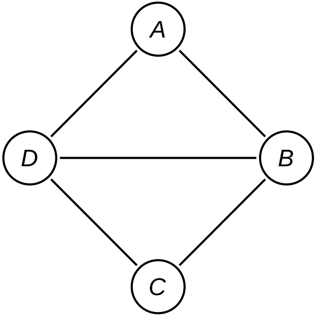
Theorem

$I(G)$ is an I-map for p if p factorizes into a product of potentials over the ***cliques*** *in G. The converse holds only if $p > 0$.*

Two undirected graphs



(d) Graph 1: $A \perp C \mid \{D, B\}$ and $B \perp D \mid \{A, C\}$.



(e) Graph 2: $A \perp C \mid \{D, B\}$

Clique potentials

- The factorization over the maximal cliques are for each Markov network:

$$\text{(Graph 1) } p(\mathrm{v}) = \;_1(a; b)\;_2(b; c)\;_3(c; d)\;_4(d; a)$$
$$\text{(Graph 2) } p(\mathrm{v}) = \;_1(a; b; d)\;_2(d; b; c)$$

- In the case of binary variables, we may define the clique potentials in the form of numerical tables.

Clique potentials

		B = 0	B = 1
A	0	2/3	3/3
	1	1/3	1/3

(a) $_1(a; b)$

		C = 0	C = 1
B	0	1/2	2/2
	1	1/2	3/2

(b) $_2(b; c)$

		D = 0	D = 1
C	0	3/3	2/3
	1	2/3	1/3

(c) $_3(c; d)$

		A = 0	A = 1
D	0	3/10	1/10
	1	1/10	2/10

(d) $_4(d; a)$

Clique potentials

A	D	$B = 0$	$B = 1$
0	0	1/8	3/8
	1	1/8	1/8
1	0	2/8	1/8
	1	2/8	2/8

(e) $_1(a; b; d)$

C	D	$B = 0$	$B = 1$
0	0	1/6	4/6
	1	1/6	1/6
1	0	1/6	2/6
	1	2/6	3/6

(f) $_2(b; c; d)$

Clique potentials

- These potentials are valid, i.e. $\sum_{a,b,c,d} p(a, b, c, d) = 1$.
- However individual clique potentials **do not necessarily sum to 1**, and therefore do not necessarily correspond to marginal or conditional probability distributions.
- Potential functions in Markov network do not lend to an intuitive probabilistic interpretation. One must go through a factorization to obtain a proper probability measure.
- Potential functions are often expressed as exponential parametric functions for practical reasons.

Clique potentials

- Not every probability distribution is UG-faithful.
- The probabilstic IC relation $X \perp Y$ and $X \not\perp Y \mid Z$ cannot be faithfully represented an UG model because $X \perp Y \mid \emptyset$ necessarily implies $X \perp Y \mid Z$ (strong union property) in UG models.
- As a result, the only undirected graph that is an I-map for p is the complete graph, which necessarily results in 7 free parameters instead of 6, as $p(x, y, z) = p(x)p(y)p(z \mid x, y)$.
- A Markov network is not perfectly suited to encode p in this situation.

Markov blanket

- The notions of *Markov blanket* and *Markov boundary* are essential in feature selection..

Definition

A *Markov blanket* of $\mathbf{X}$ in $\mathbf{V}$ is a subset $\mathbf{M} \subseteq (\mathbf{V} \setminus \mathbf{X})$ such that $\mathbf{X} \perp \mathbf{V} \setminus (\mathbf{X} \cup \mathbf{M}) \mid \mathbf{M}$. A *Markov boundary* is an inclusion-optimal Markov blanket, i.e., none of its proper subsets is a Markov blanket.

- In a faithful UG the Markov boundary of a variable X is unique and is given by neighbors of X.

Image denoising using MRF

- An UG model representing a Markov random field for image de-noising,
- x_i is a binary variable denoting the state of pixel i in the unknown noise-free image,
- y_i denotes the corresponding value of pixel i in the observed noisy image.

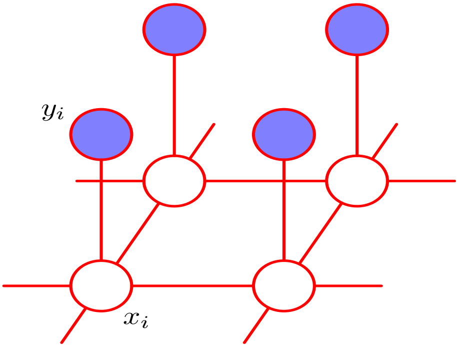


Image denoising using MRF

- Because a potential function is an arbitrary, non-negative function over a maximal clique, we may define a joint distribution over x and y by

$$p(\mathrm{x},\mathrm{y}) = \frac{1}{Z}\exp\{\quad E(\mathrm{x},\mathrm{y})\}$$

- Neighbouring pixels are correlated and only a small percentage of the pixels are corrupted.
- We want the energy to be lower when $\{x_i, x_j\}$ and $\{x_i, y_i\}$ have the same sign than when they have the opposite sign.
- The complete energy function takes the form

$$E(\mathrm{x},\mathrm{y}) = h\sum_i x_i \quad \sum_{\{i,j\}\in E} x_i x_j \quad \sum_i x_i y_i$$

Image denoising using MRF

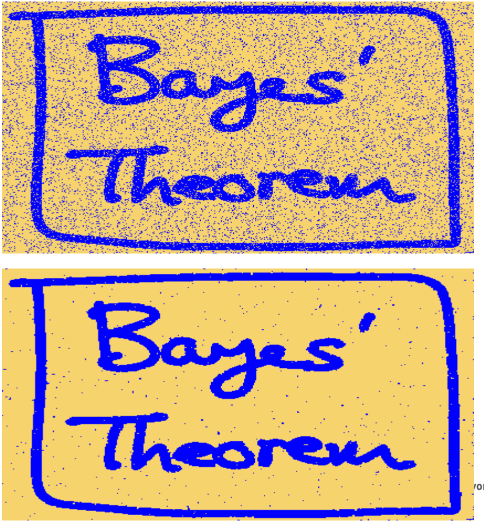
- Given y are the (observed) pixels of the noisy image, one has to solve the MAP:

$$\arg\max_{x} p(\mathbf{x};\mathbf{y}) = \arg\max_{x} \exp\{-E(\mathbf{x};\mathbf{y})\}$$

- A local maximum can be easily obtained by simple coordinate-wise gradient ascent methods.
- This is an example of the Ising model which has been widely studied in statistical physics.

Image denoising using MRF

- Illustration of image de-noising using a Markov random field (Besag, 1974).
- On the top, the corrupted image after randomly changing 10% of the pixels. On the bottom, the restored images obtained using iterated conditional models (ICM)



Outline

Bayesian networks

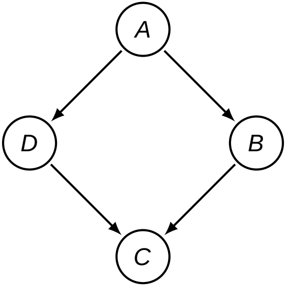
Definition

A **Bayesian network** consists in a set of random variables $\mathbf{V} = \{V_1, \ldots, V_n\}$, a simple directed acyclic graph $G = (\mathbf{V}, E)$, and a set of parameters . Together, G and define a probability distribution p over $\mathbf{V}$ which factorizes as:

$$p(\mathbf{v}) = \prod_{V_i \in \mathbf{V}} p(v_i | \mathrm{pa}_{V_i}).$$

- pa_{V_i} denotes the *parents* of node V_i in G.
- are the probabilities $p(v_i | \mathrm{pa}_{V_i})$.

Illustration



The corresponding factorization is

$$p(\mathbf{v}) = p(a)p(d|a)p(b|a)p(c|b,d)$$

.

Conditional probability tables

$A=0$	$A=1$
0.4	0.6

(g) $p(a)$

A	$D=0$	$D=1$
0	0.6	0.4
1	0.5	0.5

(h) $p(d|a)$

A	$B=0$	$B=1$
0	0.3	0.7
1	0.1	0.9

(i) $p(b|a)$

B	D	$C=0$	$C=1$
0	0	0.8	0.2
	1	0.7	0.3
1	0	0.5	0.5
	1	0.7	0.3

(j) $p(c|b,d)$

Table: A set of conditional probability tables that define a valid set of parameters for the Bayesian network structure.

Conditional probability tables

- These tables define valid conditional probability distributions that can be intuitively interpreted.
- Each of the factors $p(v_i | \mathrm{pa}_{V_i})$ can be seen as a potential function ${}_i(v_i; \mathrm{pa}_{V_i})$ in a Markov network.
- In a Bayesian network, each factor defines a conditional probability distribution for V_i, and thus respects the normalization constraint $\sum_{v_i} {}_i(v_i; \mathrm{pa}_{V_i}) = 1$.

Parametric conditional distributions

- The number of parameters required to specify a PCT grows exponentially with M the number of parents.
- A more parsimonious form uses a logistic sigmoid function acting on a linear combination of the parents. Consider a graph comprising M parents $x_1, \ldots, x_M$ and a single child y,

$$p(y = 1 \mid x_1, \ldots, x_M) = \sigma\left(w_0 + \sum_{i=1}^{M} w_i x_i\right)$$

- $\sigma(a) = (1 + \exp(-a))^{-1}$ is the **sigmoid function** and $w = (w_0, w_1 \ldots, w_M)^T$ is a vector of $M + 1$ parameters.
- The conditional distribution is now governed by a number of parameters that **grows linearly** with M.

Linear-Gaussian models

- A **multivariate Gaussian** can be expressed as a directed graph corresponding to a linear-Gaussian model.
- Examples of linear-Gaussian models: probabilistic principal component analysis, factor analysis, and linear dynamical systems.
- If node i represents a continuous random variable X_i having a Gaussian distribution of the form,

$$p(x_i \mid pa_i) = \mathcal{N}\left(x_i \mid \sum_{j \in pa_i} w_{ij} x_j + b_i, \sigma_i^2\right)$$

- w_{ij} and b_i are parameters governing the mean, and σ_i^2 is the variance of the conditional distribution.
- The mean and covariance of the joint distribution are determined recursively.

d-separation

- Every DAG G induces a formal independence model $I(G)$ over $\mathbf{V}$, by means of a graphical separation criterion called *d*-separation
- Within a path $V_1, \ldots, V_k$, an intermediate node V_i is said to be a *collider* is an intermediate node V_i in the form $V_{i-1} \to V_i \leftarrow V_{i+1}$ called a *v-structure*.
- *d*-separation is equivalent to *u*-separation when G contains no *v-structure*

d-separation

- Let $\mathbf{X} \perp_G \mathbf{Y} \mid \mathbf{Z}$ denotes a CI relation encoded in a DAG G.

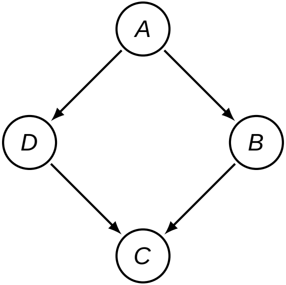
Definition

For any disjoint set of random variables $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$, $\mathbf{X} \perp_G \mathbf{Y} \mid \mathbf{Z}$ *iff* $\mathbf{Z}$ *d*-separates $\mathbf{X}$ and $\mathbf{Y}$ in G, that is, every path between $\mathbf{X}$ and $\mathbf{Y}$ contains

- a non-collider that belongs to $\mathbf{Z}$,
- or a collider that does not belong to $\mathbf{Z} \cup \mathbf{AN}_{\mathbf{Z}}$.

- $\mathbf{AN}_{\mathbf{Z}}$ are the ancestors of nodes $\mathbf{Z}$.

d-separation

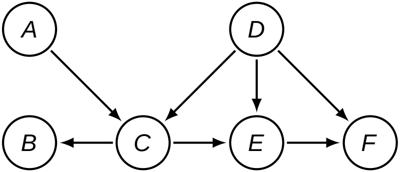


We have $A \perp C \mid \{D, B\}$ and $D \perp B \mid A$ because $D \rightarrow C \leftarrow B$ is a closed path.

d-separation

- A friendly interpretation of *d*-separation is to conider a path as an information flow.
- Consider a path between X and Y, and a conditioning set Z.
- When Z is empty, each intermediate node that is not a collider is open, that is, it lets the flow go through. Conversely, each intermediate node that is a collider is closed, and blocks the flow.
- The variables in Z change the state of the nodes, i.e. from open to closed and vice-versa.
- If Z *d*-separates X and Y, all the paths between X and Y are closed.

d-separation (again)



d-separation

- $A \perp B \mid C$ because the only path $A \to C \to B$ is closed by C that is observed.
- However, $A \not\perp \{B, F\} \mid \{C, E\}$ because in the path $A \to C \leftarrow D \to F$ the non-collider D is open, as well as the collider C that is observed.
- $A \not\perp F \mid \emptyset$, because of the open path $A \to C \to E \to F$.
- Conditioning on E does not *d*-separate A and F either, it closes the previous path but opens a new one with $A \to C \to E \leftarrow D$.
- To close all paths, it is sufficient to condition on $\{C, D\}$, E it no longer necessary in the conditioning set.

Markov blanket

- The notions of *Markov blanket* and *Markov boundary* are essential in feature selection..

Definition

A *Markov blanket* of $\mathbf{X}$ in $\mathbf{V}$ is a subset $\mathbf{M} \subseteq (\mathbf{V} \setminus \mathbf{X})$ such that $\mathbf{X} \perp \mathbf{V} \setminus (\mathbf{X} \cup \mathbf{M}) \mid \mathbf{M}$. A *Markov boundary* is an inclusion-optimal Markov blanket, i.e., none of its proper subsets is a Markov blanket.

- In a faithful DAG the Markov boundary of a variable X is unique and is given by $\mathbf{MB}_X = \mathbf{PC}_X \cup \mathbf{SP}_X$, that is, the parents, children and spouses of X.

Markov property

- As with Markov networks, a Bayesian network structure always defines an I-map of the underlying probability distribution.

Theorem

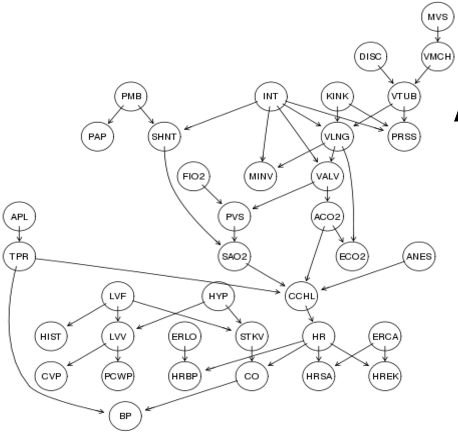
Let $\mathcal{G}$ be a DAG, $I(\mathcal{G})$ is an I-map for p iff p factorizes recursively over $\mathcal{G}$.

Local Markov property

- From the *d*-separation, **every node is independent of its non-descendants given its parents** (a.k.a. **local Markov property**), that is, $V_i \perp ND_{V_i} \setminus PA_{V_i} \mid PA_{V_i}$.
- Because G is a DAG, we may arrange its nodes in a topological ordering $V_1, \ldots, V_n$ according to G, that is, $i < j$ if $V_i \rightarrow V_j$ is in G.
- From the chain rule of probabilities, we show that

$$
\begin{aligned}
p(\mathbf{v}) &= \prod_{i=1}^{n} p(v_i \mid v_1, \ldots, v_{i-1}) \\
&= \prod_{i=1}^{n} p(v_i \mid \mathrm{pa}_{V_i})
\end{aligned}
$$

Famous networks used as benchmarks



ALARM

- 37 nodes
- 46 arcs
- 509 parameters

Graphs with three nodes

- A **Markov chain** is a particular DAG.
- We have $X \not\perp Y \mid \emptyset$:

$$\begin{aligned} p(x, y) &= \sum_z p(x)p(z|x)p(y|z) \\ &= p(x) \sum_z p(z|x)p(y|z) \\ &= p(x)p(y|x) \\ &\neq p(x)p(y) \end{aligned}$$

Graphs with three nodes

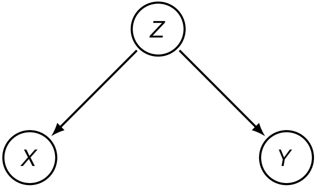
- A **Markov chain** is a special DAG.
- We verify that $X \perp Y \mid Z$:

$$
\begin{aligned}
p(y|z,x) &= \frac{p(x,y,z)}{p(x,z)} \\
&= \frac{p(x,y,z)}{\sum_{y'} p(y',x,z)} \\
&= \frac{p(x)p(z|x)p(y|z)}{\sum_{y'} p(x)p(z|x)p(y'|z)} \\
&= p(y|z)
\end{aligned}
$$

Graphs with three nodes

- Z is a **latent cause**.
- We verify that $X \not\perp Y \mid \emptyset$:

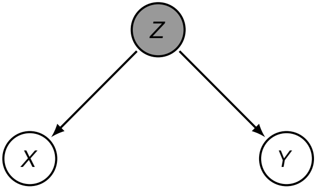
$$
\begin{aligned}
p(x, y) &= \sum_z p(z)p(x|z)p(y|z) \\
&\neq p(x)p(y)
\end{aligned}
$$



Graphs with three nodes

- Z is a **latent cause**.
- We verify that $X \perp Y \mid Z$:

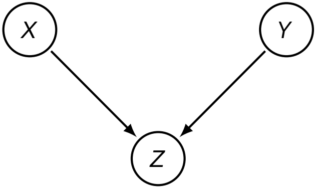
$$
\begin{aligned}
p(x, y|z) &= \frac{p(x, y, z)}{p(z)} \\
&= \frac{p(z)p(y|z)p(x|z)}{p(z)} \\
&= p(x|z)p(y|z)
\end{aligned}
$$



Graphs with three nodes

- **Explaining away** or V-structure.
- We verify that $X \perp Y \mid \emptyset$:

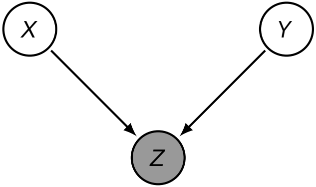
$$
\begin{aligned}
p(x,y) &= \sum_z p(x,y,z) \\
&= p(x)p(y) \sum_z p(z|x,y) \\
&= p(x)p(y)
\end{aligned}
$$



Graphs with three nodes

- **Explaining away** or V-structure.
- We verify that $X \not\perp Y \mid Z$:

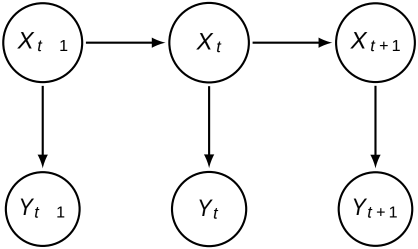
$$
\begin{aligned}
p(x, y \mid z) &= \frac{p(x, y, z)}{p(z)} \\
&= \frac{p(x)p(y)p(z \mid x, y)}{p(z)} \\
&\neq p(x \mid z)p(y \mid z)
\end{aligned}
$$



Hidden Markov chain

- A **Hidden Markov Model** is a dynamic Bayesian network. Often used because we only have a noisy observation of the random process.
- Y_t are the **visible variables**, and X_t the **hidden variables**.
- We have:

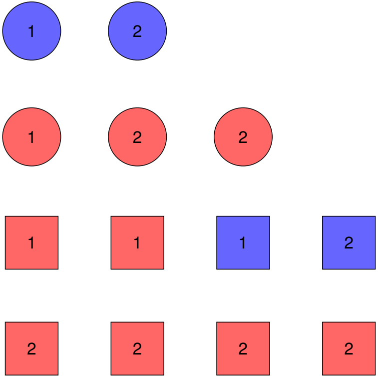
$$X_{t+1} \perp X_{t-1} \mid X_t$$
$$Y_{t+1} \perp Y_t \mid X_t$$



Outline

Toy problem 1

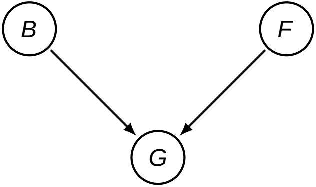
- Consider a bag containing the following tokens:



- Show that Value ? Form j Color.
- Build all the faithful DAGs of $p(V; C; F)$.
- Learn the parameters.
- Compute $P(V = 1\ j\ F =$ square) using the model.

Toy problem 2

- Consider a car with **B**attery (0=flat, 1=fully charged), **F**uel tank (0=empty, 1=full) and Fuel **G**auge reading (0=empty, 1=full).
- Assume that $B \perp F \mid \emptyset$.



$$
\begin{aligned}
P(G = 1 \mid B = 1, F = 1) &= 0.8 \\
P(G = 1 \mid B = 1, F = 0) &= 0.2 \\
P(G = 1 \mid B = 0, F = 1) &= 0.2 \\
P(G = 1 \mid B = 0, F = 0) &= 0.1 \\
P(B = 1) &= 0.9 \\
P(F = 1) &= 0.9
\end{aligned}
$$

Compute:

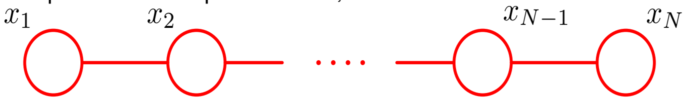
- $p(F = 0 \mid G = 0)$
- $p(F = 0 \mid G = 0, B = 0)$

Outline

Relation between DAG and UG

- A chain DAG,

- Its equivalent UG representation,



Relation between DAG and UG

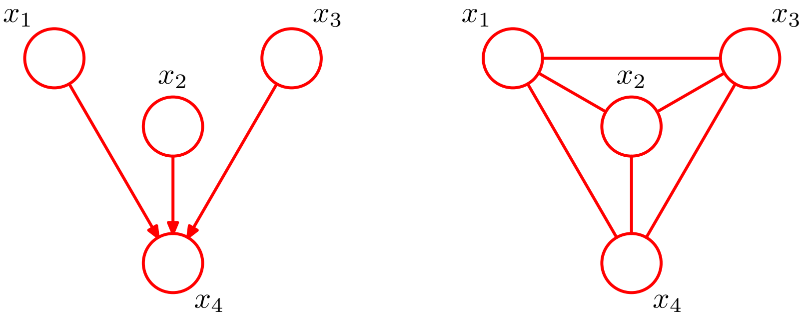
$$
\begin{aligned}
p(x) &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)\ldots p(x_N \mid x_{N-1}) \\
&= \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\ldots\psi_{N-1,N}(x_{N-1}, x_N).
\end{aligned}
$$

- This is easily done by identifying,

$$
\begin{aligned}
\psi_{1,2}(x_1, x_2) &= p(x_1)p(x_2 \mid x_1) \\
\psi_{2,3}(x_2, x_3) &= p(x_3 \mid x_2) \\
&\vdots \\
\psi_{N-1,N}(x_{N-1}, x_N) &= p(x_N \mid x_{N-1})
\end{aligned}
$$

- The maximal cliques in the UG are the pairs of neighbouring nodes in the DAG. In this case, Z = 1.

Relation between DAG and UG



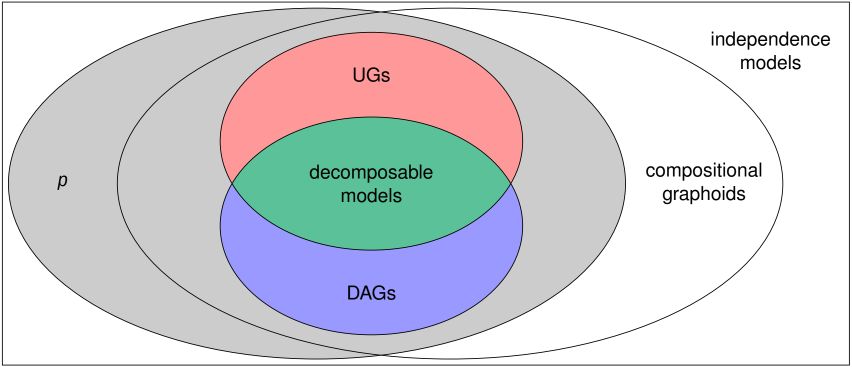
- The process of 'marrying the parents' is known as moralization, and the resulting undirected graph, after dropping the arrows, is called the "moral graph".
- Graph moralization plays an important role in exact inference techniques such as the *junction tree algorithm*.

Relation between DAG and UG

$$
\begin{aligned}
p(x) &= p(x_1)p(x_2)p(x_3)p(x_4 \mid x_1, x_2, x_3) \\
&= \frac{1}{Z} \; {}_{1,2,3,4}(x_1, x_2, x_3, x_4).
\end{aligned}
$$

- In going from a directed to an undirected representation we had to discard some CI properties from the graph (e.g. $X_1 \perp X_2 \mid X_4$).
- It turns out that the two types of graph can express different CI properties.

PGM's expressiveness



Noisy XOR

A	*B*	*C*	
		(1)=4	=4
		=4	(1)=4
		=4	(1)=4
		(1)=4	=4

- $p(a; b; c)$ for the noisy XOR (exclusive OR) relationship

$$P(A = B \quad C) = 1$$

- $p > 0$ for any $\quad 2]0; 1{=}2[\,[\,]1{=}2; 0[$.

Noisy XOR

- We have A ? B, B ? C and C ? A.
- Due to the strong union property, A ? B =) A ? B j C, no undirected graph that can encode any of the independence relations in p
- p is not UG-faithful. The complete graph is the I-map.
- The Markov network model requires 7 free parameters to encode p.

Noisy XOR

- p is not DAG-faithful either. Due to the composition property, any DAG that encodes two of the independence relations in p necessarily breaks a dependence relation as well ($A \perp B \wedge A \perp C \implies A \perp \{B, C\}$).
- The DAG $A \rightarrow C \leftarrow B$ encodes only one of the independence relation. This BN structure results in the factorization $p(a, b, c) = p(a)p(b)p(c|a, b)$, which encodes p with 6 free parameters.
- In this example p is neither UG-faithful nor DAG-faithful, so both Markov networks and Bayesian networks are not well-suited models to encode p efficiently.
- Yet, p can be encoded efficiently with only 4 parameters.

Extensions

- Classical PGMs have a limited expressive power as independence models.
- Over the years, many alternative PGMs have been proposed to overcome these limitations, by extending and unifying UGs and DAGs.
  - Ancestral graphs, Anterial graphs, LWF chain graphs, AMP chain graphs$\dots$
  - Four types of edges are allowed: directed edges and three types of undirected edges.
- Increased expressive power comes at the expense of an increased complexity.
- Factorization of p? Practical parametrization of the model? Learning and inference?

Outline

The 4 basic problems with PGMs

There are 4 basic problems to be solved for the model to be useful in real-world applications:

1. Problem 1: **Inference**. Given some observation, compute the conditional distribution of the remaining variables (NP-hard if loops in the graph).
2. Problem 2: **MAP Inference**. Find the MAP over this conditional distribution (NP-hard).
3. Problem 3: **Learning**. Given a sequence of observations, estimate the MAP of the **parameters** (Easy problem with a complete data set).
4. Problem 4: **Learning**. Given a sequence of observations, learn the **topological structure** of the PGM (NP-hard).

Problem 1: Inference

- Suppose we have a set of correlated random variables with joint distribution $p(x_1, \ldots, x_N | \;)$.
- Let us partition this vector into the **visible variables** $\mathbf{x}_v$, which are observed, and the **hidden variables**, $\mathbf{x}_h$, which are unobserved.
- Inference refers to computing the posterior distribution of the unknowns given the evidence:

$$p(\mathbf{x}_h | \mathbf{x}_v, \;) = \frac{p(\mathbf{x}_h, \mathbf{x}_v | \;)}{p(\mathbf{x}_v | \;)} = \frac{p(\mathbf{x}_h, \mathbf{x}_v | \;)}{\sum_{\mathbf{x}'_h} p(\mathbf{x}'_h, \mathbf{x}_v | \;)}$$

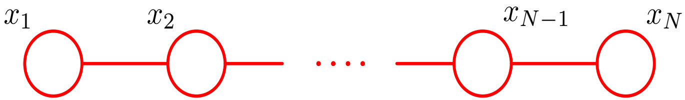
Problem 1: Inference

- Sometimes only some of the hidden variables are of interest.
- Let's partition the hidden variables into **query variables**, X_q, whose value we wish to know, and the remaining **nuisance variables**, X_n, which we are not interested in.
- We can compute what we are interested in by **marginalizing out** the nuisance variables:

$$p(x_q | x_v, \ ) = \sum_{\mathbf{x}_n} p(x_q, x_n | x_v, \ )$$

Outline

Problem 1: Inference in a chain



- Exact inference on a graph comprising a chain of nodes can be performed efficiently in time that is linear in the number of nodes.

$$p(x) = \frac{1}{Z} \; {}_{1;2}(x_1; x_2) \; {}_{2;3}(x_2; x_3) ::: \; {}_{N \;\; 1;N}(x_{N \;\; 1}; x_N):$$

- The algorithm that can be interpreted in terms of messages passed along the chain.

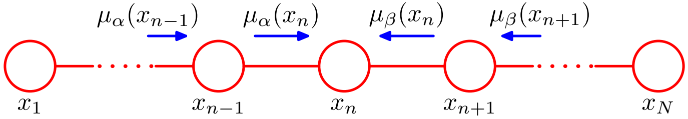
Problem 1: Inference in a chain

- Consider the inference problem of finding the marginal distribution $p(x_n)$

$$
\begin{aligned}
p(x_n) &= \sum_{x_1} \cdots \sum_{x_{n-1}} \sum_{x_{n+1}} \cdots \sum_{x_N} p(\mathbf{x}) \\
&= \frac{1}{Z} \left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \cdots \right] \\
&\quad \left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right] \\
&= \frac{1}{Z} \mu_\alpha(x_n) \, \mu_\beta(x_n).
\end{aligned}
$$

Problem 1: Inference in a chain

- The algorithm that can be interpreted in terms of messages passed along the chain.



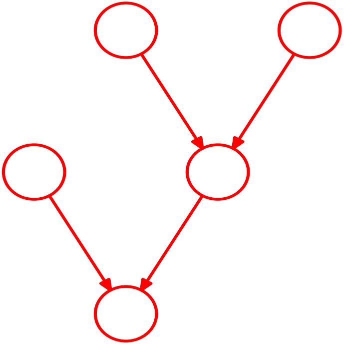
- With N discrete variables each having K states, the messages (x_n) and (x_n) can be evaluated recursively in $O(NK^2)$. by exploiting the IC properties of this simple graph in order to obtain an efficient calculation.
- This is linear in the length of the chain, in contrast to the exponential cost of a naive approach.

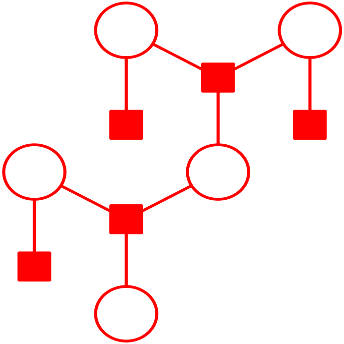
Exact inference in a (poly)tree

- As for chains, inference can be performed efficiently using local message passing in **trees and polytrees**.
- A polytree is a directed acyclic graph whose underlying undirected graph is a tree.
- The message passing formalism is also applicable to undirected and directed trees and to polytrees. It is called the **sum-product algorithm**.
- It requires a graphical construction called a **factor graph**.

Factor graphs

- First transform the PGM into a **factor graph**:

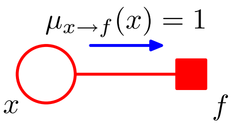


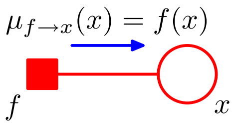


Outline

Sum-product algorithm

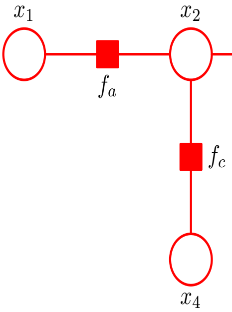
- Start form the leaves:





Sum-product algorithm

- Consider a simple example to illustrate the operation of the sum-product algorithm:

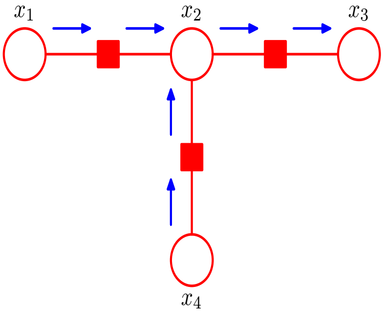


$$p(x) = \frac{1}{Z} f_a(x_1; x_2) f_b(x_2; x_3) f_c(x_2; x_4):$$

Sum-product algorithm

- Say node x_3 is the root node. Start from the leaf nodes x_1 and x_4 towards the root x_3 and perform the following sequence of messages:

$$
\begin{aligned}
{}_{x_1 \to f_a}(x_1) &= 1 \\
{}_{f_a \to x_2}(x_2) &= \sum_{x_1} f_a(x_1, x_2) \\
{}_{x_4 \to f_c}(x_4) &= 1 \\
{}_{f_c \to x_2}(x_2) &= \sum_{x_4} f_c(x_2, x_4) \\
{}_{x_2 \to f_b}(x_2) &= {}_{f_a \to x_2}(x_2) \; {}_{f_c \to x_2}(x_2) \\
{}_{f_b \to x_3}(x_3) &= \sum_{x_2} f_b(x_2, x_3) \; {}_{x_2 \to f_b}(x_2)
\end{aligned}
$$



Sum-product algorithm

- Then, from the root node towards the leaf nodes:

$$x_3 \to f_b(x_3) = 1$$

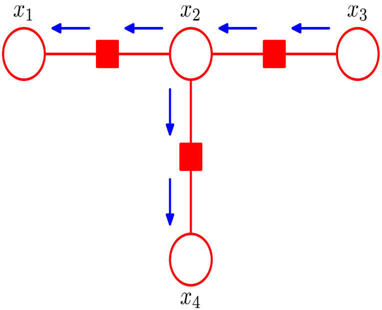
$$f_b \to x_2(x_2) = \sum_{x_3} f_b(x_2, x_3)$$

$$x_2 \to f_a(x_2) = f_b \to x_2(x_2) \quad f_c \to x_2(x_2)$$

$$f_a \to x_1(x_1) = \sum_{x_2} f_a(x_1, x_2) \quad x_2 \to f_a(x_2)$$

$$x_2 \to f_c(x_2) = f_a \to x_2(x_2) \quad f_b \to x_2(x_2)$$

$$f_c \to x_4(x_4) = \sum_{x_2} f_c(x_2, x_4) \quad x_2 \to f_c(x_2)$$



Sum-product algorithm

- One message has now passed in each direction across each link,
- To evaluate the marginals:

$$
\begin{aligned}
p(x_2) &= \frac{1}{Z}\, {}_{f_a \to x_2}(x_2)\; {}_{f_b \to x_2}(x_2)\; {}_{f_c \to x_2}(x_2) \\
&= \frac{1}{Z}\left[\sum_{x_1} f_a(x_1, x_2)\right]\left[\sum_{x_3} f_b(x_2, x_3)\right]\left[\sum_{x_4} f_c(x_2, x_4)\right] \\
&= \frac{1}{Z}\sum_{x_1}\sum_{x_3}\sum_{x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4) \\
&= \frac{1}{Z}\sum_{x_1}\sum_{x_3}\sum_{x_4} p(\mathbf{x})
\end{aligned}
$$

Outline

Problem 2: MAP Inference

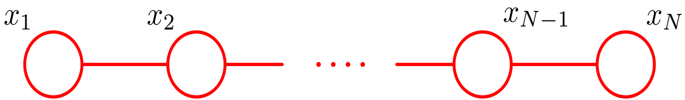
- Suppose we have a set of correlated random variables with joint distribution $p(x_1, \ldots, x_N | \;)$.
- Let us partition this vector into the **visible variables** $\mathbf{x}_v$, which are observed, and the **hidden variables**, $\mathbf{x}_h$, which are unobserved.
- MAP Inference refers to computing the MAP of the posterior distribution:

$$\mathbf{x}_h^{\star} = \arg\max_{\mathbf{x}_h} p(\mathbf{x}_h | \mathbf{x}_v, \;)$$

Problem 2: The max-sum algorithm

- The sum-product algorithm takes a joint distribution $p(\mathrm{x})$ expressed as a factor graph and efficiently find marginals over the component variables.
- MAP inference: find a setting of the variables that has the largest probability and give the probability.
- This can be addressed through a closely related algorithm called **max-sum algorithm**, which can be viewed as an application of dynamic programming in the context of graphical models

Problem 2: MAP inference in a chain



- MAP inference on a graph comprising a chain of nodes can be performed efficiently in time that is linear in the number of nodes.

$$p(x) = \frac{1}{Z} \max_{x_1,\dots,x_N} [\psi_{1,2}(x_1,x_2)\,\psi_{2,3}(x_2,x_3)\cdots\psi_{N-1,N}(x_{N-1},x_N)]$$

$$= \frac{1}{Z} \max_{x_1} \max_{x_2} \psi_{1,2}(x_1,x_2) \cdots \max_{x_N} \psi_{N-1,N}(x_{N-1},x_N) \quad .$$

- The structure of this calculation is identical to that of the sum-product algorithm,
- Application: find the most probable sequence of hidden states in a HMM, known as the Viterbi algorithm.

Exact inference in general graphs

- The message passing framework (i.e. sum-product and max-sum algorithms) can be generalized to graphs having loops, using the **junction tree algorithm** (Lauritzen et al., 1988).
- A DAG is first converted to an UG by moralization, (not required for an UG).
- Next the graph is triangulated, i.e. adding extra links to eliminate chord-less cycles containing four or more nodes.

The junction tree algorithm

- Then, construct a tree-structured undirected graph called a join tree, whose nodes correspond to the maximal cliques of the triangulated graph, and whose links connect pairs of cliques that have variables in common.
- The selection of which pairs of cliques to connect in this way is important and is done so as to give a maximal spanning tree .
- If the number of variables in the largest clique is high, the junction tree algorithm becomes impractical.

Approximate inference in general graphs

- For many problems of practical interest, it is not be feasible to use exact inference, effective approximation methods are needed.
- A simple idea to approximate inference in graphs with loops is to apply the sum-product algorithm as it is.
- This approach is known as loopy belief propagation (Frey and MacKay, 1998) and is possible because the message passing rules are purely local, even though there is no guarantee that it will yield good results.

Outline

The 4 basic problems with PGMs

There are 4 basic problems to be solved for the model to be useful in real-world applications:

1. Problem 1: Inference . Given some observation, compute the conditional distribution of the remaining variables (NP-hard if loops in the graph).
2. Problem 2: MAP Inference . Find the MAP over this conditional distribution (NP-hard).
3. Problem 3: Learning . Given a sequence of observations, estimate the MAP of the parameters (Easy problem with a complete data set).
4. Problem 4: Learning . Given a sequence of observations, learn the topological structure of the PGM (NP-hard).

Problem 3: Parameter learning

- Find the MAP estimate for the parameters:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{N} \log p(x_i \mid \theta) + \log p(\theta)$$

- $p(\theta)$ is the prior on the parameters.

Problem 3: Learning from complete data

- If all the variables are fully observed (i.e. no missing data and no hidden variables), the data is complete .
- For a DGM with complete data, the likelihood is given by

$$
\begin{aligned}
p(x \mid \theta) &= \prod_{i=1}^{N} p(x_i \mid \theta) \\
&= \prod_{i=1}^{N} \prod_{t=1}^{V} p(x_{it} \mid x_{i,pa(t)}, \theta_t) \\
&= \prod_{t=1}^{V} p(D_t \mid \theta_t)
\end{aligned}
$$

- D_t is the data associated with node t and its parents.

Problem 3: Learning from complete data

- Now suppose that the prior factorizes as well:

$$p(\theta) = \prod_{t=1}^{V} p(\theta_t)$$

- Then clearly the posterior also factorizes:

$$p(\theta|D) \propto p(D|\theta)p(\theta) = \prod_{t=1}^{V} p(D_t|\theta_t)p(\theta_t)$$

Problem 3: Learning with missing and/or latent variables

- If we have missing data and/or hidden variables, the likelihood no longer factorizes, and indeed it is no longer convex.
- This means we will usually can only compute a locally optimal ML or MAP estimate.
- Bayesian inference of the parameters is even harder and requires suitable approximate inference techniques.

Problem 4: structure learning

- Given a sequence of observations, learn the topological structure of the PGM (NP-hard). The problem of learning a BN structure has attracted much attention.
- Problem: the number of possible DAGs with n variables is superexponential w.r.t n. For instance, NS(5) = 29281 and NS(10) = 4.2×10^{18}.
- Search-and-score methods search over a space of structures employing a scoring function to guide the search. The most prominent algorithm in this class is the Greedy Equivalent Search (GES).
- Constraint-based algorithms use statistical independence tests to impose constraints on the network structure and infer the final DAG. PC is prototypical constraint-based algorithm.

PC algorithm

Start End

PC algorithm

End

True graph

Outline

Markov Models

- A Markov model is a stochastic model used to model randomly changing systems where it is assumed that future states depend only on the current state not on the events that occurred before it.
- This assumption is called the:

Markov property

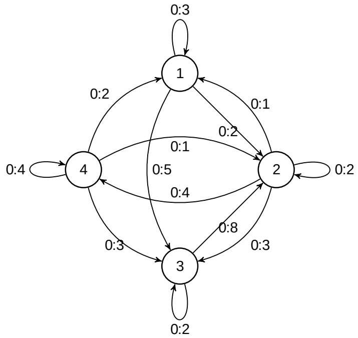
$$P(X_{n+1} = j | X_1 = i_1, \ldots, X_n = i_n) = P(X_{n+1} = j | X_n = i_n)$$

Markov chains

- The Markov property enables reasoning and computation with the model that would otherwise be intractable.
- If $P(X_{n+1} = j | X_n = i_n) = p_{ij}$ does not depend on n then the Markov model is homogeneous .
- The simplest Markov model is the Markov chain . It models the state of a system with a random variable that changes through time.

Markov chains

Markov chain



Transition matrix

$$P = \begin{pmatrix} 0.3 & 0.2 & 0.5 & 0 \\ 0.1 & 0.2 & 0.3 & 0.4 \\ 0 & 0.8 & 0.2 & 0 \\ 0.2 & 0.1 & 0.3 & 0.4 \end{pmatrix}$$

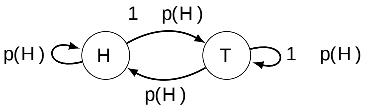
Such that $\forall i, \sum_j P_{ij} = 1$

Coin toss Models

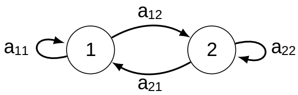
- Someone is performing coin tosses in a room. He tells you the result of the coin ips, nothing else (e.g. probability of heads, number of coins, transition probabilities).
- We only observe a sequence of heads (H) and tails (T).
- Which model (1 or 2 coins) best matches the observations?

Two scenarios:

Single coin



Two coins



$P(H) = p_1$ $P(H) = p_2$

Hidden Markov models

- A hidden Markov model (HMM) is a Markov model in which the system is assumed to be a Markov process with unobserved (hidden) states.
- In simpler Markov models (like a Markov chain), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters.
- In a HMM, the output, dependent on the state, is visible. Each state has a probability distribution over the possible outputs . Therefore, the sequence of outputs generated by an HMM gives some information about the sequence of state.
- Many applications in temporal pattern recognition such as speech, handwriting, gesture recognition, and bioinformatics.

Hidden Markov models

- The random variable q_t is the hidden state at time t. which is assumed to consist of one of N possible values $\{s_1, \dots, s_n\}$, modeled as a categorical distribution.
- The random variable O_t is the observation at time t (with $y(t) \in \{y_1, y_2, y_3, y_4\}$). O_t is typically a letter from an alphabet of M symbols $V = \{v_1, \dots, v_M\}$.
- In the standard HMM, the state space is discrete, while the observations themselves can either be discrete or continuous (e.g. Gaussian distribution).
- The parameters of a hidden Markov model are of two types: N^2 transition probabilities and NM emission probabilities (also known as output probabilities).

Example: Urn and ball model

Discrete symbol HMM

Notations

A complete specification of an HMM is given its paremeters $= (A; B;)$ and is defined by:

- Its n hidden states $S = \{s_1; \ldots; s_n\}$.
- The M observable symbols $V = \{v_1; \ldots; v_M\}$. O_t denotes the symbol at time t.
- The state transition matrix $a_{ij} = A(i; j)$
- The observation symbol probability distribution B: $b_j(k) = P(O_t = v_k | q_t = s_j)$ with $\sum_{k=1}^{M} b_j(k) = 1$
- The initial state distribution $= \{ {}_j \}_{j=1;\ldots;n}$ where ${}_j = P(q_1 = s_j)$ and $\sum_{j=1}^{n} {}_j = 1$.

The 3 basic problems with HMMs

There are 3 basic problems to be solved for the model to be useful in real-world applications:

1. Problem 1: Evaluation . Compute of the probability $P(O|\ )$ of the observation sequence $\{O_1; \ldots; O_T\}$ given an HMM $= (\ A; B;\ )$.
2. Problem 2: Inference . Given a sequence $\{O_1; \ldots; O_T\}$ and the model , chose a state sequence $Q = q_1; \ldots; q_T$ which is meaningful (i.e. that best explains the observations) in some sense ? Several optimality criteria to be imposed.
3. Problem 3: Training . Given a sequence $\{O_1; \ldots; O_T\}$, how do we adjust the model $= (\ A; B;\ )$ to maximize $P(O|\ )$?

Direct evaluation of $P(O|\lambda)$

- The most straightforward way to compute of $P(O|\lambda)$ is through enumerating all every state sequence $q_1, \ldots, q_T$:

$$
\begin{aligned}
P(O|\lambda) &= \sum_{Q} P(O, Q|\lambda) = \sum_{Q} P(O|Q, \lambda)\, P(Q|\lambda) \\
&= \sum_{Q} P(q_1|\lambda) \prod_{t=1}^{T} P(O_t|q_t, \lambda) \prod_{t=2}^{T} P(q_t|q_{t-1}, \lambda) \\
&= \sum_{Q} \pi_{q_1} \prod_{t=2}^{T} b_{q_t}(O_t)\, a_{q_{t-1}, q_t}
\end{aligned}
$$

- The calculation of $P(O|\lambda)$ involves $O(2T \cdot n^T)$ calculations. A more efficient procedure is needed.

Forward Approach

- Let $\alpha_t(i) = P(O_1, \ldots, O_t, q_t = s_i | \lambda) = P(O_1^t, q_t = s_i | \lambda)$

$$\alpha_1(i) = P(O_1, q_1 = s_i | \lambda) = \pi_i b_i(O_1)$$

$$\begin{aligned}
\alpha_{t+1}(j) &= P(O_1, \ldots, O_t, O_{t+1}, q_{t+1} = s_j | \lambda) \\
&= \sum_{i=1}^{n} P(O_1^t, O_{t+1}, q_t = s_i, q_{t+1} = s_j | \lambda) \\
&= \sum_{i=1}^{n} P(O_{t+1} | q_{t+1} = s_i, \lambda) P(O_1^t, q_t = s_i | \lambda) a_{ij} \\
&= [\sum_{i=1}^{n} \alpha_t(i) a_{ij}] b_j(O_{t+1})
\end{aligned}$$

- Finally: $P(O | \lambda) = \sum_{i=1}^{n} \alpha_T(i)$

Backward Approach

- Likewise, let $_t(i) = P(O_{t+1}, \ldots, O_T \mid q_t = s_i,)$

$$_T(i) = 1$$

- As previously

$$_t(i) = \sum_{j=1}^{n} a_{ij}\, b_j(O_{t+1})\, _{t+1}(j)$$

- Finally : $P(O \mid) = \sum_{i=1}^{n} {}_i\, b_i(O_1)\, _1(i)$.
- In both cases, the complexity is $O(n^2 T)$

Which state q_t is the most likely?

There are several ways of nding the optimal state sequence .

- Which state q_t is the most likely?
- Let ${}_t(i) = P(q_t = s_i | O_1^T)$

$$
\begin{aligned}
{}_t(i) &= P(q_t = s_i | O_1^T) \\
&= P(q_t = s_i | O_1, \ldots, O_t, O_{t+1}, \ldots, O_T) \\
&= \frac{P(O_1, \ldots, O_t, q_t = s_i |\ ) \ P(O_{t+1}, \ldots, O_T | q_t = s_i, \ )}{P(O_1^T |\ )} \\
&= \frac{{}_t(i)\ {}_t(i)}{\sum_{j=1}^{n} {}_t(j)\ {}_t(j)}
\end{aligned}
$$

- Then we solve $q_t = \mathrm{argmax}_i [\ {}_t(i)]$
- One problem is that the state sequence may not even be valid, for instance if state transitions have zero probability.

Optimal state sequence: Viterbi algorithm

- The single best path sequence is given by $\max_Q P(O; Q j\ )$. De ne

$$_t(i) = \max_{q_1;:::;q_{T}} P(q_1; :::; q_{t\ \ 1}; q_t = s_i; O_1; :::; O_T j\ )$$

- By induction (dynamic programming), we have the recursion:

$$_{t+1}(j) = [\max_i\ _t(i) a_{ij}] b_j(O_{t+1})$$

- Hence the complete recursive procedure,

1. $_1(i) =\ _i b_i(O_1)$
2. $_{t+1}(j) = [\max_i\ _t(i) a_{ij}] b_j(O_{t+1})$
3. $\max_Q P(O; Q j\ ) = \max_i\ _T(i)$

Training

- The most dif cult problem of HMMs is to adjust the model parameters to maximize the likelihood.
- Suppose we have the sequence $O = \{O^1, \ldots, O^n\}$, the goal is to nd the parameters $\;= (\;\; A; B; \;)$ such that $P(O| \;) = \prod_{k=1}^{n} P(O^k| \;)$ is locally maximum using gradient or EM techniques.
- We compute $_{k+1}$ from $_k$ such that $P(O|\;_{k+1}) \quad P(O|\;_k)$.
- Eventually, the likelihood function converges to a critical point.
- We de ne next the Baum-Welch iterative procedure for choosing model parameters.

Baum-Welch Algorithm

$$\text{Let } {}^k_t(i;j) = P(q_t = s_i; q_{t+1} = s_j | O^k;)$$

$$= \frac{P(q_t = s_i; q_{t+1} = s_j; O^k |)}{P(O^k |)}$$

$$= \frac{{}^k_t(i) a_{ij} b_j(O^k_{t+1}) \; {}^k_{t+1}(j)}{P(O^k |)}$$

$$\text{We have } {}_t(i) = P(q_t = s_i | O_1^T)$$

$$= \sum_{j=1}^{n} P(q_t = s_i; q_{t+1} = s_j | O^T;)$$

$$= \sum_{j=1}^{n} {}^k_t(i;j) = \frac{{}_t(i) \; {}_t(i)}{\sum_{j=1}^{n} {}_t(j) \; {}_t(j)}$$

Baum-Welch Algorithm

- The parameters of new HMM model $_{p+1}$ are re-estimated from the previous one $_p$:

$$a_{ij} = \frac{\sum_{k=1}^{m} \sum_{t} {}_{t}^{k}(i,j)}{\sum_{k=1}^{m} \sum_{t} {}_{t}^{k}(i)}$$

$$b_j(l) = \frac{\sum_{k=1}^{m} \sum_{\{t : O_t^k = v_l\}} {}_{t}^{k}(j)}{\sum_{k=1}^{m} \sum_{t} {}_{t}^{k}(i)}$$

$$_i = \frac{1}{m} \sum_{k=1}^{m} {}_{1}^{k}(i)$$

Baum-Welch Algorithm

1. Given $_0 = (A;B;\ )$ et $p = 0$
2. Do: Compute $^k_t(i;j)$ with $^k_1(i); 81\ \ i;j\ \ n$ with $1\ \ t\ \ T\ \ 1$ and
   $_p$
3. Estimate $a_{ij}\ ; b_j(l);\ _i$
4. Let $_p = (A;B;\ )$
5. $p\ \ p{+}1$
6. Until convergence

Extensions

- HMMs are generative models: they model a joint distribution of observations and hidden states.
- A discriminative model can be used in place of the generative model of standard HMMs. This type of model directly models the conditional distribution of the hidden states given the observations X .
- HMM can also be generalized to allow continuous observations and/or state spaces (typically Gaussian), however, in general, exact inference in HMMs with continuous latent variables is infeasible.
- A uniform prior distribution over the transition probabilities was implicitly assumed. Another prior candidate is the Dirichlet distribution.

Outline

Sum product networks

- Sum-product networks (SPNs) are PGMs of a joint distribution $p(v)$, though not in the classical sense.
- In classical PGMs, the graphical structure G is de ned over a node set corresponding to the random variables in V.
- In a SPN, $p(v)$ is constrained by a directed tree structure with three types of nodes: sum, product and leaf nodes.
- $p(v)$ encoded by the SPN corresponds to the local distribution of the root node, and decomposes recursively into products and weighted sums of local distributions according to the SPN structure.
- SPNs can also be represented as DAGs.

Sum product networks

- A product node N_i performs the product of the local distributions of its children CH_{N_i},

$$p_i(v_i) = \prod_{N_j \in CH_{N_i}} p_j(v_j),$$

- The scopes of the child nodes form a partition of the scope of the product node V_i.
- A sum node N_i is a weighted sum of the local distributions of its children,

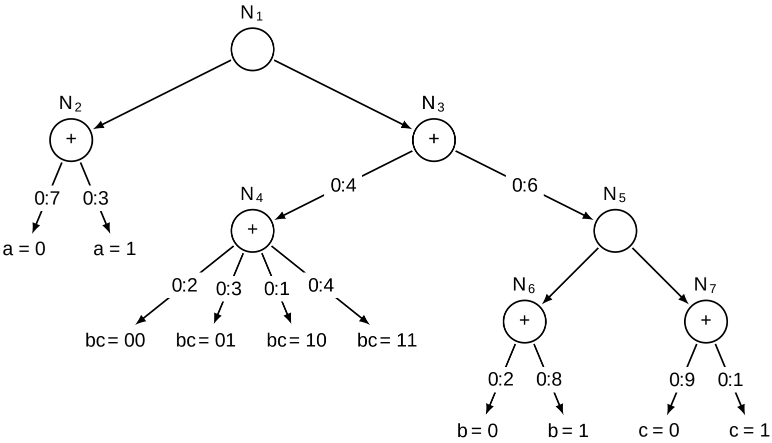
$$p_i(v_i) = \sum_{N_j \in CH_{N_i}} \omega_{i;j} \, p_j(v_j),$$

- $\sum_j \omega_{i;j} = 1$ and scope of the sum child nodes is that of their parent, $V_j = V_i$.

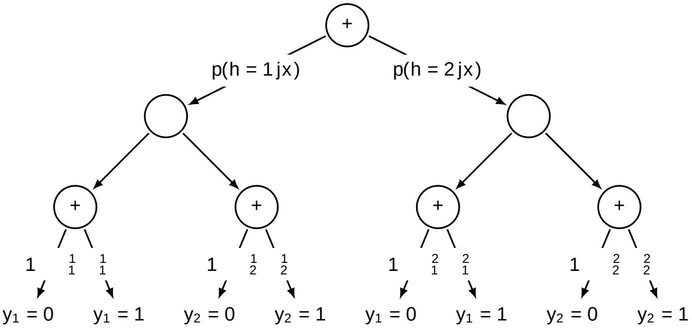
SPN nodes

- Each node N_i represents a local joint distribution $p_i(v_i)$ over a subset of variables V_i V called its scope, not be confused with the marginal distribution $p(v_i)$.
- A leaf node N_i is associated with a probability distribution $p_i(v_i)$, e.g. a probability table or a parametric probability density function.
- In a leaf node, $p_i(v_i)$ equals to 1 for a particular instanciation of v_i, and 0 elsewhere.
- A SPN encodes the full joint distribution $p(v)$ solely with the parameters corresponding to the weights of the sum nodes.

SPN over three binary random variables



SPN for $p(y_1; y_2 j x)$



- The SPN model for $p(y_1; y_2 j x)$ is equivalent to a mixture of conditional Bernoulli distributions with two components ($k = 2$). ^j_i denotes $p(y_i = 1 j x; h = j)$.

Inference

- Computing joint, marginal and conditional probabilities has complexity linear to the size of the the number of nodes of SPN.
- For example, consider the query $p(a=1, b=1 | c=0)$. To compute $p(a=1, b=1, c=0)$, it suf ces to propagate the evidence values from the leaf nodes with a bottom-up pass up to the root node, we obtain

$$p(a=1, b=1, c=0) = 0.3 \quad (0.4 \quad 0.1 + \dots = 0.14$$
$$p(c=0) = (0.7 + 0.3) \quad (0.4 \quad (0.2 + 0.1) + \dots = 0.66$$

- Finally, in two passes through the SPN, we get $p(a=1, b=1 | c=0) = 0.14/0.66 = 0.215$.
- MAP inference in SPN remains a hard problem in general.

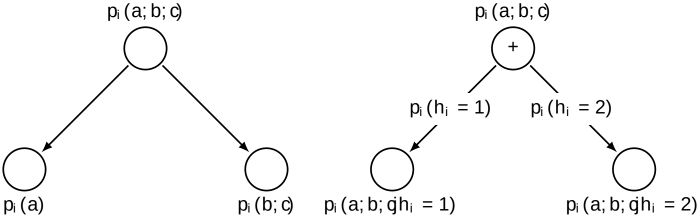
Inference

- If $p(y|x)$ is represented as a mixture of k conditional Bernoulli distributions.

$$p(y|x) = \sum_{j=1}^{k} p(h = j|x) \prod_{i=1}^{n} p(y_i|x; h = j).$$

- where a hidden variable H takes values in $\{1, \dots, k\}$.
- The model can be expressed as a three-layer SPN in which the weights of the sum nodes are not fixed but inferred from a set of probabilistic models.
- $p(h = j|x)$ can be obtained with a multinomial probabilistic regression and $p(y_i|x; h = j)$ with a binary probabilistic regression.
- Learning with expectation-maximization (EM) and MAP inference can be performed at a reasonable cost.

Latent variable interpretation



- The product node decomposes $p_i(a; b; c)$ into a product over disjoint factors, $p_i(a; b; c) = p_i(a) p_i(b; c)$;
- The sum node is a mixture model with a hidden discrete variable, $p_i(a; b; c) = \sum_{h_i} p_i(a; b; c; h_i)$.

Outline

References I

D. Koller and N. Friedman.
Probabilistic Graphical Models: Principles and Techniques.
MIT Press, 2009.

Christopher M. Bishop.
Pattern Recognition and Machine Learning.
Springer, 2006.

G. Bontempi and S. Ben Taieb.
Statistical foundations of machine learning
OTexts: Melbourne, Australia, 2016.

Maxime Gasse.
Probabilistic Graphical Model Structure Learning: Application to Multi-Label Classification.
Ph.D. thesis, Université de Lyon, 2017.

References II

Lawrence R. Rabiner.
A tutorial on Hidden Markov Models and selected applications in speech recognition, Proceedings of the IEEE, vol. 77, no 2, 1989.

G. Obozinski and F. Bach
Introduction to Graphical Models, Master ”Mathématiques Appliquées”, Parcours ”Mathematiques, Vision et Apprentissage”, ENS Cachan 2017.