Graphical Models

Christophe Ambroise

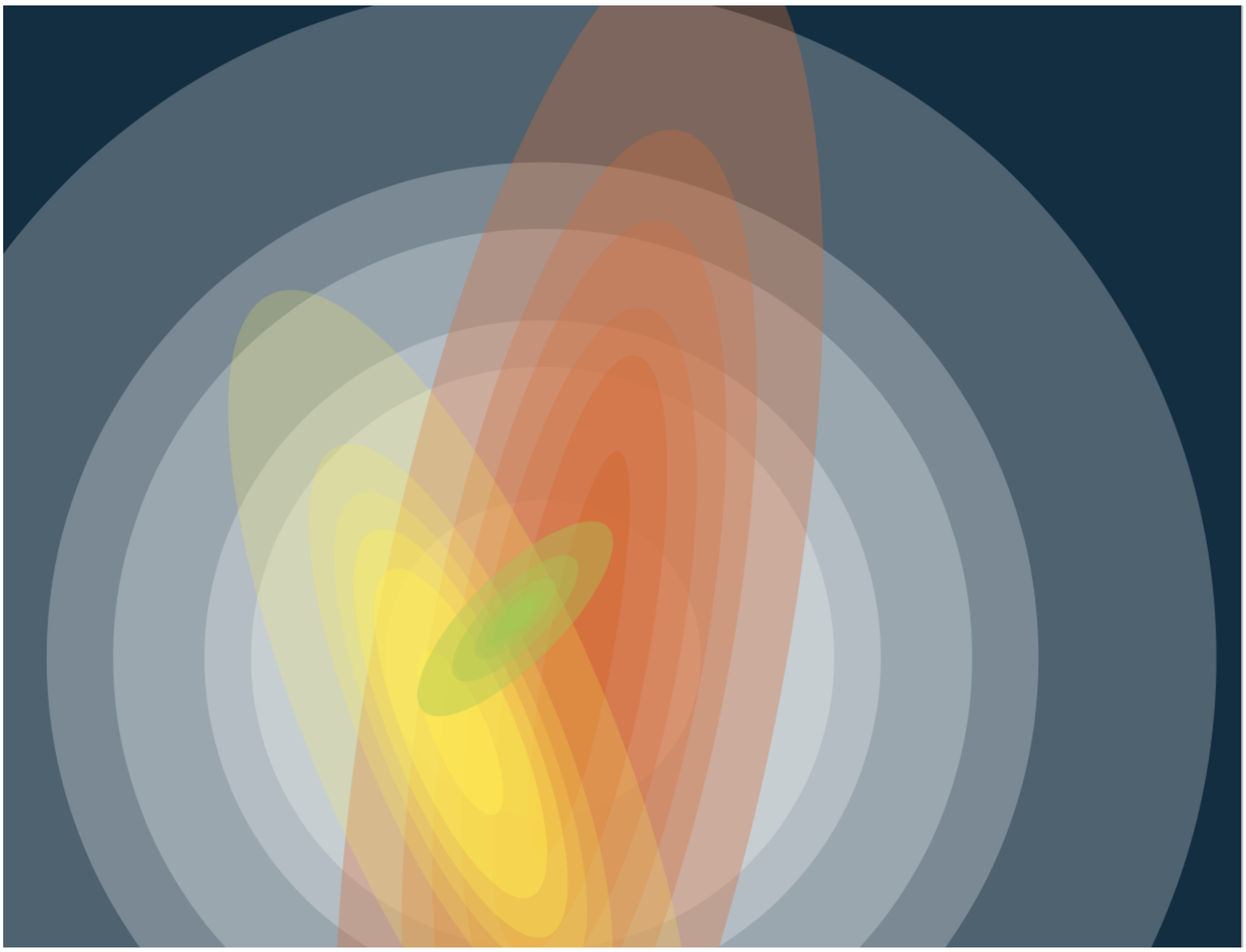
christophe.ambroise@univ-evry.fr

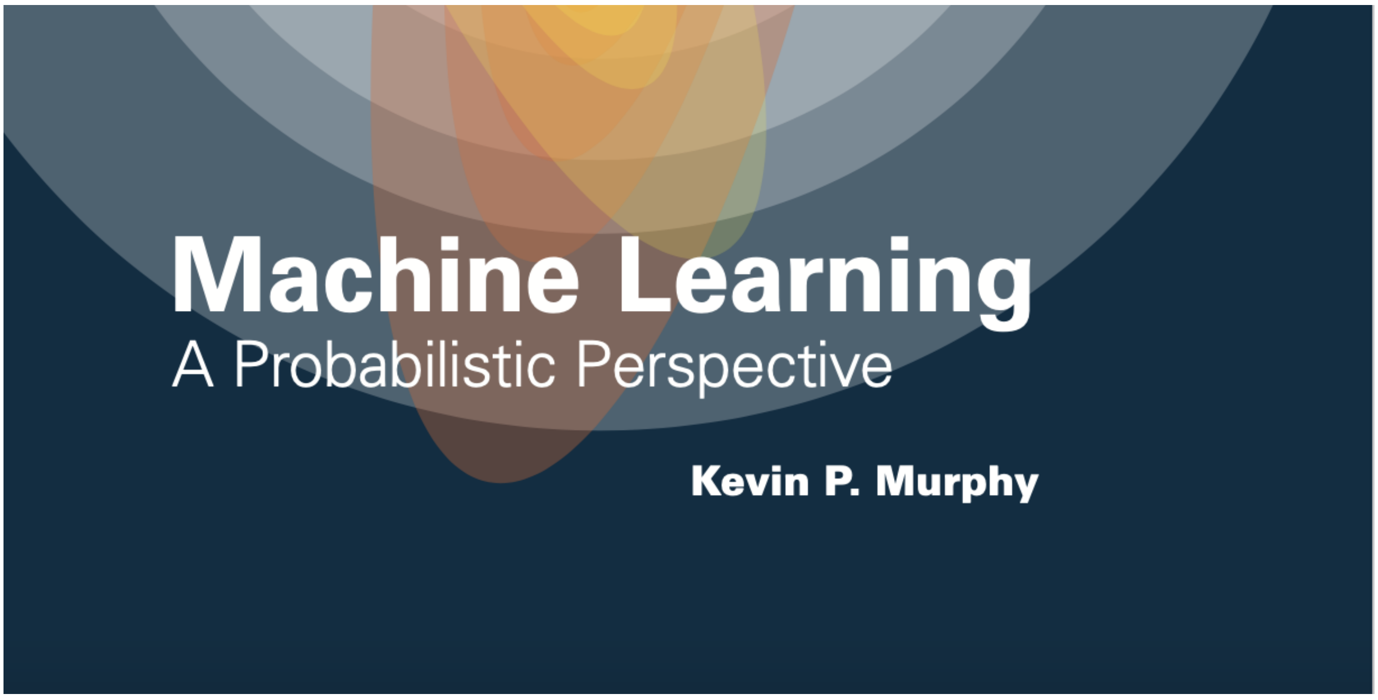
UEVE, UMR CNRS 8071

November 7, 2023

Introduction

Practical matters





Reference document

The lecture closely follows and largely borrows material from "Machine Learning: A Probabilistic Perspective" (MLAPP) from Kevin P. Murphy, chapters:

Practical matters

Evaluation

The project will be evaluated through of a project in R or Python (realized by 2 or 3 student). Each project will be different and rated on the basis of

- a code (1/3)
- a presentation (15 minutes) (1/3)
- a report (1/3)

What is a graphical model ?

A graphical model is a probability distribution in a factorized form

There a two main type of representation of the factorization:

- directed graphical model
- undirected graphical model

Why the term graph ?

Conditionnal independences between variables are well modeled via Graphs

What is it usefull for ?

- reduce the number of parameters
  - → may be used for supervised or unsupervised approaches
- allow exploratory data analysis by providing a simple graphical representation
  - → “approach causality”

What problems does it raise ?

- learning the parameters of a given factorized form
- learning the structure of the graphical model (factorized form)

Directed Graphical Models (Chapter 10 MLAPP)

Joint distribution

Observation

Suppose we observe multiple correlated variables, such as words in a document, pixels in an image, or genes in a microarray.

Joint distribution

How can we compactly represent the joint distribution $p(x|\theta)$?

Chain Rule

By the chain rule of probability, we can always represent a joint distribution as follows, using any ordering of the variables:

$$p(x_{1:V}) = p(x_1)p(x_2|x_1)p(x_3|x_2,x_1)p(x_4|x_1,x_2,x_3)\ldots p(x$$

The problem of the number of parameters

$O(K) + O(K^2) + O(K^3) + \ldots$ There are $O(K^V)$ parameters in the system

Conditional independence

The key to efficiently representing large joint distributions is to make some assumptions about conditional independence (CI).

$$X \perp Y|Z \Leftrightarrow p(X, Y|Z) = p(X|Z)p(Y|Z)$$

X is conditionaly independent of Y knowing Z if once you know Z knowing Y does not help you to guess X

Conditional independence: an example

Setting: picking a card at random in a traditional set of cards

1. if full set of color and values then $color \perp value$
2. if all diamond faces (♦) are discarded from the set then $color \not\perp value$ but still $color \perp value|Facecard$

$$P(King|Facecard) = 1/3 = P(\clubsuit|Facecard)$$

$$P(King\clubsuit|Facecard) = 1/9 = P(King|Facecard)P(\clubsuit|F$$

Simplification of chain rule

Simplficiation of chain rule factorization

Let assume that $x_{t+1} \perp x_{1:t-1} | x_t$, first order Markov assumption.

$$p(x_{1:V}) = p(x_1) \prod_{t=2}^{V} p(x_t | x_{t-1})$$

$K - 1 + K^2$ parameters

Graphical models

A graphical model (GM) is a way to represent a joint distribution by making Conditional Independence (CI) assumptions.

- the nodes in the graph represent random variables,
- and the (lack of) edges represent CI assumptions.

A better name for these models would in fact be "independence diagrams"

There are several kinds of graphical model, depending on whether

- the graph is directed,
- undirected,
- or some combination of directed and undirected.

Example of directed and undirected graphical model

Graph terminology

A graph $G = (V, E)$ consists of

- a set of nodes or vertices, $V = \{1, \ldots, V\}$, and
- a set of edges, $E = \{(s, t) : s, t \in V\}$.

Adjacency matrix

We can represent the graph by its adjacency matrix, in which we write $G(s, t) = 1$ to denote $(s, t) \in E$, that is, if $s \to t$ is an edge in the graph. If $G(s, t) = 1$ iff $G(t, s) = 1$, we say the graph is undirected, otherwise it is directed.

We usually assume $G(s, s) = 0$, which means there are no self loops.

Graph terminology

- **Parent**: For a directed graph, the parents of a node is the set of all nodes that feed into it: $pa(s) \triangleq \{t : G(t, s) = 1\}$.
- **Child**: For a directed graph, the children of a node is the set of all nodes that feed out of it: $ch(s) \triangleq \{t : G(s, t) = 1\}$.
- **Family**: For a directed graph, the family of a node is the node and its parents, $fam(s) = s \cup pa(s)$.
- **Root**: For a directed graph, a root is a node with no parents.
- **Leaf**: For a directed graph, a leaf is a node with no children.
- **Ancestors**: For a directed graph, the ancestors are the parents, grand-parents, etc of a node. That is, the ancestors of t is the set of nodes that connect to t via a trail:
  $anc(t) \triangleq \{s : s \rightsquigarrow t\}$.
- **Descendants**: For a directed graph, the descendants are the children, grand-children, etc of a node. That is, the descendants

Graph terminology

- **Clique**: For an undirected graph, a clique is a set of nodes that are all neighbors of each other.
- **A maximal clique** is a clique which cannot be made any larger without losing the clique property.
- **Neighbors** For any graph, we define the neighbors of a node as the set of all immediately connected nodes, $nbr(s) \triangleq \{t : G(s,t) = 1 v G(t,s) = 1\}$. For an undirected graph, we write $s \sim t$ to indicate that s and t are neighbors.
- **Degree**: The degree of a node is the number of neighbors. For directed graphs, we speak of the in-degree and out-degree, which count the number of parents and children.
- **Cycle or loop**: For any graph, we define a cycle or loop to be a series of nodes such that we can get back to where we started by following edges
- **DAG** A directed acyclic graph or DAG is a directed graph with no

Directed graphical models

- A directed graphical model or DGM is a GM whose graph is a DAG.
- These are more commonly known as **Bayesian networks**
- These models are also called **belief networks**
- Finally, these models are sometimes called **causal networks**, because the directed arrows are sometimes interpreted as representing causal relations.

Topological ordering of DAGs

- nodes can be ordered such that parents come before children
- it can be constructed from any DAG

The ordered Markov property

a node only depends on its immediate parents

$$x_s \perp x_{pred(s)\setminus pa(s)} | x_{pa(s)}$$

where pa(s) are the parents of node s, and pred(s) are the predecessors of node s in the ordering.

General form of factorization

$$p(x_{1:V}) = \prod_{t=1}^{V} p(x_t | x_{pa(t)})$$

if the Conditional Independence assumptions encoded in DAG G are correct

Examples

Naive Bayes classifiers

$$p(y, x) = p(y) \prod_j p(x_j | y)$$

The naive Bayes assumption is rather naive, since it assumes the features are conditionally independent.

Markov and hidden Markov models

Markov chain

$$p(x_{1:T}) = p(x_1)p(x_2|x_1)p(x_3|x_2)\ldots = p(x_1)\prod_{t=2}^{T} p(x_t|x_{t-1})$$

Hidden Markov Model

The hidden variables often represent quantities of interest, such as the identity of the word that someone is currently speaking. The observed variables are what we measure, such as the acoustic waveform.

Directed Gaussian graphical models

Consider a DGM where all the variables are real-valued, and all the Conditional Proba. Distributions have the following form:

$$p(x_t|x_{pa(t)}) = \mathcal{N}(x_t|\mu_t + \boldsymbol{w}_t^T x_{pa(t)}, \sigma_t^2)$$

Directed GGM (Gaussian Bayes net)

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Directed GGM (Gaussian Bayes net)

For convenience let rewrite the CPDs

$$x_t = \mu_t + \sum_{s \in pa(t)} w_{ts}(x_s - \mu_s) + \sigma_t z_t$$

where $z_t \sim \mathcal{N}(0, 1)$, σ_t is the conditional standard deviation of x_t given its parents, wts is the strength of the $s \rightarrow t$ edge, and μ_t is the local mean.

Mean

The global mean is just the concatenation of the local means

$$\mu = (\mu_1, \ldots, \mu_D)^t.$$

Directed GGM (Gaussian Bayes net)

Covariance matrix

$$(\boldsymbol{x} - \boldsymbol{\mu}) = W(\boldsymbol{x} - \boldsymbol{\mu}) + S\boldsymbol{z}$$

where $S \triangleq diag(S)$ Let consider $\boldsymbol{e} \triangleq S\boldsymbol{z} = (I - W)(\boldsymbol{x} - \boldsymbol{\mu})$

We have

$$\Sigma = cov(\boldsymbol{x} - \boldsymbol{\mu}) = cov((I - W)^{-1}\boldsymbol{e}) = cov(USz) = US^2U$$

where $U = (I - W)^{-1}$

Examples

Two extreme cases

- Isolated vertices : Naive Bayes where $\mathbf{\Sigma} = S$, p vertices, no edges
- Fully connected Graph: p vertices, $p(p-1)/2$ directed edges

Click to goto exercice on Directed GGM

Learning {#learning}

Learning from complete data (with known graph structure)

If all the variables are fully observed in each case, so there is no missing data and there are no hidden variables, we say the data is complete.

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \prod_{i=1}^{N} \prod_{t \in V} p(\boldsymbol{x}_{it}|\boldsymbol{x}_{i,pa(t)}, \boldsymbol{\theta}_t)$$

The likelihhod decomposes according the **graph structure**

Click to goto exercice on Sprinkler

Discrete distribution

$$N_{tck} \triangleq \sum_{i=1}^{N} \mathbb{I}(\boldsymbol{x}_{i,t} = \boldsymbol{k}, \boldsymbol{x}_{i,pa(t)} = \boldsymbol{c})$$

and thus $\hat{\boldsymbol{p}}(\boldsymbol{x}_t = \boldsymbol{k}, \boldsymbol{x}_{pa(t)} = \boldsymbol{c}) = \frac{N_{tck}}{\sum_{k'} N_{tck'}}$ Of course, the MLE suffers from the zero-count

Conditional independence properties of DGMs

Diverging edges (fork)

With the DAG

$$A \leftarrow C \rightarrow B$$

with have

$$A \not\perp B$$

but

$$A \perp B | C$$

Exercice

Chain (Head - tail)

With the DAG

$$A \to C \to B$$

with have

$$A \not\perp B$$

but

$$A \perp B | C$$

Exercice

Converging edges (V) and collider

With the DAG

$$A \to C \leftarrow B$$

with have

$$A \perp B$$

but

$$A \not\perp B | C$$

Exercice

Show it

Independence map

a directed graph G is an I-map (independence map) for p, or that p is Markov wrt G,

- iff $I(G) \subseteq I(p)$, where $I(p)$ is the set of all CI statements that hold for distribution p.

This allows us to use the graph as a safe proxy for p

Minimal I-map

d-separation

- The “d” in d-separation and d-connection stands for dependence.
- d-separation is related the ideas of active path and active vertex on a path
- **a path is active if it carries information, or dependence.**
- Thus, when the conditioning set is empty, only paths that correspond to “causal connection” are active (creating dependance).

d-separation: example of Pearl (1988)

two independent causes of your car refusing to start: having no gas and having a dead battery.

dead battery –> car won't start <– no gas

- Telling you that the battery is charged tells you nothing about whether there is gas,
- Telling you that the battery is charged after I have told you that the car won't start tells me that the gas tank must be empty.

So independent causes are made dependent by conditioning on a common effect, which in the directed graph representing the causal structure is the same as conditioning on a collider.

d-separation

When a vertex is in the conditioning set, its status with respect to being active or inactive **flip-flops**. **If we condition by C**

Are variables A and B are d-separated by C (in boldface).

1. A –> **C** –> B **Inactive**
2. A <– **C** <– B **Inactive**
3. A <– **C** –> B **Iactive**
4. A –> **C** <– B, C is a collider and thus inactive when the conditioning set is empty, so condiitionning by C it becomes **Active** (produce dependence)

Formulation d-separation definition

an undirected path P is d-separated by a set of nodes E iff at least one of the following conditions hold:

- P contains a chain, $s \rightarrow m \rightarrow t$ or $s \leftarrow m \leftarrow t$ where $m \in E$
- P contains a fork, $s \leftarrow m \rightarrow t$ where $m \in E$
- P contains a collider, $s \rightarrow m \leftarrow t$ where $m \notin E$ and nor is any descendant of m.

Alternative formulation of d-connection:

If G is a directed graph in which X, Y and E are disjoint sets of vertices, then X and Y are d-connected by E in G if and only if there exists an undirected path P between some vertex in X and some vertex in Y such that

- for every collider C on P, either C or a descendent of C is in E (active path),
- and no non-collider on P is in E (no inactive path).

X and Y are d-separated by E in G if and only if they are not d-connected by E in G (all path are inactives…).

Independance requires **all possible paths** to be inactive whereas dependence requires only on leak (one active path)

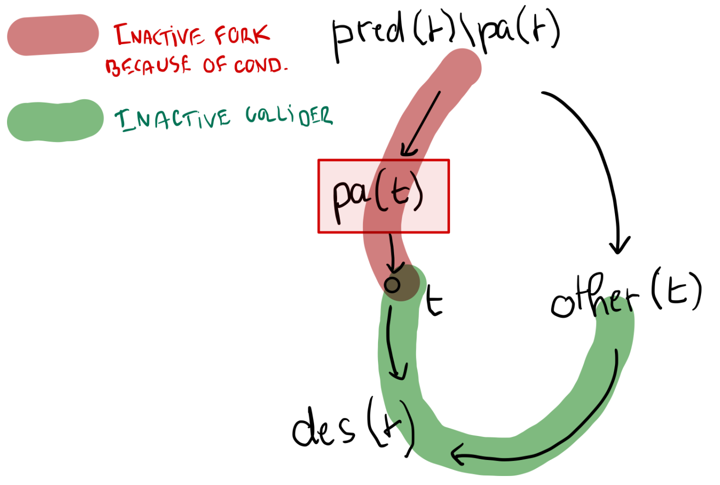
see https://www.youtube.com/watch?v=yDs_q6jKHb0 for examples

d-separation versus conditional independence

a set of nodes A is d-separated from a different set of nodes B given a third observed set E iff each undirected path from every node $\boldsymbol{a} \in \boldsymbol{A}$ to every node $\boldsymbol{b} \in \boldsymbol{B}$ is d-separated by E:

$\boldsymbol{x}_A \perp_G \boldsymbol{x}_B | \boldsymbol{x}_E \Leftrightarrow$ A is d-separated from B given E

Consequences of d-separation



Directed local Markov property

From the d-separation criterion, one can conclude that $t \perp nd(t)\backslash pa(t)|pa(t)$ where the non-descendants of a node $nd(t)$ are all the nodes except for its descendants

Consequences of d-separation

Ordered Markov property

A special case of directed local Markov property is when we only look at predecessors of a node according to some topological ordering. We have $t \perp pred(t)\backslash pa(t)|pa(t)$

Markov blanket

The set of nodes that renders a node t conditionally independent of all the other nodes in the graph is called t's Markov blanket

$$mb(t) \triangleq pa(t) \cup ch(t) \cup copa(t)$$

The Markov blanket of a node in a DGM is equal to the parents, the children, and the co-parents.

Markov blanket

To understand the Markov blanket, one could start with the local Markov property which block the dependence to non-descendant by conditioning on the parents.

To further block the path the descendants of t one has to

- Condition on the children of t.
- But **conditioning on the children** open the path to the coparents.
- Thus one needs conditioning **on the coparents** to block all paths.

Graphical Model Learning Structure (chapter 26 MLAPP)

Introduction

Two main applications of structure learning:

1. knowledge discovery (requires a graph topology)
2. density estimation (requires a fully specified model).

main obstacle

the number of possible graphs is exponential in the number of nodes: a simple upper bound is $O(2^{V(V-1)/2})$.

Relevance network

A relevance network is a way of visualizing the pairwise mutual information between multiple random variables:

- we simply choose a threshold α
- draw an edge from node i to node j if $\mathbb{I}(X_i; X_j) > \alpha$

Major problem

- the graphs are usually very dense,
- most variables are dependent on most other variables, even after thresholding the MIs.

Gaussian case

In the Gaussian case, $\mathbb{I}(X_i; X_j) = -1/2 \log\left(1 - \rho_{ij}^2\right)$ where ρ_{ij} is the correlation coefficient so we are essentially visualizing Σ;

this is known as the covariance graph.

Exercice : Gaussian mutual information

Show the previous statement

Dependency networks

Learning tree structures

Since the problem of structure learning for general graphs is NP-hard (Chickering 1996), we start by considering the special case of trees. Trees are special because we can learn their structure efficiently

Joint Distribution associated to a directed tree

A directed tree, with a single root node r, defines a joint distribution as follows

$$p(x|T) = \prod_{t \in V} p(x_t | x_{pa(t)})$$

The distribution is a product over the edges and the choice of root does not matter

Symetrization

To make the model more symmetric, it is preferable to use an undirected tree:

$$p(x|T) = \prod_{t \in V} p(x_t) \prod_{(s,t) \in E} \frac{p(x_s, x_t)}{p(x_s)p(x_t)}$$

Chow-Liu algorithm for finding the ML tree structure (1968)

Goal: Chow Liu algorithm constructs tree distribution approximation that has the minimum Kullback–Leibler divergence to the actual distribution (that maximizes the data likelihood)

Principle

1. Compute weight $I(s,t)$ of each (possible) edge (s,t)
2. Find a maximum weight spanning tree (MST)
3. Give directions to edges in MST by chosing a root node

Chow-Liu algorithm for finding the ML tree structure (1968)

log-likelihood

$$\log P(\boldsymbol{\theta}|\mathcal{D},T) = \sum_{tk} N_{tk} \log p(x_t = k) + \sum_{st}\sum_{jk} N_{stjk} \log$$

thus $\hat{p}(x_t = k) = \frac{N_{tk}}{N}$ and $\hat{p}(x_s = j, x_t = k) = \frac{N_{stjk}}{N}$.

Mutual information of a pair of variables

$$I(s,t) = \sum_{jk} \hat{p}(x_s = j, x_t = k) \log \frac{\hat{p}(x_s = j, x_t = k)}{\hat{p}(x_s = j)\hat{p}(x_t = k)}$$

The Kullback–Leibler divergence

$$\frac{\log P(\hat{\boldsymbol{\theta}}_{\boldsymbol{ML}}|\mathcal{D},T)}{N} = \sum_{tk} \hat{p}(x_t = k) \log \hat{p}(x_t = k) + \sum_{st} I(s$$

Chow-Liu algorithm

There are several algorithms for finding a max spanning tree (MST). The two best known are - Prim's algorithm and - Kruskal's algorithm.

Both can be implemented to run in $O(ElogV)$ time, where $E = V^2$ is the number of edges and V is the number of nodes.

Exercice Gaussian Chow-Liu

1. Show that in the Gaussian case, $I(s,t) = -\frac{1}{2}\log(1-\rho_{st}^2)$,where ρ_{st} is the correlation coefficient (see Exercise 2.13, Murphy)
2. Given a realisation of n gaussian vector of size p find the ML tree structured covariance matrix using Chow-Liu algorithm.

TAN: Tree-Augmented Naive Bayes

- Naive Bayse with Chow-Liu

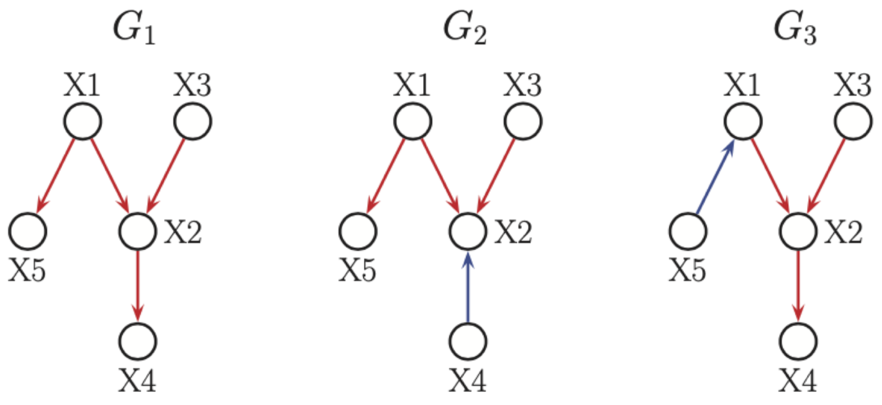
Mixtures of trees

- A single tree is rather limited in its expressive power.
- learning a mixture of trees (Meila and Jordan 2000), where each mixture component may have a different tree topology is an alternative

Tntegrating out over all possible trees.

This can be done in V^3 time using the matrix tree theorem.

Learning DAG structures



Three DAGs. G1 and G3 are Markov equivalent,G2 is not.

Graphs are Markov equivalent

if they encode the same set of CI assumptions

Learning DAG structures

An ill posed problem

when we learn the DAG structure from data, we will not be able to uniquely identify all of the edge directions

we can learn DAG structure “up to Markov equivalence”.

Do not read too much into the meaning of particular edge orientations, since we can often change them without changing the model in any observable way.

Exact structural inference

Exact structural inference is based on the computation of exact posterior over graphs, $p(G|D)$.

It requires:

- the computation of the likelihood $p(D|G)$
- the computation of the prior $p(G)$

This solution allows to compared different graph in terms of posterior and eventually find the MAP if the search space is small

Exact structural inference (categorical case)

Consider $x_{it} \in \{1, \cdots, K_t\}$ be the value of node t in case i, where

- K_t is the number of states for node t.
- $\theta_{tck} \triangleq p(x_t = k | x_{pa(t)} = c)$, for $k = 1 : K_t$, and $c = 1 : C_t$, where C_t is the number of parent combinations (possible conditioning cases).

Let $d_t = dim(pa(t))$ be the degree or fan-in of node t, so that $C_t = K^{d_t}$.

Exact structural inference (categorical case)

Prior

$$p(\theta) = \prod_{t=1}^{V} p(\theta_t) = \prod_{t=1}^{V} \prod_{c=1}^{C_t} p(\theta_{tc})$$

where C_t is the number of parent combinations (possible conditioning cases)

Likelihood

$$p(D|G, \theta) = \prod_{t=1}^{V} \prod_{c=1}^{C_t} \prod_{k=1}^{K_t} \theta_{tck}^{N_{tck}}$$

where N_{tck} is the number of time node t is in state k and its parent in state c.

Exact structural inference (categorical case)

Chosing a Dirichlet prior $p(\theta_{tc}) = Dir(\theta_{tc}|\alpha_{tc})$ allows to compute the posterior

$$p(D|G) = \prod_{t=1}^{V} \prod_{c=1}^{C_t} \frac{B(N_{tc} + \alpha_{tc})}{B(\alpha_{tc})}$$

where $N_{tc} = \sum_k N_{tck}$, and $\alpha_{tc} = \sum_k \alpha_{tck}$.

Local scoring

For node t and its parents

$$score(N_{t,pa(t)}) \triangleq \prod_{c=1}^{C_t} \frac{B(N_{tc} + \alpha_{tc})}{B(\alpha_{tc})}$$

Marginal likelihood factorizes according to the graph structure.

Setting the prior

How should we set the hyper-parameters α_{tck} ?

- Jeffreys prior of the form $\alpha_{tck} = 1/2$ violates a property called likelihood equivalence
- This property says that if G1 and G2 are Markov equivalent , they should have the same marginal likelihood

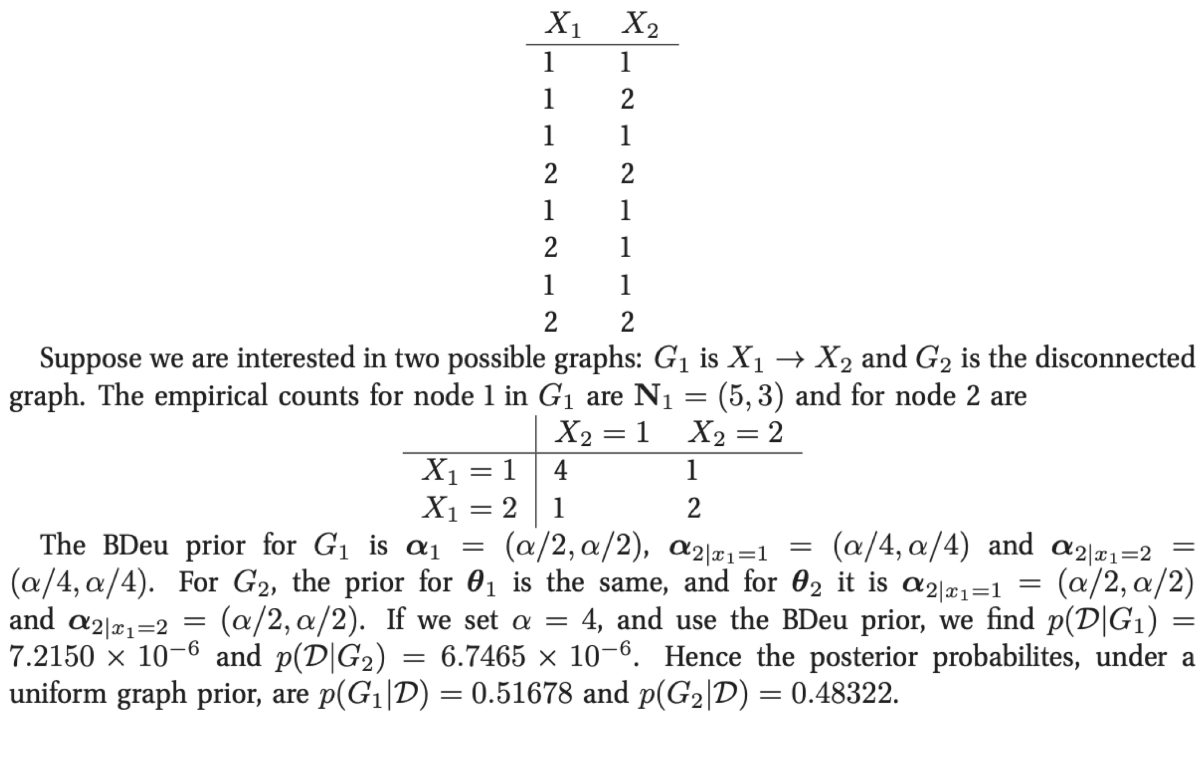
BDe prior

- Geiger and Heckerman (1997) proved that, for complete graphs, the only prior that satisfies likelihood equivalence and parameter independence is the Dirichlet prior, where the pseudo counts have the form

$$\alpha_{tck} = \alpha p_0(x_t = k, x_{pa(t)} = c)$$

where $\alpha > 0$ is called the equivalent sample size, and p_0 is some prior joint probability distribution. This is called the BDe prior (Bayesian Dirichlet likelihood equivalent).

Example of Exact structural inference (Neapolitan 2003, p.438)



X_1	X_2
1	1
1	2
1	1
2	2
1	1
2	1
1	1
2	2

Suppose we are interested in two possible graphs: G_1 is $X_1 \to X_2$ and G_2 is the disconnected graph. The empirical counts for node 1 in G_1 are $\mathbf{N}_1 = (5, 3)$ and for node 2 are

	$X_2 = 1$	$X_2 = 2$
$X_1 = 1$	4	1
$X_1 = 2$	1	2

The BDeu prior for G_1 is $\boldsymbol{\alpha}_1 = (\alpha/2, \alpha/2)$, $\boldsymbol{\alpha}_{2|x_1=1} = (\alpha/4, \alpha/4)$ and $\boldsymbol{\alpha}_{2|x_1=2} = (\alpha/4, \alpha/4)$. For G_2, the prior for $\boldsymbol{\theta}_1$ is the same, and for $\boldsymbol{\theta}_2$ it is $\boldsymbol{\alpha}_{2|x_1=1} = (\alpha/2, \alpha/2)$ and $\boldsymbol{\alpha}_{2|x_1=2} = (\alpha/2, \alpha/2)$. If we set $\alpha = 4$, and use the BDeu prior, we find $p(\mathcal{D}|G_1) = 7.2150 \times 10^{-6}$ and $p(\mathcal{D}|G_2) = 6.7465 \times 10^{-6}$. Hence the posterior probabilites, under a uniform graph prior, are $p(G_1|\mathcal{D}) = 0.51678$ and $p(G_2|\mathcal{D}) = 0.48322$.

Scaling up to larger graphs

The main challenge in computing the posterior over DAGs is that there are so many possible graphs.

Consequently, we must settle for finding a locally optimal MAP DAG.

Popular solution: Greedy hill climbing

Learning causal DAGs

Causal models

- predict the effects of interventions to, or manipulations of, a system.
- Causal claims are inherently stronger, yet more useful, than purely associative claims

Causal interpretation of DAGs

- $A \to B$ in a DAG to mean that "A directly causes B" so if we manipulate A, then B will change.
- Known as the causal Markov assumption.

Intervention

Perfect intervention

- represents the act of setting a variable to some known value
- A real world example of such a perfect intervention is a gene knockout experiment

do calculus notation

$do(X_i = x_i)$ to denote the event that we set X_i to x_i

- A causal model makes inferences of the form $p(x|do(X_i = x_i))$,
- Different from making inferences of the form $p(x|X_i = x_i)$.

Observing versus doing

Consider a 2 node DGM $S \to Y$

- $S = 1$ if you smoke
- $S = 0$ otherwise,
- $Y = 1$ if you have yellow-stained fingers
- $Y = 0$ otherwise.

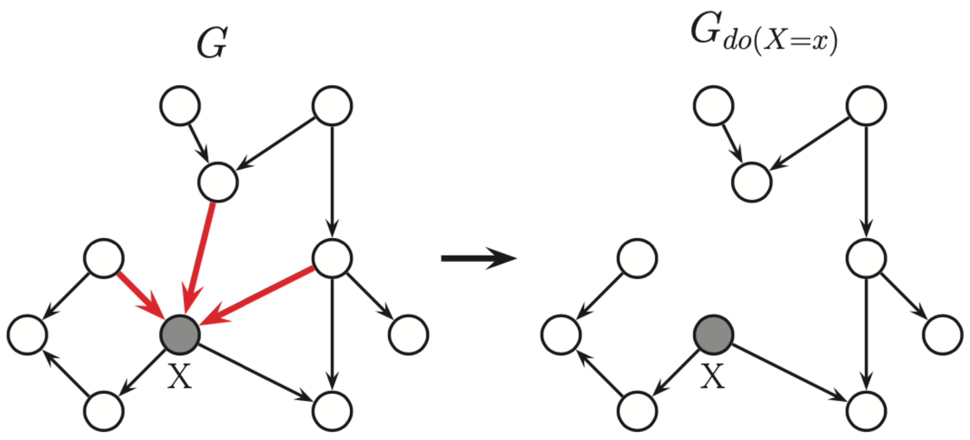
If I observe you have yellow fingers, I am licensed to infer that you are probably a smoker (since nicotine causes yellow stains):

$$p(S = 1|Y = 1) > p(S = 1)$$

If I intervene and paint your fingers yellow, I am no longer licensed to infer this, since I have disrupted the normal causal mechanism. Thus

$$p(S = 1 | do(Y = 1)) = p(S = 1)$$

Graph surgery



One way to model perfect interventions is to use graph surgery: - represent the joint distribution by a DGM, - cut the arcs coming into any nodes that were set by intervention.

Exercices on Directed Graphical Models

Exercice Gaussian Bayesian Network

Data

Let consider the following graph $x_1 \to x_2 \to x_3$ where - $\mathbb{E}[x_1] = b_1$, $\mathbb{E}[x_2] = b_2$, $\mathbb{E}[x_3] = b_3$ - $x_1 = b_1 + z_1$ - $x_2 = b_2 + (x_1 - b_1) + z_2$ - $x_3 = b_3 + 1/2(x_2 - b_2) + z_3$ - $\sigma_1 = \sigma_2 = \sigma_3 = 1$,

Problem

Exercice Directed GGM

- $\boldsymbol{\mu}^T = (0,1,2)$
- $diag(S) = (1,1,1)$
- $W = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1/2 & 0 \end{pmatrix}$

Exercice Directed GGM

We can observe that the precision matrix has the some support as W

```r
n=1000
mu=c(0,1,2)
sigma=c(1,1,1)
W=matrix(c(0,1,0,0,0,1/2,0,0,0),3,3)
U=solve(diag(rep(1,3))-W)
S=diag(sigma)
Sigma=U%*%S^2%*%t(U)
solve(Sigma)
```

```
     [,1]  [,2] [,3]
[1,]    2 -1.00  0.0
[2,]   -1  1.25 -0.5
[3,]    0 -0.50  1.0
```

Exercice Directed GGM

First solution (direct)

```r
library(mvtnorm)
Xprime=rmvnorm(n,mean=c(0,1,2),sigma=Sigma)
```

Second solution (constructive)

```r
X=matrix(0,n,3)
Z=matrix(rnorm(n*3),n,3)
for (i in 1:n)
  for (j in 1:3)
    X[i,j]=mu[j]+sigma[j]*Z[i,j] + sum(W[j,]*(X[i,]-mu))
```

Click to go Back to Lecture

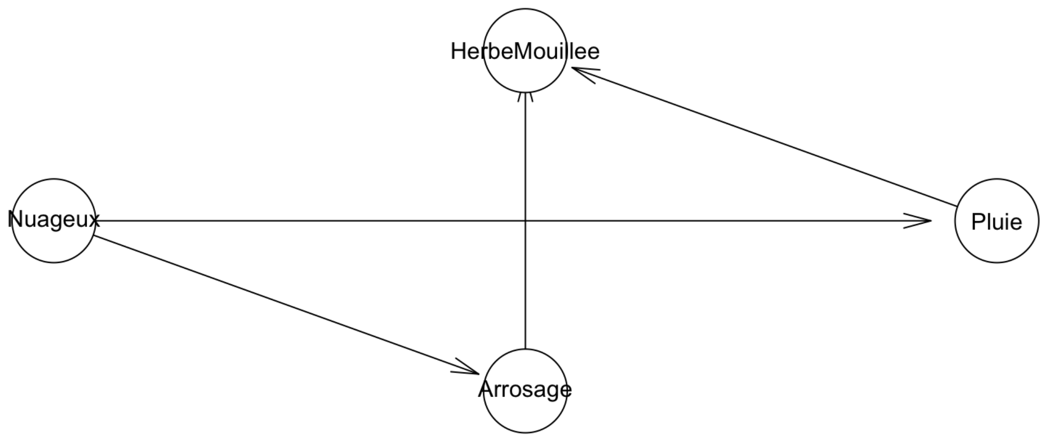
Sprinkler Exercice

Let us define the structure of the network

```r
library(bnlearn)
library(visNetwork)
variables<-c("Nuageux","Arrosage","Pluie","HerbeMouillee")
net<-empty.graph(variables)
adj = matrix(0L, ncol = 4, nrow = 4, dimnames=list(variables, variables))
adj["Nuageux","Arrosage"]<-1
adj["Nuageux","Pluie"]<-1
adj["Arrosage","HerbeMouillee"]<-1
adj["Pluie","HerbeMouillee"]<-1
amat(net)=adj
```

Sprinkler Exercice

```r
#plot.network(net) # for a nice html plot
plot(net)
```



Sprinkler Exercice

Simulate a sample according the model

Basic Simulation with using conditional probability tables

Function for one event (one line of dataframe)

```r
NAPHM1<-function(n){
  N<-rbinom(1,size = 1,prob = 1/2)
  if (N==1)  {A<-rbinom(1,size = 1,prob = 0.1)} else {A<-rbinom(1,size =1,prob = 0.5
  if (N==1)  {P<-rbinom(1,size = 1,prob = 0.8)} else {P<-rbinom(1,size = ,1,prob = 0
 if (A+P==0)  HM<-rbinom(1,size = 1,prob = 0.1) else if
 (A+P==1) HM<-rbinom(1,size = 1,prob = 0.9) else
  HM<-rbinom(1,size = 1,prob = 0.99)
 X<-as.logical(c(N,A,P,HM))
}
```

Basic Simulation with using conditional probability tables

```r
n<-1000
X<-data.frame(t(sapply(1:n,NAPHM1)))
names(X)<-c("Nuageux","Arrosage","Pluie","HerbeMouillee")
head(X)
```

```
  Nuageux Arrosage Pluie HerbeMouillee
1    TRUE    FALSE  TRUE          TRUE
2    TRUE    FALSE  TRUE          TRUE
3    TRUE    FALSE  TRUE          TRUE
4   FALSE     TRUE FALSE          TRUE
5   FALSE     TRUE FALSE         FALSE
6    TRUE     TRUE  TRUE          TRUE
```

Learning the parameters

```r
mean(X$Nuageux) -> pNuageux
lapply(sousTableauxNuageux<-split(X,X$Nuageux),
       function(XsousTableau){mean(XsousTableau$Arrosage)})
lapply(sousTableauxNuageux<-split(X,X$Nuageux),
       function(XsousTableau){mean(XsousTableau$Pluie)})
lapply(sousTableauxNuageux<-split(X,X$Arrosage + X$Pluie),
       function(XsousTableau){mean(XsousTableau$HerbeMouillee)})
```

Back to lecture

Exercices directed Graphical Model

Joint distribution and graphical decomposition (Bishop 8.3)

The joint distribution over three binary variables

Exercices directed Graphical Model

Bishop 8.3

Consider three binary variables $a,\ b,\ c\ \in \{0,1\}$ having the joint distribution given in Table above. Show by direct evaluation that this distribution has the property that a and b are marginally dependent, so that $p(a,b) \neq= p(a)p(b)$, but that they become independent when conditioned on c, so that $p(a,b \mid c) = p(a \mid c)p(b \mid c)$ for both $c = 0$ and $c = 1$.

Exercices directed Graphical Model

Bishop 8.4

Show by direct evaluation that $p(a,b,c) = p(a)p(c \mid a)p(b \mid c)$. Draw the corresponding directed graph.

Local Markov Property

directed local Markov property

$t \perp nd(t) \backslash pa(t) | pa(t)$ where the non-descendants of a node $nd(t)$ are all the nodes except for its descendants

We the topological ordering we have

$$p(x_t | x_1, \cdots, x_{t-1}) = p(x_t | x_{nd(t)}) = p(x_t | x_{pa(t)})$$

Thus

$$\begin{aligned} p(x_t, x_{nd(t)\backslash pa(t)} | x_{pa(t)}) &= p(x_{nd(t)\backslash pa(t)} | x_{pa(t)}) p(x_t | x_{pa(t)}, x_{nc} \\ &= p(x_{nd(t)} | x_{pa(t)}) p(x_t | x_{pa(t)}) \end{aligned}$$

Gaussian mutual information

$$\begin{aligned} I(s,t) =& \mathbb{E}[\log \frac{p(x_s, x_t)}{p(x_s)p(x_t)}] \\ =& -\frac{1}{2}\log \frac{|\mathbf{\Sigma}|}{|diag(\sigma_1^2, \sigma_2^2)|} - \frac{1}{2}\mathbb{E}[z^t \mathbf{\Sigma}^{-1} z - z^t \begin{bmatrix} 1/\sigma_1^2 \\ 0 \end{bmatrix} \\ =& -\frac{1}{2}\log(1-\rho^2) - 1/2 trace(E[zz^t(\Sigma^{-1} - \begin{bmatrix} 1/\sigma_1^2 \\ 0 \end{bmatrix} \\ =& -\frac{1}{2}\log(1-\rho^2) - trace(I - \begin{bmatrix} 1 & \sigma_{12}/\sigma_2^2 \\ \sigma_{12}/\sigma_1^2 & 1 \end{bmatrix}) \\ =& -\frac{1}{2}\log(1-\rho^2) \end{aligned}$$

where $z = \begin{bmatrix} x_s \\ x_t \end{bmatrix} - \begin{bmatrix} \mu_s \\ \mu_t \end{bmatrix}$ and $E[zz^t] = \Sigma$

KL-divergence

Maximizing log-likelihood is equivalent to minimizing KL-divergence

Projects

List 2023

Explain a concept and illustrate with an example:

- 10 minutes OBS recording
- Commented Code Notebook (not a full report).

1. Simulation of images using a Strauss model (Markov Random Field). You may use the paper “Markov Random Field Texture Models” Code for simulation Bonus : estimation of the parameters
2. Programmation of Graphical Lasso. You may use the paper “Sparse inverse covariance estimation with the graphical lasso” Original code of the algorithm illustrated with sachs data
3. Program you own Restricted Boltzmann Machine for prediction. You may use the paper “A Practical Guide to Training Restricted Boltzmann Machines” Original code of the algorithm with illustration on MNIST dataset
4. Structural equation models (SEM) using the NoTears approach. You may use the paper “DAGs with NO TEARS: Continuous Optimization for Structure Learning” Use the code from https://github.com/xunzheng/notears and illustrate with one