Graphical Models
(Lecture 1 - Introduction)

Tibério Caetano

tiberiocaetano.com
Statistical Machine Learning Group
NICTA Canberra

LLSS, Canberra, 2009

Material on Graphical Models

Many good books

- Chris Bishop's book‘“Pattern Recognition and Machine Learning” (Graphical Models chapter available from his webpage in pdf format, as well as all the figures – many used here in these slides!)
- Judea Pearl's “Probabilistic Reasoning in Intelligent Systems”
- Stephen Lauritzen's “Graphical Models”
- . . .

Unpublished material

- Michael Jordan's unpublished book “An Introduction to Probabilistic Graphical Models”
- Koller and Friedman's unpublished book “Structured Probabilistic Models”

Videos

- Sam Roweis' videos on videolectures.net (Excellent!)

Introduction

Query in quotes “ ”	# results in Google Scholar
Kalman Filter	>103,000
EM algorithm	> 64,000
Hidden Markov Models	> 57,000
Bayesian Networks	> 31,600
Markov Random Fields	> 15,000
Particle Filters	> 14,000
Mixture Models	> 43,000
Conditional Random Fields	> 2,500
Markov Chain Monte Carlo	> 76,000
Gibbs Sampling	> 18,000

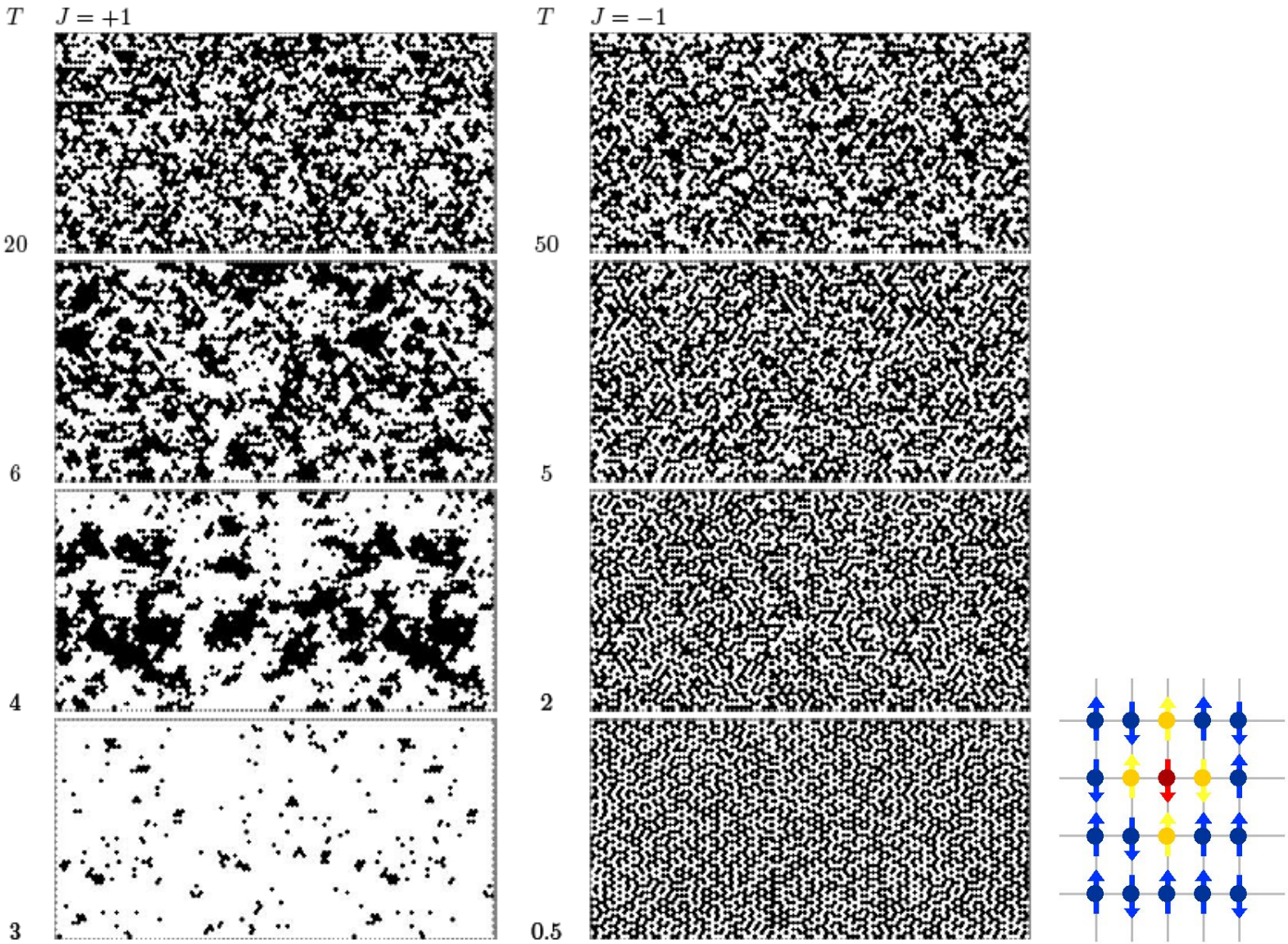
. . .

Introduction

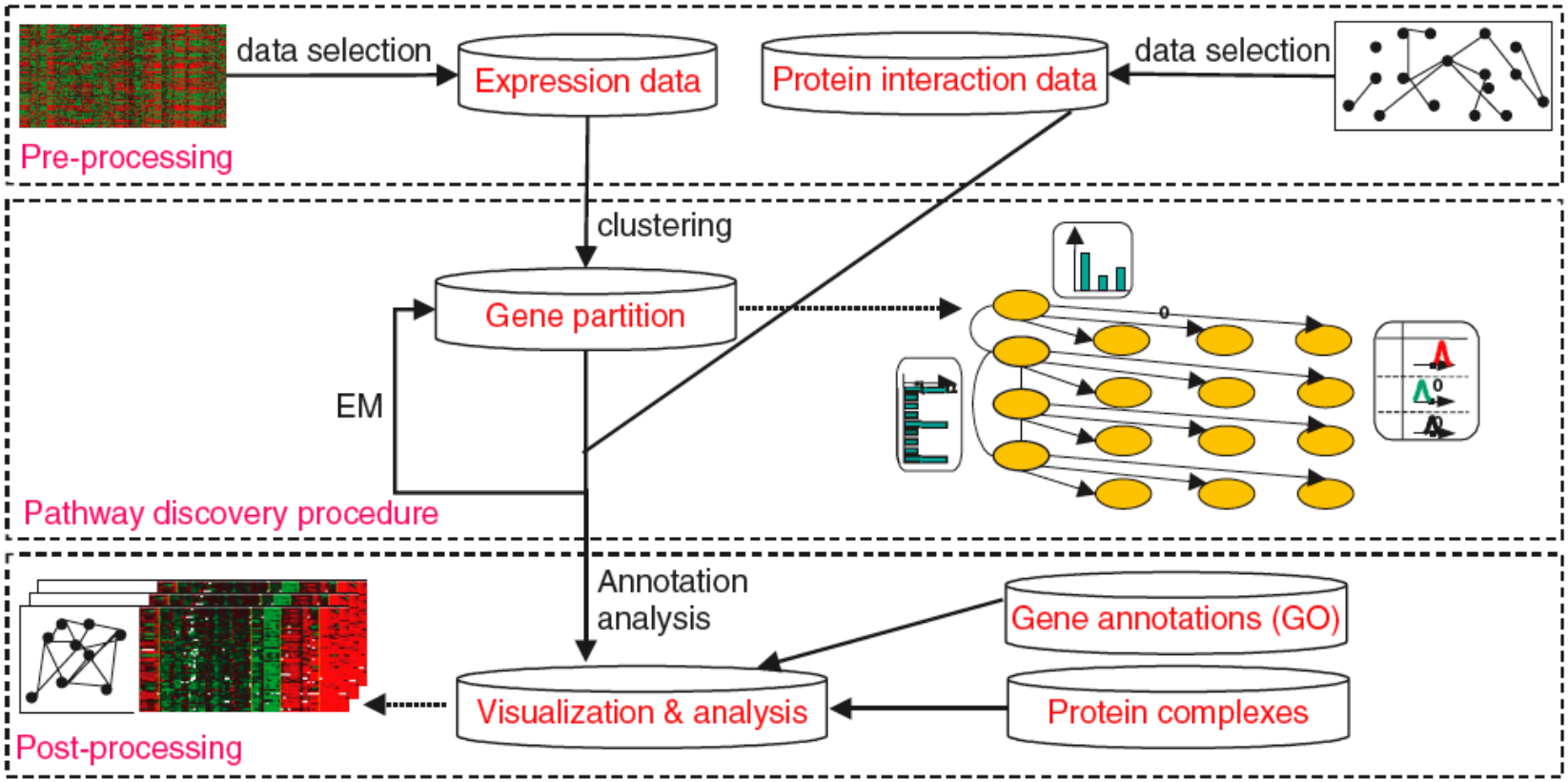
Graphical Models have been applied to

- Image Processing
- Speech Processing
- Natural Language Processing
- Document Processing
- Pattern Recognition
- Bioinformatics
- Computer Vision
- Economics
- Physics
- Social Sciences
- ...

Physics



Biology


data selection
Expression data
Protein interaction data
data selection
Pre-processing
clustering
Gene partition
EM
Pathway discovery procedure
Annotation
analysis
Gene annotations (GO)
Visualization & analysis
Protein complexes
Post-processing

Music

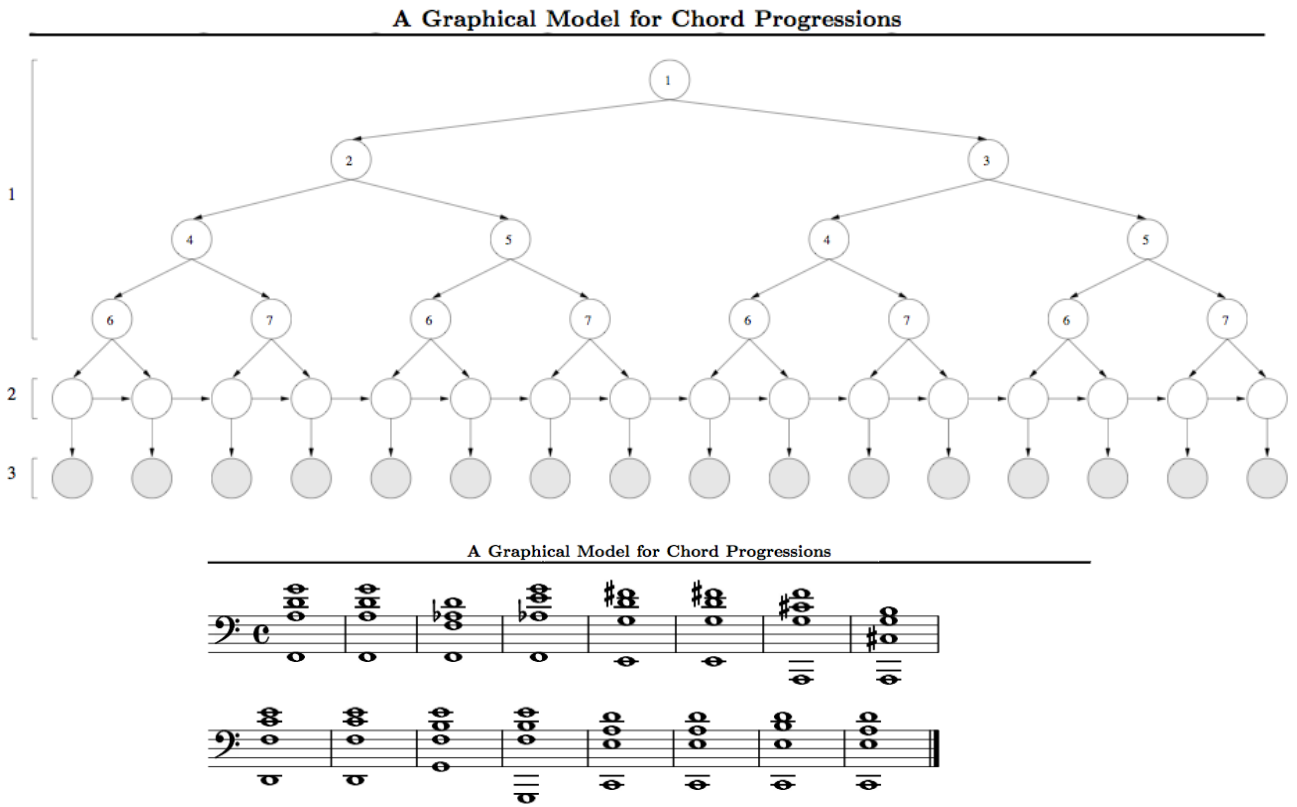


Figure 3. A chord progression generated by the proposed model. This chord progression is very similar to a standard jazz chord progression.

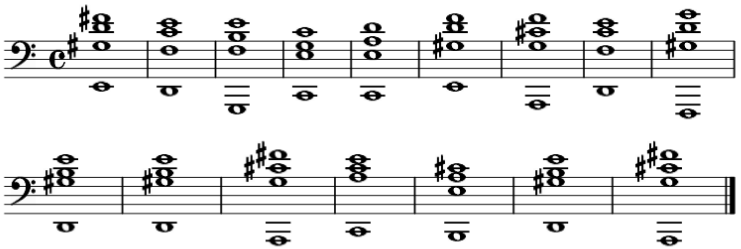
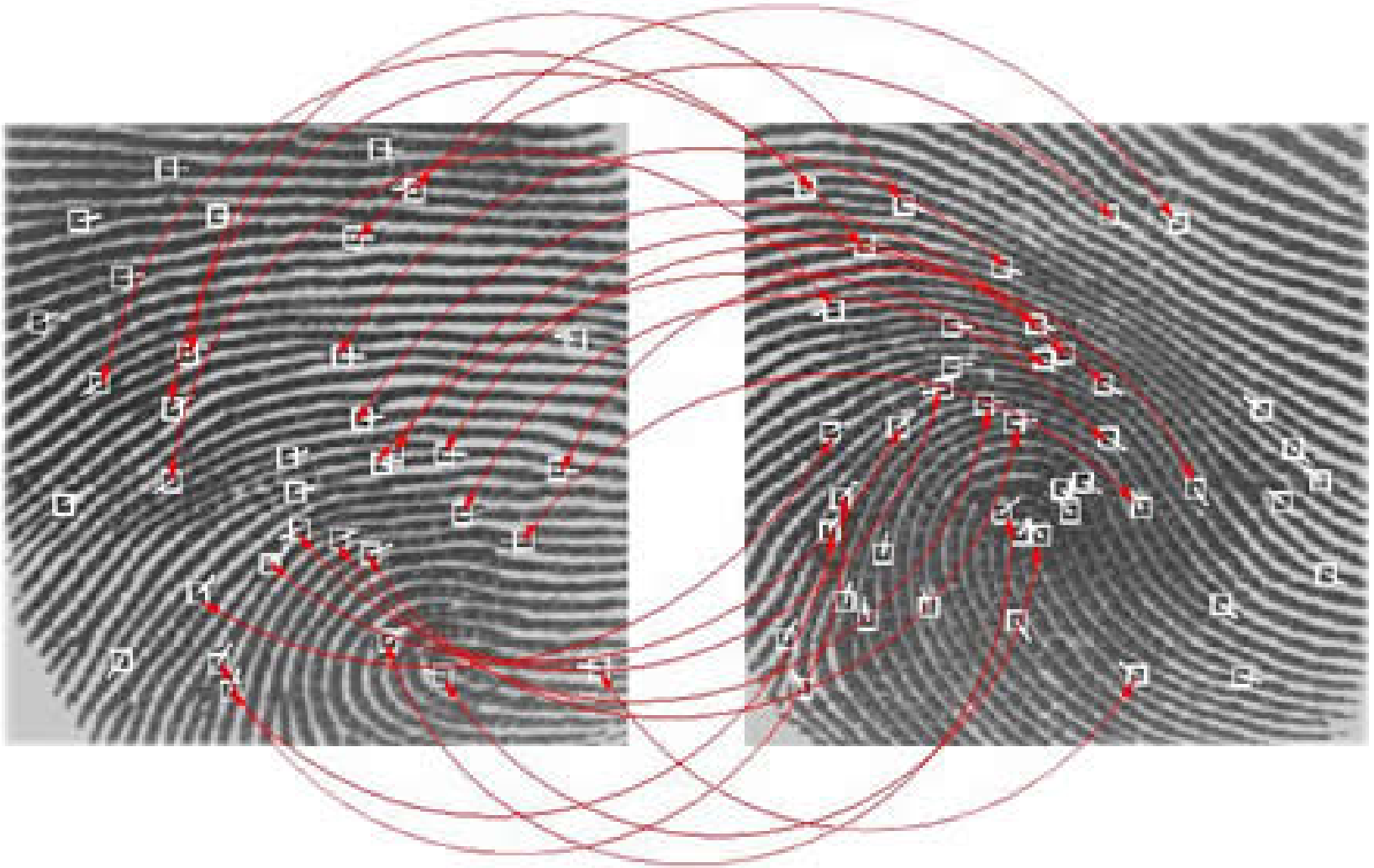


Figure 4. A chord progression generated by the HMM model. While the individual chord transitions are smooth and likely, there is no global chord structure.

Computer Vision



Computer Vision

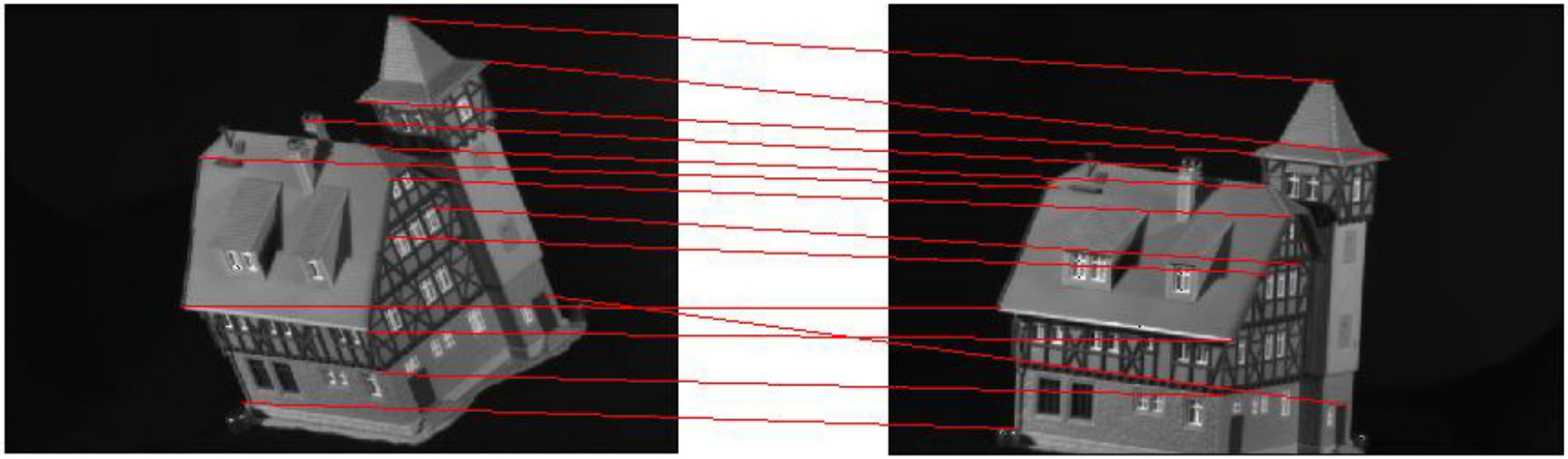


Image Processing

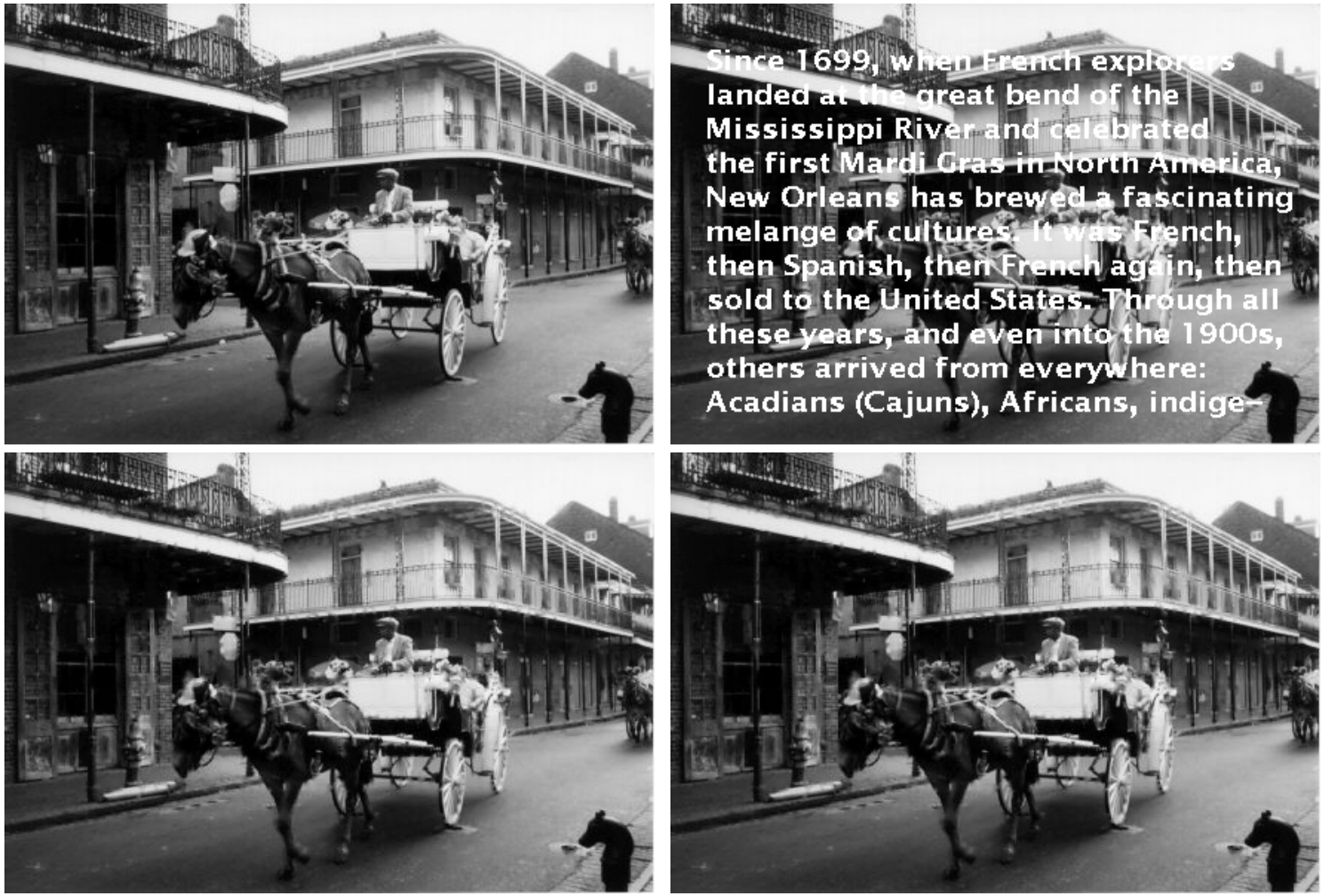


Image Processing

Introduction

Technically, Graphical Models are

Multivariate probabilistic models...

which are structured...

in terms of conditional independence statements

Informally

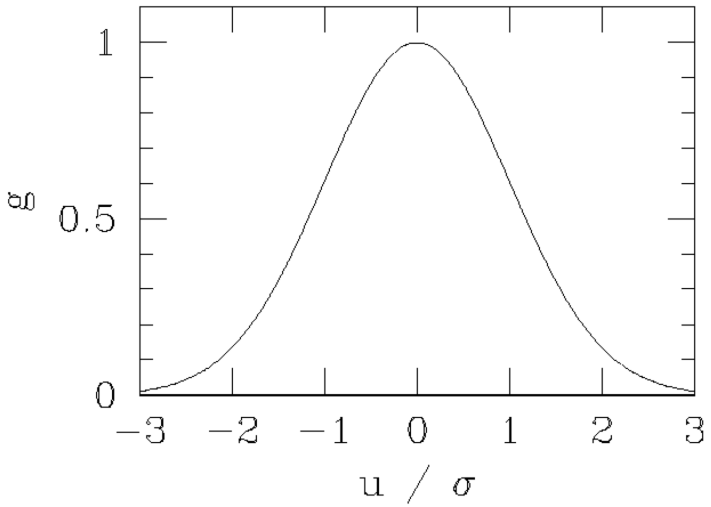
Models that represent a system by its parts and the possible relations among them in a probabilistic way

Introduction

Questions we want to ask about these models

- Estimating the parameters of the model given data
- Obtaining data samples from the model
- Computing probabilities of particular outcomes
- Finding most likely outcome

Univariate Example



$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{[-(x-\mu)^2/(2\sigma^2)]}$$

- Estimate μ and σ given $X = \{x^1, \ldots, x^n\}$
- Sample from $p(x)$
- Compute $P(\mu - \sigma \leq x \leq \mu + \sigma) := \int_{\mu-\sigma}^{\mu+\sigma} p(x)dx$
- Find $\text{argmax}_x\, p(x)$

Multivariate Case

We want to do the same things

- For multivariate distributions structured according to CI
- Efficiently
- Accurately

For example, given $p(x_1, \ldots, x_n; \theta)$

- Estimate θ given a sample X (and criterion, e.g. ML)
- Compute $p(x_A)$, $A \subseteq \{x_1, \ldots, x_n\}$ (marginal distributions)
- Find $\text{argmax}_{x_1,\ldots,x_n} p(x_1, \ldots, x_n; \theta)$ (MAP assignment)

Naively...

When trying to answer the relevant questions...

$$p(x_1) = \sum_{x_2,\ldots,x_N} p(x_1, \ldots, x_N)$$

$$O(|\mathcal{X}_1| \cdot |\mathcal{X}_2| \cdot \ldots \cdot |\mathcal{X}_N|)$$

and similarly for other questions: NOT GOOD

We need compact representations of multivariate distributions

When to Use Graphical Models

- When the compactness of the model arises from conditional independence statements involving its random variables.

- **CAUTION:** Graphical Models are useful in such cases. If the probability space is structured in different ways, Graphical Models may not (and in principle should not) be the right framework to represent and deal with the probability distributions involved.

Graphical Models
(Lecture 2 - Basics)

Tibério Caetano

tiberiocaetano.com
Statistical Machine Learning Group
NICTA Canberra

LLSS, Canberra, 2009

Notation

Basic definitions involving random quantities

- X - A random variable $X = (X_1, \ldots, X_N)$, $N \geq 1$
- x - A particular realization of X: $x = (x_1, \ldots, x_N)$
- $\mathcal{X}$ - Set of all realizations (sample space)
- X_A - A random vector of variables indexed by $A \subseteq \{1, \ldots, N\}$ (x_A for realizations)
- $X_{\tilde{A}}$ - The random vector comprised of all variables other than those in X_A ($A \cup \tilde{A} = \{1, \ldots, N\}$, $A \cap \tilde{A} = \emptyset$). $x_{\tilde{A}}$ for realizations
- $\mathcal{X}_A := \{x_A\}$, ($\mathcal{X}_A := \{x_A\}$)
- $p(x) :=$ probability that X assumes realization x

Basic properties of probabilities

- $0 \leq p(x) \leq 1, \forall x \in \mathcal{X}$
- $\sum_{x \in \mathcal{X}} p(x) = 1$

Conditioning and Marginalization

The two 'rules' you will always need

Conditioning

- $p(x_A, x_B) = p(x_A|x_B)p(x_B)$, for $p(x_B) > 0$

Marginalization

- $p(x_A) = \sum_{x_{\tilde{A}} \in \mathcal{X}_{\tilde{A}}} p(x_A, x_{\tilde{A}})$

Independence and Conditional Indep.

Independence

- $p(x_A, x_B) = p(x_A)p(x_B)$

Conditional Independence

- $p(x_A, x_B|x_C) = p(x_A|x_C)p(x_B|x_C)$, or equivalently
- $p(x_A|x_B, x_C) = p(x_A|x_C)$, or equivalently
- $p(x_B|x_A, x_C) = p(x_B|x_C)$

Notation: $X_A \perp\!\!\!\perp X_B \mid X_C$

Conditional Independence

Examples

- Weather tomorrow $\perp\!\!\!\perp$ Weather yesterday $\mid$ Weather today
- My Genome $\perp\!\!\!\perp$ my grandparents' Genome $\mid$ my parent's Genome
- My mood $\perp\!\!\!\perp$ my wife's boss mood $\mid$ my wife's mood
- A pixel's color $\perp\!\!\!\perp$ color of far away pixels $\mid$ color of surrounding pixels

Conditional Independence

The KEY Fact is

$$\underbrace{p(x_A, x_B|x_C)}_{f_1(\text{3 variables})} = \underbrace{p(x_A|x_C)}_{f_2(\text{2 variables})} \times \underbrace{p(x_B|x_C)}_{f_3(\text{2 variables})}$$

p factors as functions over proper subsets of variables

Conditional Independence

Therefore

$$\underbrace{p(x_A, x_B|x_C)}_{f_1(\text{3 variables})} = \underbrace{p(x_A|x_C)}_{f_2(\text{2 variables})} \times \underbrace{p(x_B|x_C)}_{f_3(\text{2 variables})}$$

- $p(x_A, x_B|x_C)$ *cannot* assume arbitrary values for arbitrary x_A, x_B, x_C
- If you vary x_A and x_B for fixed x_C, you can only realize probabilities that satisfy the above condition

What is a Graphical Model?

What is a Graphical Model?

Given a set of conditional independence statements for the random vector $X = (X_1, \ldots, X_N)$:

$$\{X_{A_i} \perp\!\!\!\perp X_{B_i} \mid X_{C_i}\}$$

Our object of study will be the family of probability distributions

$$p(x_1, \ldots, x_N)$$

where these statements hold

Questions to be addressed

Typical questions when we have a probabilistic model

- Estimate parameters of the model given data
- Compute probabilities of particular outcomes
- Find particularly interesting realizations (e.g. MAP assignment)

In order to manipulate the probabilistic model

- We need to know the mathematical structure of $p(x)$
- We need to find ways of computing efficiently (and accurately) in such structure

An Exercise

How does $p(x)$ look like?

Example:

$p(x_1, x_2, x_3)$ where $x_1 \perp\!\!\!\perp x_3 \mid x_2$

$p(x_1, x_3|x_2) = p(x_1|x_2)p(x_3|x_2)$

$p(x_1, x_2, x_3) = p(x_1, x_3|x_2)p(x_2) = p(x_1|x_2)p(x_3|x_2)p(x_2)$

so,

$$p(x_1, x_2, x_3) = p(x_1|x_2)p(x_3|x_2)p(x_2)$$

An Exercise

However, $p(x_1, x_2, x_3) = p(x_1|x_2)p(x_3|x_2)p(x_2)$ **is also**

$p(x_1, x_2, x_3) = p(x_1|x_2)p(x_2|x_3)p(x_3)$
since $p(x_3|x_2)p(x_2) = p(x_2|x_3)p(x_3)$

$p(x_1, x_2, x_3) = p(x_3|x_2)p(x_2|x_1)p(x_1)$
since $p(x_1|x_2)p(x_2) = p(x_2|x_1)p(x_1)$

Cond. Indep. and Factorization

So CI seems to generate factorization of $p(x)$!

- Is this useful for the questions we want to ask?

Let's see an example of how expensive it is to compute $p(x_2)$

$p(x_2) = \sum_{x_1,x_3} p(x_1, x_2, x_3)$

Without factorization:

$p(x_2) = \sum_{x_1,x_3} p(x_1, x_2, x_3)$, $O(|\mathcal{X}_1||\mathcal{X}_2||\mathcal{X}_3|)$

With factorization:

$p(x_2) = \sum_{x_1,x_3} p(x_1, x_2, x_3) = \sum_{x_1,x_3} p(x_1|x_2)p(x_2|x_3)p(x_3)$

$p(x_2) = \sum_{x_3} p(x_2|x_3)p(x_3) \sum_{x_1} p(x_1|x_2)$, $O(|\mathcal{X}_2||\mathcal{X}_3|)$

Cond. Indep. and Factorization

Therefore

- Conditional Independence seems to induce a structure in $p(x)$ that allows us to exploit the distributive law in order to make computations more tractable

However, what about the general case $p(x_1, \ldots, x_N)$?

- What is the form that $p(x)$ will take in general, given a set of conditional independence statements?
- Will we be able to exploit the distributive law in this general case as well?

Re-Writing the Joint Distribution

A little exercise

$$p(x_1, \ldots, x_N) = p(x_1, \ldots, x_{N-1}) p(x_N | x_1, \ldots, x_{N-1})$$

$$p(x_1, \ldots, x_N) = p(x_1) p(x_2|x_1) p(x_3|x_1, x_2) \ldots p(x_N|x_1, \ldots, x_{N-1})$$

$$p(x) = \prod_{i=1}^{N} p(x_i | x_{<i})$$

where

"$< i$ ":= $\{j : j < i, j \in \mathbb{N}^+\}$

now denote by π a permutation of the labels $\{1, \ldots, N\}$ such that $\pi_j < \pi_i, \forall i, \forall j \in < i$. Above we have $\pi = \mathbf{1}$ (i.e. $\pi_i = i$)

So we can write

$$p(x) = \prod_{i=1}^{N} p(x_{\pi_i} | x_{<\pi_i}).$$

Re-Writing the Joint Distribution

So

Any $p(x)$ can be written as $p(x) = \prod_{i=1}^{N} p(x_{\pi_i}|x_{<\pi_i})$.

Now, assume that the following CI statements hold

$p(x_{\pi_i}|x_{<\pi_i}) = p(x_{\pi_i}|x_{pa_{\pi_i}}), \forall i$, where $pa_{\pi_i} \subset < \pi_i$.

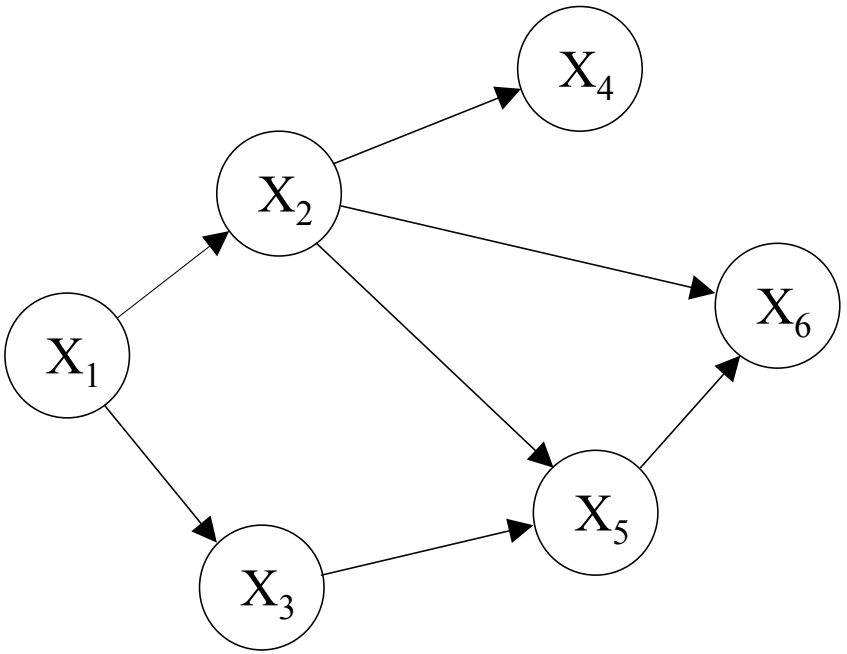
Then we immediately get

$p(x) = \prod_{i=1}^{N} p(x_{\pi_i}|x_{pa_{\pi_i}})$

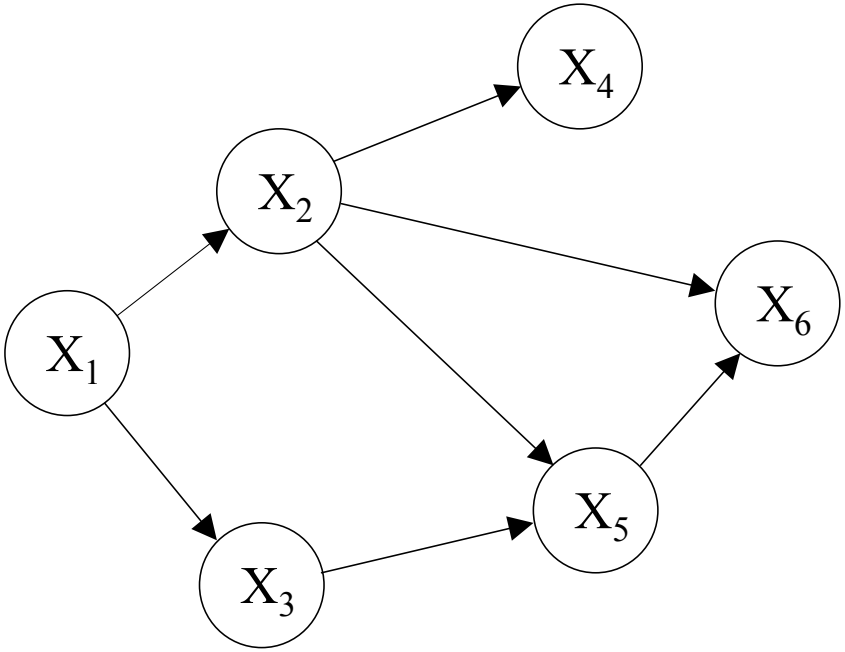
Computing in Graphical Models

Algebra is boring, so let's draw this

- Let's represent variables as circles
- Let's draw an arrow from j to i if $j \in pa_i$
- The resulting drawing will be a Directed Graph
- Moreover it will be Acyclic (no directed cycles) (Exercise:why?)



Computing in Graphical Models



$p(x) =$? (Exercise)

Bayesian Networks

This is why the name “Graphical Models”

Such Graphical Models with arrows are called

- Bayesian Networks
- Bayes Nets
- Bayes Belief Nets
- Belief Networks
- Or, more descriptively: Directed Graphical Models

Bayesian Networks

A Bayesian Network associated to a DAG is a set of probability distributions where each element $p(x)$ can be written as

$$p(x) = \prod_i p(x_i | x_{pa_i})$$

where random variable x_i is represented as a node in the DAG and $pa_i = \{x_j : \exists \text{ arrow } x_j \rightarrow x_i \text{ in the DAG } \}$. "*pa*" is for *parents*.

(Colloquially, we say the BN "is" the DAG)

Topological Sorts

A permutation π of the node labels which, for every node, makes each of its parents have a smaller index than that of the node is called a topological sort of the nodes in the DAG.

Theorem: Every DAG has at least one topological sort

(Exercise: Prove)

A Little Exercise Revisited

Remember?

$$p(x_1, \ldots, x_N) = p(x_1, \ldots, x_{N-1})p(x_N|x_1, \ldots, x_{N-1})$$
$$p(x_1, \ldots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \ldots p(x_N|x_1, \ldots, x_{N-1})$$
$$p(x) = \prod_{i=1}^{N} p(x_i|x_{<i})$$

where

"$< i$ ":= $\{j : j < i, j \in \mathbb{N}^+\}$

now denote by π a permutation of the labels $\{1, \ldots, N\}$ such that $\pi_j < \pi_i, \forall i, \forall j \in < i$. Above we have $\pi = \mathbf{1}$

So we can write

$$p(x) = \prod_{i=1}^{N} p(x_{\pi_i}|x_{<\pi_i}).$$

Exercises

Exercises:

- How many topological sorts has a BN where no CI statements hold?
- How many topological sorts has a BN where all CI statements hold?

Graphical Models
(Lecture 3 -Bayesian Networks)

Tibério Caetano

tiberiocaetano.com
Statistical Machine Learning Group
NICTA Canberra

LLSS, Canberra, 2009

Some Key Elements of Lecture 1

- We can always write $p(x) = \prod_i p(x_i|x_{<i})$
- Create a DAG with arrow $j \mapsto i$ whenever $j \in \ < i$
- Impose CI statements by removing some arrows
- The result will be $p(x) = \prod_i p(x_i|x_{pa(i)})$
- Now there will be permutations π, other than the identity, such that $p(x) = \prod_i p(x_{\pi_i}|x_{pa(\pi_i)})$ with $\pi_i > k$, where $k \in pa(\pi_i)$.

Refreshing Exercise

Exercise

Prove that the factorized form for the probability distribution of a Bayesian Network is indeed normalized to 1.

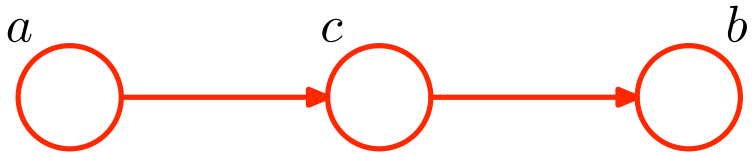
Hidden CI Statements?

We have obtained a BN by

- Introducing very "convenient" CI statements (namely those that shrink the factors of the expansion $p(x) = \prod_{i=1}^{N} p(x_{\pi_i}|x_{<\pi_i})$)
- By doing so, have we induced other CI statements?
- The answer is YES

Head-to-Tail Nodes (Independence)

Are a and b independent?



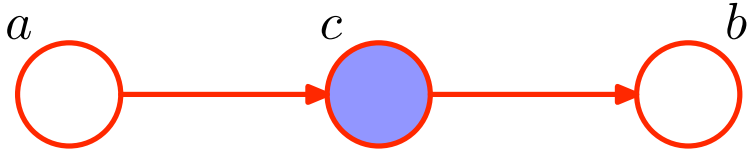
Does $a \perp\!\!\!\perp b$ hold?

Check whether $p(ab) = p(a)p(b)$

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

Head-to-Tail Nodes (Cond. Indep.)

Factorization $\Rightarrow$ CI ?



Does $p(abc) = p(a)p(c|a)p(b|c) \Rightarrow a \perp\!\!\!\perp b \mid c$?

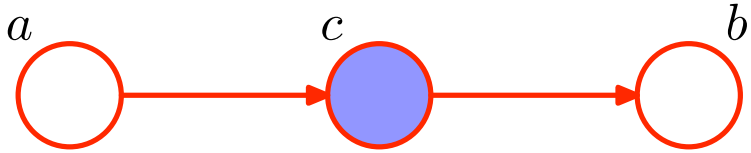
Assume $p(abc) = p(a)p(c|a)p(b|c)$ holds

Then

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

Head-to-Tail Nodes (Cond. Indep.)

CI $\Rightarrow$ Factorization ?



Does $a \perp\!\!\!\perp b \mid c \Rightarrow p(abc) = p(a)p(c|a)p(b|c)$?

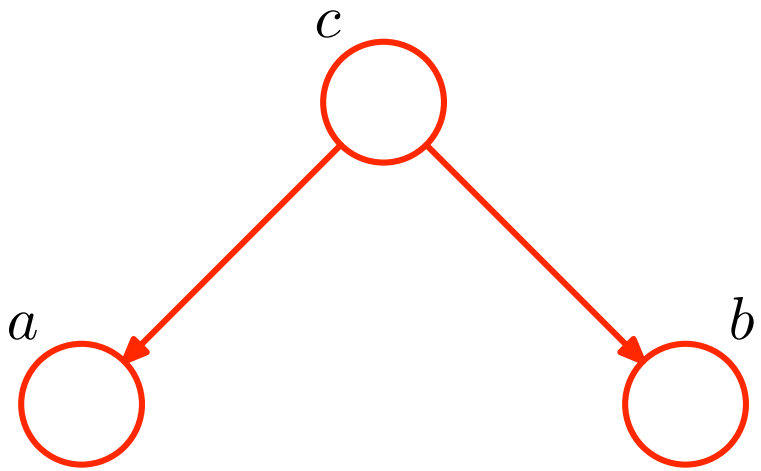
Assume $a \perp\!\!\!\perp b \mid c$, i.e. $p(ab|c) = p(a|c)p(b|c)$

Then

$p(abc) := p(ab|c)p(c) = p(a|c)p(b|c)p(c) \overset{\text{Bayes}}{=} p(a)p(c|a)p(b|c)$

Tail-to-Tail Nodes (Independence)

Are a and b independent?



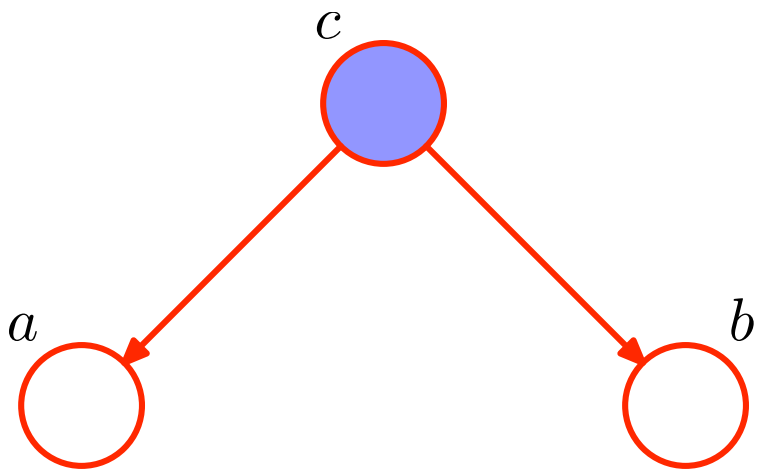
Does $a \perp\!\!\!\perp b$ hold?

Check whether $p(ab) = p(a)p(b)$

$p(ab) = \sum_c p(abc) = \sum_c p(c)p(a|c)p(b|c) =$
$\sum_c p(b)p(a|c)p(c|b) = p(b)p(a|b) \neq p(a)p(b)$, in generai

Tail-to-Tail Nodes (Cond. Indep.)

Factorization $\Rightarrow$ CI ?



Does $p(abc) = p(c)p(a|c)p(b|c) \Rightarrow a \perp\!\!\!\perp b \mid c$?

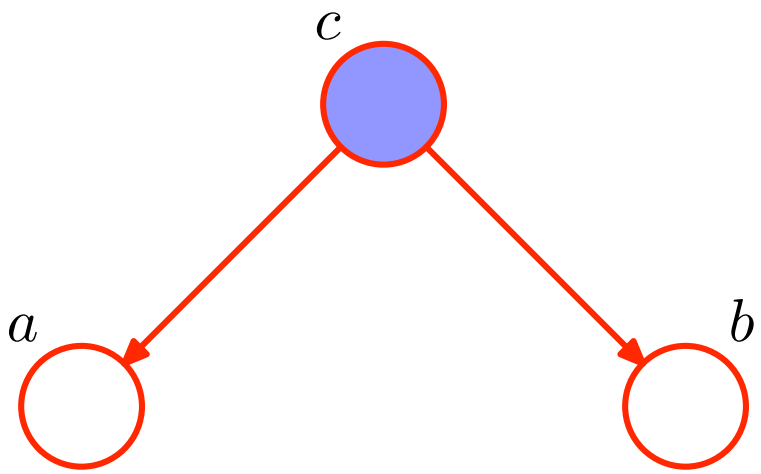
Assume $p(abc) = p(c)p(a|c)p(b|c)$.

Then

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

Tail-to-Tail Nodes (Cond. Indep.)

CI $\Rightarrow$ Factorization ?



Does $a \perp\!\!\!\perp b \mid c \Rightarrow p(abc) = p(c)p(a|c)p(b|c)$?

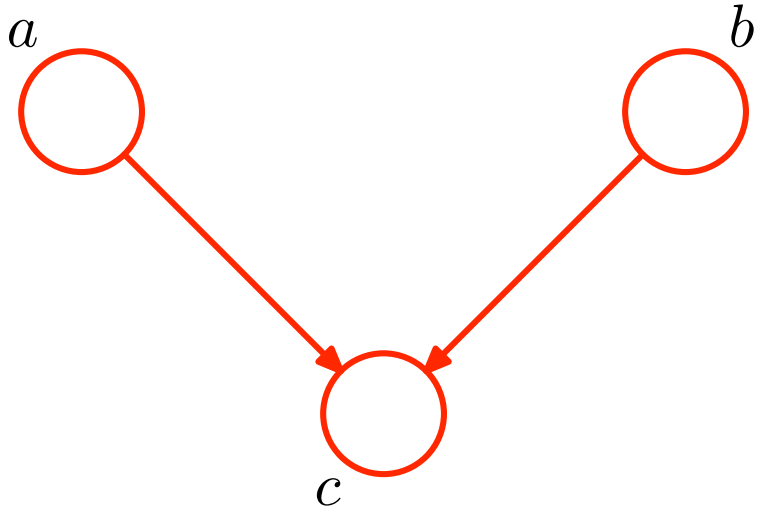
Assume $a \perp\!\!\!\perp b \mid c$, holds, i.e. $p(ab|c) = p(a|c)p(b|c)$ holds

Then

$$p(abc) = p(ab|c)p(c) = p(a|c)p(b|c)p(c)$$

Head-to-Head Nodes (Independence)

Are a and b independent?



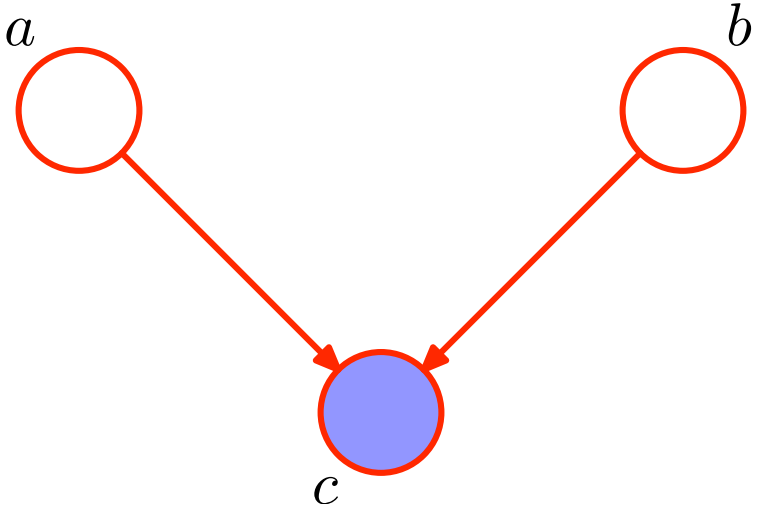
Does $a \perp\!\!\!\perp b$ hold?

Check whether $p(ab) = p(a)p(b)$

$$p(ab) = \sum_c p(abc) = \sum_c p(a)p(b)p(c|ab) = p(a)p(b)$$

Head-to-head Nodes (Cond. Indep.)

Factorization $\Rightarrow$ CI ?



Does $p(abc) = p(a)p(b)p(c|ab) \Rightarrow a \perp\!\!\!\perp b \mid c$?

Assume $p(abc) = p(a)p(b)p(c|ab)$ holds

Then

$$p(ab|c) = \frac{p(abc)}{p(c)} = \frac{p(a)p(b)p(c|ab)}{p(c)} \neq p(a|c)p(b|c) \text{ in general}$$

CI $\Leftrightarrow$ Factorization in 3-Node BNs

Therefore, we conclude that

- Conditional Independence and Factorization are equivalent for the "atomic" Bayesian Networks with only 3 nodes.

Question

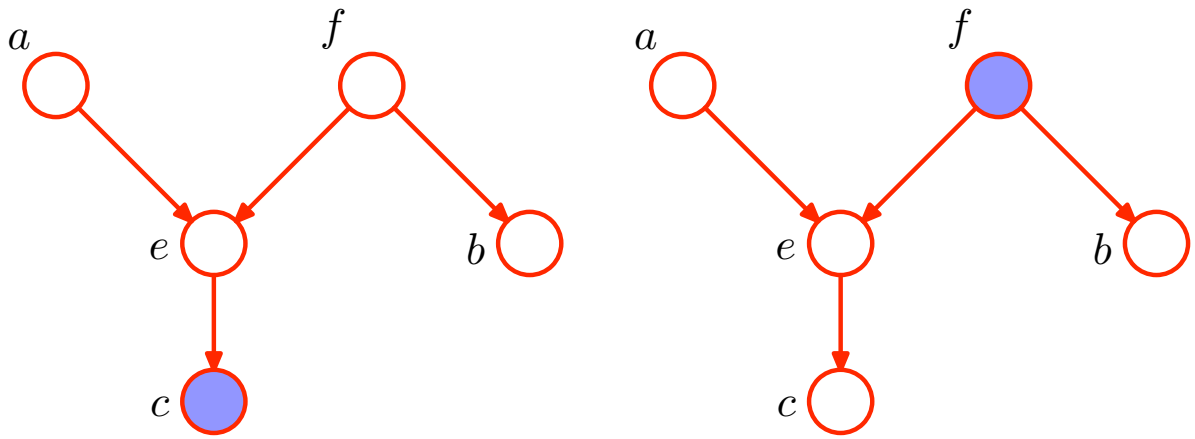
- Are they equivalent for any Bayesian Network?

- To answer we need to characterize which conditional independence statements hold for an arbitrary factorization and check whether a distribution that satisfies those statements will have such factorization.

Blocked Paths

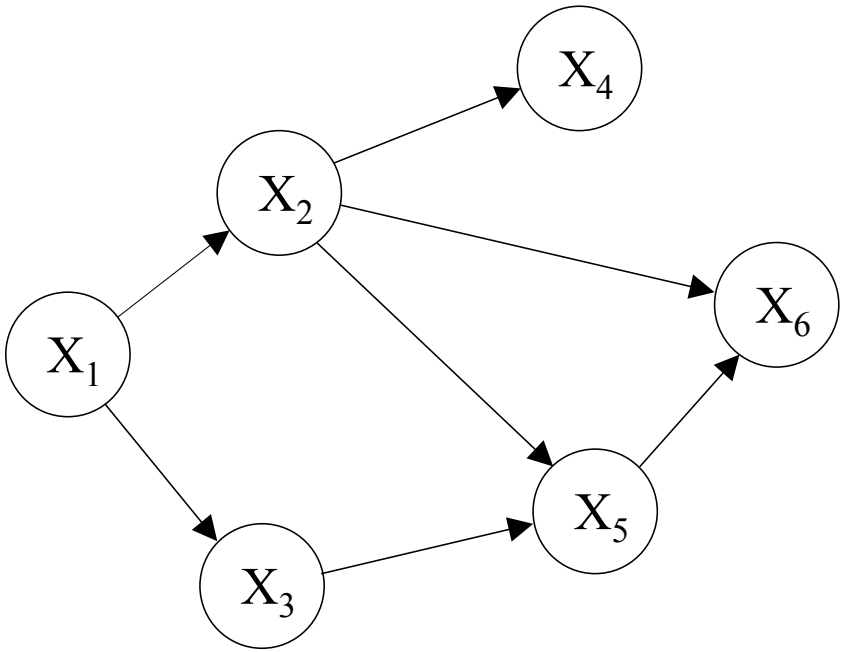
We start by defining a blocked path, which is one containing

- An observed TT or HT node, or
- A HH node which is not observed, nor any of its descendants is observed



D-Separation

- A set of nodes A is said to be d-separated from a set of nodes B by a set of nodes C if every path from A to B is blocked when C is in the conditioning set.



Exercise: Is X_3 d-separated from X_6 when the conditioning set is $\{X_1, X_5\}$?

CI $\Leftrightarrow$ Factorization for BNs

Theorem: Factorization $\Rightarrow$ CI

- If a probability distribution factorizes according to a directed acyclic graph, and if A, B and C are disjoint subsets of nodes such that A is d-separated from B by C in the graph, then the distribution satisfies $A \perp\!\!\!\perp B \mid C$.

Theorem: CI $\Rightarrow$ Factorization

- If a probability distribution satisfies the conditional independence statements implied by d-separation over a particular directed graph, then it also factorizes according to the graph.

Factorization $\Rightarrow$ CI for BNs

Proof Strategy:

$$\text{DF} \Rightarrow \text{d-sep}$$

d-sep: d-separation property

DF: Directed Factorization Property

CI $\Rightarrow$ Factorization for BNs

Proof Strategy:

$$\text{d-sep} \Rightarrow \text{DL} \Rightarrow \text{DF}$$

DL: Directed Local Markov Property: $\alpha \perp\!\!\!\perp \mathit{nd}(\alpha) \mid \mathit{pa}(\alpha)$

Thus we obtain DF $\Rightarrow$ d-sep $\Rightarrow$ DL $\Rightarrow$ DF

Relevance of CI $\Leftrightarrow$ Factorization for BNs

Has local, wants global

- CI statements are usually what is known by the expert
- The expert needs the model $p(x)$ in order to compute things
- The CI $\Rightarrow$ Factorization part gives $p(x)$ from what is known (CI statements)

Graphical Models
(Lecture 4 - Markov Random Fields)

Tibério Caetano

tiberiocaetano.com
Statistical Machine Learning Group
NICTA Canberra

LLSS, Canberra, 2009

Changing the class of CI statements

We obtained BNs by assuming

- $p(x_{\pi_i}|x_{<\pi_i}) = p(x_{\pi_i}|x_{pa_{\pi_i}}), \forall i$, where $pa_{\pi_i} \subset < \pi_i$.
- We saw in general that such types of CI statements would produce others, and in general all CI statements can be read as d-separation in a DAG.
- However, there are sets of CI statements which cannot be satisfied by any BN.

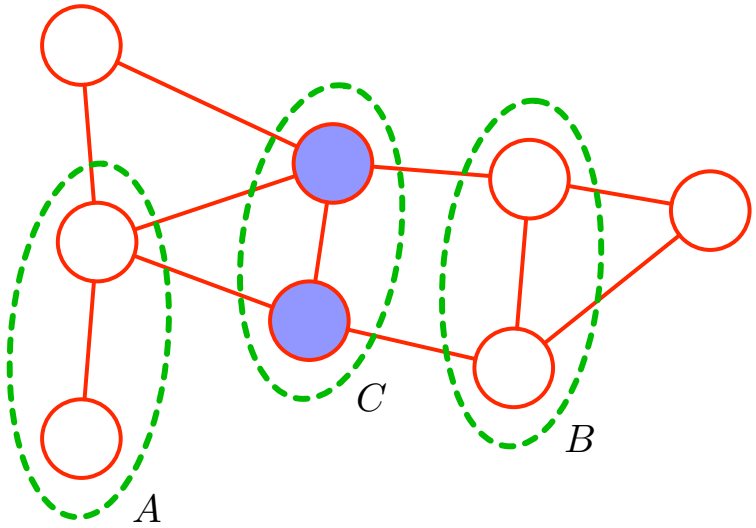
Markov Random Fields

- Ideally we would like to have more freedom
- There is another class of Graphical Models called Markov Random Fields (MRFs)
- MRFs allow for the specification of a different class of CI statements
- The class of CI statements for MRFs can be easily defined by graphical means in undirected graphs.

Graph Separation

Definition of Graph Separation

- In an undirected graph G, being A, B and C disjoint subsets of nodes, if every path from A to B includes at least one node from C, then C is said to separate A from B in G.



Graph Separation

Definition of Markov Random Field

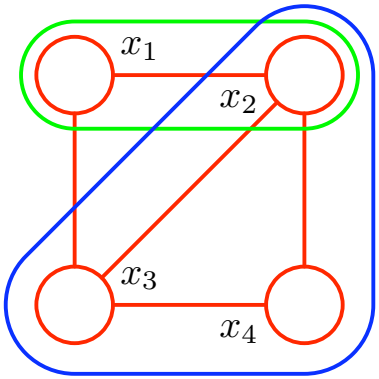
- An MRF is a set of probability distributions $\{p(x) : p(x) > 0 \ \forall p, x\}$ such that there exists an undirected graph G with disjoint subsets of nodes A, B, C, in which whenever C separates A from B in G, $A \perp\!\!\!\perp B \mid C$ in $p(x)$, $\forall p(x)$

- Colloquially, we say that the MRF "is" such undirected graph. But in reality it is the set of all probability distributions whose conditional independency statements are precisely those given by graph separation in the graph.

Cliques and Maximal Cliques

Definitions concerning undirected graphs

- A clique of a graph is a complete subgraph of it (i.e. a subgraph where every pair of nodes is connected by an edge).
- A maximal clique of a graph is clique which is not a proper subset of another clique



$\{X_1, X_2\}$ form a clique and $\{X_2, X_3, X_4\}$ a maximal clique.

Factorization Property

Definition of factorization w.r.t. an undirected graph

A probability distribution $p(x)$ is said to factorize with respect to a given undirected graph if it can be written as

$$p(x) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(x_c)$$

where $\mathcal{C}$ is the set of maximal cliques, c is a maximal clique, x_c is the domain of x restricted to c and $\psi_c(x_c)$ is an arbitrary non-negative real-valued function. Z ensures $\sum_x p(x) = 1$.

CI $\Leftrightarrow$ Factorization for positive MRFs

Theorem: Factorization $\Rightarrow$ CI

- If a probability distribution factorizes according to an undirected graph, and if A, B and C are disjoint subsets of nodes such that C separates A from B in the graph, then the distribution satisfies $A \perp\!\!\!\perp B \mid C$.

Theorem: CI $\Rightarrow$ Factorization (Hammersley-Clifford)

- If a strictly positive probability distribution ($p(x) > 0 \; \forall x$) satisfies the conditional independence statements implied by graph separation over a particular undirected graph, then it also factorizes according to the graph.

Factorization $\Rightarrow$ CI for MRFs

Proof...

CI $\Rightarrow$ Factorization for +MRFs (H-C Thm)

Möbius Inversion: for $C \subseteq B \subseteq A \subseteq S$ and $F : \mathcal{P}(S) \mapsto \mathbb{R}$:

$$F(A) = \sum_{B:B\subseteq A} \sum_{C:C\subseteq B} (-1)^{|B|-|C|} F(C)$$

Define $F = \phi = \log p$ and compute the inner sum for the case where B is not a clique (i.e. $\exists X_1, X_2$ not connected in B). Then CI $\phi(X_1, C, X_2) + \phi(C) = \phi(C, X_1) + \phi(C, X_2)$ holds and

$$\begin{aligned}
\sum_{C\subseteq B} (-1)^{|B|-|C|}\phi(C) = &\sum_{C\subseteq B; X_1, X_2 \notin C} (-1)^{|B|-|C|}\phi(C) + \\
&\sum_{C\subseteq B; X_1, X_2 \notin C} (-1)^{|B|-|C\cup X_1|}\phi(C, X_1) + \\
&\sum_{C\subseteq B; X_1, X_2 \notin C} (-1)^{|B|-|C\cup X_2|}\phi(C, X_2) + \\
&\sum_{C\subseteq B; X_1, X_2 \notin C} (-1)^{|B|-|X_1\cup C\cup X_2|}\phi(X_1, C, X_2) = \\
= \sum_{C\subseteq B; X_1, X_2 \notin C} (-1)^{|B|-|C|} &\underbrace{[\phi(X_1, C, X_2) + \phi(C) - \phi(C, X_1) - \phi(C, X_2)]}_{=0}
\end{aligned}$$

Relevance of CI $\Leftrightarrow$ Factorization for MRFs

Relevance is analogous to the BN case

- CI statements are usually what is known by the expert
- The expert needs the model $p(x)$ in order to compute things
- The CI $\Rightarrow$ Factorization part gives $p(x)$ from what is known (CI statements)

Comparison BNs vs. MRFs

In both types of Graphical Models

- A relationship between the CI statements satisfied by a distribution and the associated simplified algebraic structure of the distribution is made in term of graphical objects.
- The CI statements are related to concepts of separation between variables in the graph.
- The simplified algebraic structure (factorization of $p(x)$ in this case) is related to "local pieces" of the graph (child + its parents in BNs, cliques in MRFs)

Comparison BNs vs. MRFs

Differences

- The set of probability distributions that can be represented as MRFs is different from the set that can be represented as BNs.
- Although both MRFs and BNs are expressed as a factorization of local functions on the graph, the MRF has a normalization constant $Z = \sum_x \prod_{c \in \mathcal{C}} \psi_c(x_c)$ that couples all factors, whereas the BN has not.
- The local "pieces" of the BN are probability distributions themselves, whereas in MRFs they need only be non-negative functions (i.e. they may not have range [0 1] as probabilities do).

Comparison BNs vs. MRFs

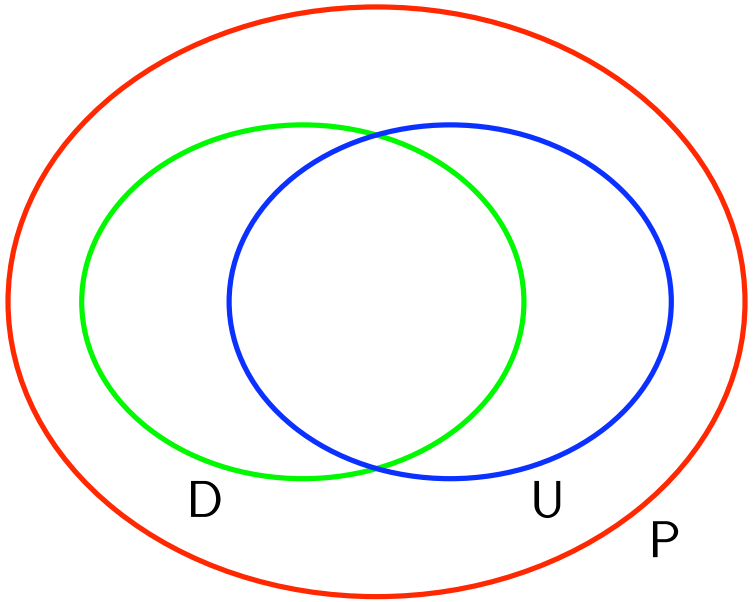
Exercises

- When are the CI statements of a BN and a MRF precisely the same?
- A graph has 3 nodes, A, B and C. We know that $A \perp\!\!\!\perp B$, but $C \perp\!\!\!\perp A$ and $C \perp\!\!\!\perp B$ both do not hold. Can this represent a BN? An MRF?

I-Maps, D-Maps and P-Maps

- A graph is said to be a D-map (for dependence map) of a distribution if every conditional independence statement satisfied by the distribution is reflected in the graph.
- A graph is said to be an I-map (for independence map) of a distribution if every conditional independence statement implied by the graph is satisfied in the distribution.
- A graph is said to be an P-map (for perfect map) of a distribution if it is both a D-map and an I-map for the distribution.

I-Maps, D-Maps and P-Maps

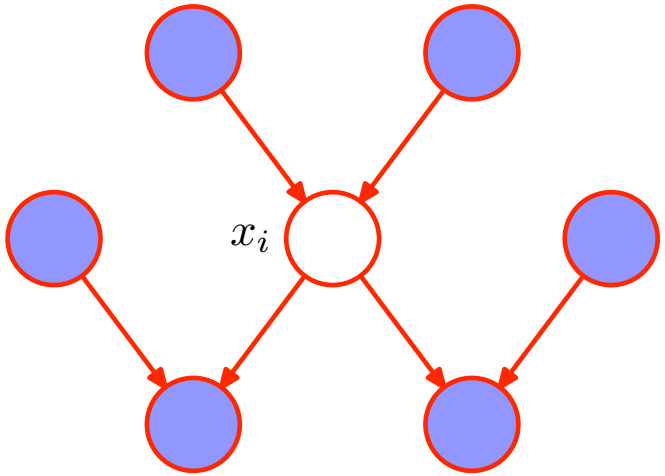


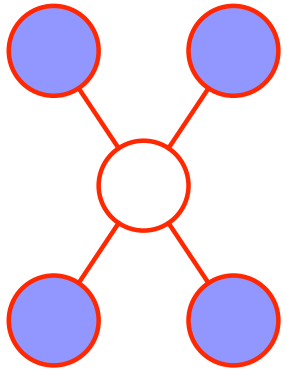
D: set of distributions on *n* variables that can be represented as a perfect map by a DAG

U: set of distributions on *n* variables that can be represented as a perfect map by an Undirected graph

P: set of all distributions on *n* variables

Markov Blankets





- The Markov Blanket of a node X_i in either a BN or an MRF is the smallest set of nodes A such that $p(x_i|x_{\tilde{i}}) = p(x_i|x_A)$
- BN: parents, children and co-parents of the node
- MRF: neighbors of the node

Exercises

Exercises

- Show that the Markov Blanket of a node x_i in a BN is given by it's children, parents and co-parents

- Show that the Markov Blanket of a node x_i in a MRF is given by its neighbors

Graphical Models
(Lecture 5 - Inference)

Tibério Caetano

tiberiocaetano.com
Statistical Machine Learning Group
NICTA Canberra

LLSS, Canberra, 2009

Factorized Distributions

Our $p(x)$ as a factorized form

For BNs, we have

$$p(x) = \prod_i p(x_i|pa_i)$$

for MRFs, we have

$$p(x) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(x_c)$$

Will this enable us to answer the relevant questions in practice?

Key Concept: Distributive Law

Distributive Law

$$\underbrace{ab + ac}_{\text{3 operations}} = \underbrace{a(b + c)}_{\text{2 operations}}$$

I.e. if the same constant factor ('a' here) is present in every term, we can gain by "pulling it out"

Consider computing the marginal $p(x_1)$ for the MRF with factorization

$p(x) = \frac{1}{Z} \prod_{i=1}^{N-1} \psi(x_i, x_{i+1})$ (Exercise: which graph is this?)

$p(x_1) = \sum_{x_2,\ldots,x_N} \frac{1}{Z} \prod_{i=1}^{N-1} \psi(x_i, x_{i+1})$

$p(x_1) = \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_2, x_3) \cdots \sum_{x_N} \psi(x_{N-1}, x_N)$

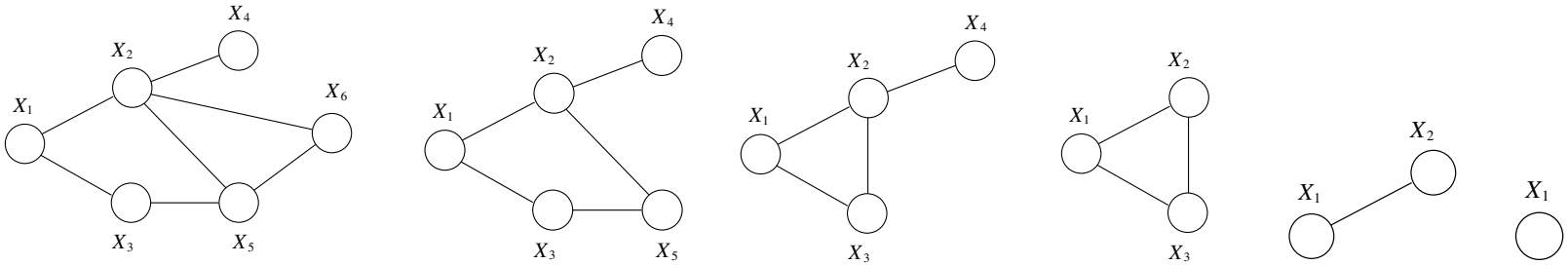
$O(\prod_{i=1}^{N} |\mathcal{X}_i|)$ vs. $O(\sum_{i=1}^{i=N-1} |\mathcal{X}_i||\mathcal{X}_{i+1}|))$

Elimination Algorithm

Distributive Law (DL) is the key to efficient inference in GMs

- The simplest algorithm using the DL is the Elimination Algorithm
- This algorithm is appropriate when we have a single query
- Just like in the previous example of computing $p(x_1)$ in a givel MRF
- This algorithm can be seen as successive elimination of nodes in the graph

Elimination Algorithm



Compute $p(x_1)$ with elimination order $(6, 5, 4, 3, 2)$

$$p(x_1) = Z^{-1} \sum_{x_2,\ldots,x_6} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_5)\psi(x_2, x_5, x_6)\psi(x_2, x_4)$$

$$p(x_1) = Z^{-1} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4) \underbrace{\sum_{x_5} \psi(x_3, x_5) \underbrace{\sum_{x_6} \psi(x_2, x_5, x_6)}_{m_6(x_2, x_5)}}_{m_5(x_2, x_3)}$$

$$p(x_1) = Z^{-1} sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3) \underbrace{\sum_{x_4} \psi(x_2, x_4)}_{m_4(x_2)}$$

$$p(x_1) = Z^{-1} \underbrace{\sum_{x_2} \psi(x_1, x_2) m_4(x_2) \underbrace{\sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3)}_{m_3(x_1, x_2)}}_{m_2(x_1)}$$

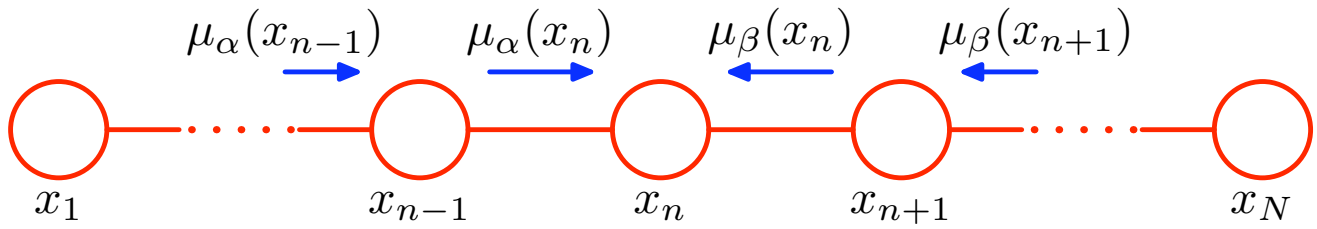
Belief Propagation

Belief Propagation Algorithm, also called

- Probability Propagation
- Sum-Product Algorithm

- Does not repeat computations
- Is specifically targeted at tree-structured graphs

Belief Propagation in a Chain



$$p(x_n) = \sum_{x_{<n}, x_{>n}} \frac{1}{Z} \prod_{i=1}^{N-1} \psi(x_i, x_{i+1})$$

$$p(x_n) = \frac{1}{Z} \sum_{x_{<n}, x_{>n}} \prod_{i=1}^{n-1} \psi(x_i, x_{i+1}) \prod_{i=n}^{N-1} \psi(x_i, x_{i+1})$$

$$p(x_n) = \frac{1}{Z} \left[\sum_{x_{<n}} \prod_{i=1}^{n-1} \psi(x_i, x_{i+1}) \right] \cdot \left[\sum_{x_{>n}} \prod_{i=n}^{N-1} \psi(x_i, x_{i+1}) \right]$$

$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{<n}} \prod_{i=1}^{n-1} \psi(x_i, x_{i+1}) \right]}_{\mu_\alpha(x_n) = O(\sum_{i=1}^{i=n-1} |\mathcal{X}_i||\mathcal{X}_{i+1}|))} \cdot \underbrace{\left[\sum_{x_{>n}} \prod_{i=n}^{N-1} \psi(x_i, x_{i+1}) \right]}_{\mu_\beta(x_n) = O(\sum_{i=n}^{N-1} |\mathcal{X}_i||\mathcal{X}_{i+1}|))}$$

Belief Propagation in a Chain

- So, in order to compute $p(x_n)$, we only need the "incoming messages" to x_n
- But n is arbitrary, so in order to answer an arbitrary query, we need an arbitrary pair of "incoming messages"
- So we need all messages
- To compute a message to the right (left), we need all previous messages coming from the left (right)
- So the protocol should be: start from the leaves up to x_n, then go back towards the leaves
- Chain with N nodes $\Rightarrow 2(N-1)$ messages to be computed

Computing Messages

Defining trivial messages:

$$m_0(x_1) := 1, \quad m_{N+1}(x_N) := 1$$

For $i = 2$ to N compute

$$m_{i-1}(x_i) = \sum_{x_{i-1}} \psi(x_{i-1}, x_i) m_{i-2}(x_{i-1})$$

For $i = N - 1$ back to 1 compute

$$m_{i+1}(x_i) = \sum_{x_{i+1}} \psi(x_i, x_{i+1}) m_{i+2}(x + 1)$$

Belief Propagation in a Tree

- The reason why things are so nice in the chain is that every node can be seen as a leaf after it has received the message from one side (i.e. after the nodes from which the message come have been “eliminated”)
- “Original Leaves” give us the right place to start the computations, and from there the adjacent nodes “become leaves” as well
- However, this property also holds in a tree

Belief Propagation in a Tree

Message Passing Equation

$$m_j(x_i) = \sum_{x_j} \psi(x_j, x_i) \prod_{k:k\sim j, k\neq i} m_k(x_j)$$

$$\left(\prod_{k:k\sim j, k\neq i} m_k(x_j) := 1 \text{ whenever } j \text{ is a leaf}\right)$$

Computing Marginals

$$p(x_i) = \prod_{j:j\sim i} m_j(x_i)$$

Max-Product Algorithm

There are important queries other than computing marginals. For example, we may want to compute the most likely assignment:

$$x^* = \text{argmax}_x \, p(x)$$

as well as its probability

$$p(x^*)$$

one possibility would be to compute $p(x_i) = \sum_{x_{\tilde{i}}} p(x)$ for all i, then $x_i^* = \text{argmax}_{x_i} \, p(x_i)$ and then simply

$$x^* = (x_1^*, x_2^*, \ldots, x_N^*)$$

What's the problem with this?

Exercises

Exercise

- Construct $p(x_1, x_2)$, with $x_1, x_2 \in \{0, 1, 2\}$, such that $p(x_1^*, x_2^*) = 0$ (where $x_i^* = \text{argmax}_{x_i} p(x_i)$)

Max-Product (and Max-Sum) Algorithms

Instead we need to compute directly

$$x^* = \text{argmax}_{x_1,\ldots,x_N} p(x_1, \ldots, x_N)$$

We can use the distributive law again, since

$$\max(ab, ac) = a \max(b, c)$$

for $a > 0$

Exactly the same algorithm applies here with 'max' instead of $\sum$: max-product algorithm.

To avoid underflow we compute x^* via

$$\log(\text{argmax}_x p(x)) = \text{argmax}_x \log p(x) = \text{argmax}_x \sum_s \log f_s(x_s)$$

since log is a monotonic function. We can still use the distributive law since $(\max, +)$ is also a commutative semiring, i.e.

$$\max(a + b, a + c) = a + \max(b, c)$$

A Detail in Max-Sum

After computing the max-marginal for the root x:

$$p_i^* = max_{x_i} \sum_{s \sim x} \mu_{f_s \to x}(x)$$

and its maximizer

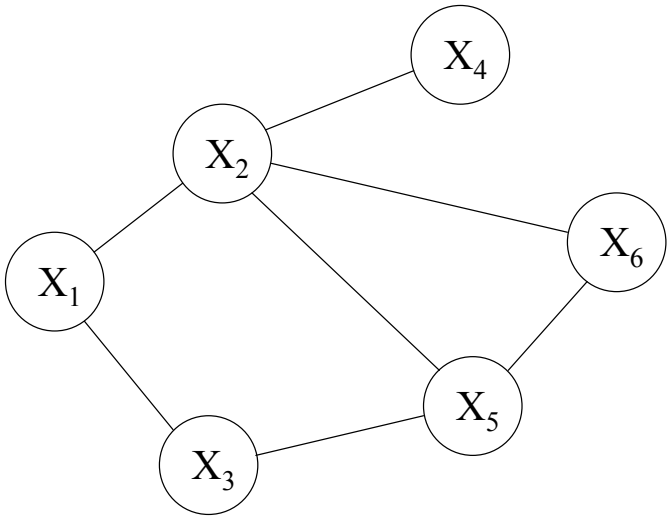
$$x_i^* = \text{argmax}_{x_i} p_i^*$$

It's not a good idea simply to pass back the messages to the leaves and then terminate (Why?)

In such cases it is safer to store the maximizing configurations of previous variables with respect to the next variables and then simply backtrack to restore the maximizing path.

In the particular case of a chain, this is called Viterbi algorithm, an instance of dynamic programming.

Arbitrary Graphs

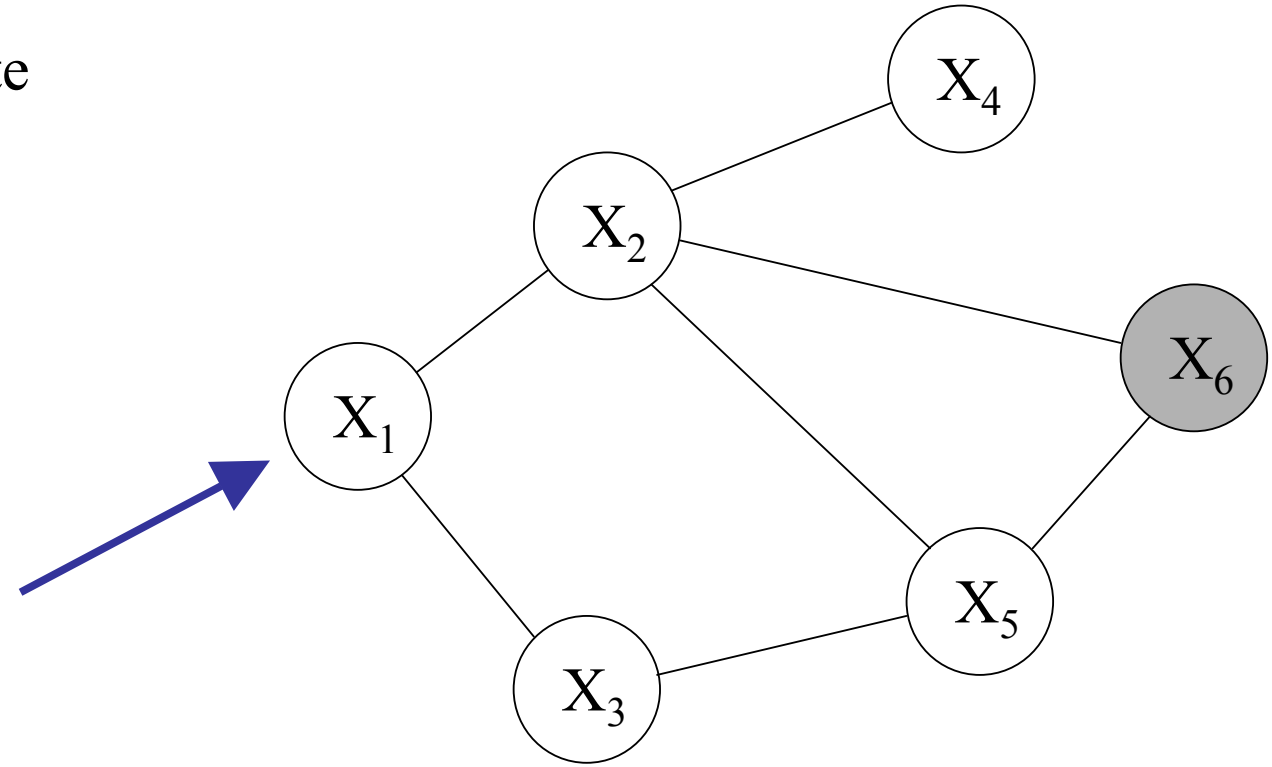


- Elimination algorithm is needed to compute marginals

A Problem with the Elimination Algorithm

How to compute

$$p(x_1 \mid \bar{x}_6)$$



A Problem with the Elimination Algorithm

$$p(x_1, \bar{x}_6) = \frac{1}{Z} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \sum_{x_6} \psi(x_1, x_2) \psi(x_1, x_3) \psi(x_2, x_4) \psi(x_3, x_5) \psi(x_2, x_5, x_6) \delta(x_6, \bar{x}_6)$$

$$p(x_1, \bar{x}_6) = \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4) \sum_{x_5} \psi(x_3, x_5) \sum_{x_6} \psi(x_2, x_5, x_6) \delta(x_6, \bar{x}_6)$$

$$p(x_1, \bar{x}_6) = \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4) \sum_{x_5} \psi(x_3, x_5) m_6(x_2, x_5)$$

$$p(x_1, \bar{x}_6) = \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3) \sum_{x_4} \psi(x_2, x_4)$$

$$p(x_1, \bar{x}_6) = \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) m_4(x_2) \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3)$$

$$p(x_1, \bar{x}_6) = \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) m_4(x_2) m_3(x_1, x_2)$$

$$p(x_1, \bar{x}_6) = \frac{1}{Z} m_2(x_1)$$

A Problem with the Elimination Algorithm

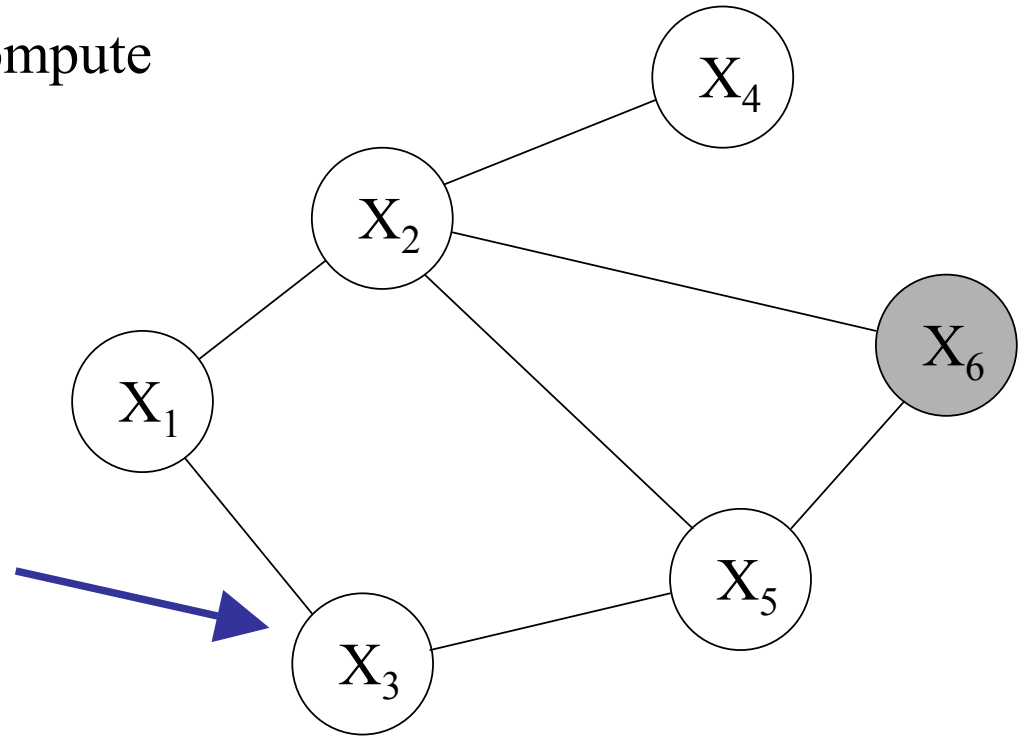
$$p(\bar{x}_6) = \frac{1}{Z}\sum_{x_1} m_2(x_1) \qquad p(x_1, \bar{x}_6) = \frac{1}{Z} m_2(x_1)$$

$$p(x_1 \mid \bar{x}_6) = \frac{m_2(x_1)}{\sum_{x_1} m_2(x_1)}$$

A Problem with the Elimination Algorithm

What if now we want to compute

$$p(x_3 \mid \bar{x}_6)$$



A Problem with the Elimination Algorithm

$$p(x_3,\bar{x}_6)=\frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5}\sum_{x_6}\psi(x_1,x_2)\psi(x_1,x_3)\psi(x_2,x_4)\psi(x_3,x_5)\psi(x_2,x_5,x_6)\delta(x_6,\bar{x}_6)$$

$$p(x_3,\bar{x}_6)=\frac{1}{Z}\sum_{x_1}\psi(x_1,x_3)\sum_{x_2}\psi(x_1,x_2)\sum_{x_4}\psi(x_2,x_4)\sum_{x_5}\psi(x_3,x_5)\sum_{x_6}\psi(x_2,x_5,x_6)\delta(x_6,\bar{x}_6)$$

$$p(x_3,\bar{x}_6)=\frac{1}{Z}\sum_{x_1}\psi(x_1,x_3)\sum_{x_2}\psi(x_1,x_2)\sum_{x_4}\psi(x_2,x_4)\sum_{x_5}\psi(x_3,x_5)m_6(x_2,x_5)$$

$$p(x_3,\bar{x}_6)=\frac{1}{Z}\sum_{x_1}\psi(x_1,x_3)\sum_{x_2}\psi(x_1,x_2)m_5(x_2,x_3)\sum_{x_4}\psi(x_2,x_4)$$

$$p(x_3,\bar{x}_6)=\frac{1}{Z}\sum_{x_1}\psi(x_1,x_3)\sum_{x_2}\psi(x_1,x_2)m_4(x_2)m_5(x_2,x_3)$$

$$p(x_3,\bar{x}_6)=\frac{1}{Z}\sum_{x_1}m_2(x_1,x_3)$$

$$p(x_3,\bar{x}_6)=\frac{1}{Z}m_1(x_3)$$

A Problem with the Elimination Algorithm

Repeated Computations!!

$$p(x_1 \mid \bar{x}_6)$$

$$p(x_3 \mid \bar{x}_6)$$

$$p(x_3,\bar{x}_6) = \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5}\sum_{x_6}\psi(x_1,x_2)\psi(x_1,x_3)\psi(x_2,x_4)\psi(x_3,x_5)\psi(x_2,x_5,x_6)\delta(x_6,\bar{x}_6)$$

$$p(x_3,\bar{x}_6) = \frac{1}{Z}\sum_{x_1}\psi(x_1,x_3)\sum_{x_2}\psi(x_1,x_2)\sum_{x_4}\psi(x_2,x_4)\sum_{x_5}\psi(x_3,x_5)\sum_{x_6}\psi(x_2,x_5,x_6)\delta(x_6,\bar{x}_6)$$

$$p(x_3,\bar{x}_6) = \frac{1}{Z}\sum_{x_1}\psi(x_1,x_3)\sum_{x_2}\psi(x_1,x_2)\sum_{x_4}\psi(x_2,x_4)\sum_{x_5}\psi(x_3,x_5)m_6(x_2,x_5)$$

$$p(x_3,\bar{x}_6) = \frac{1}{Z}\sum_{x_1}\psi(x_1,x_3)\sum_{x_2}\psi(x_1,x_2)m_5(x_2,x_3)\sum_{x_4}\psi(x_2,x_4)$$

$$p(x_3,\bar{x}_6) = \frac{1}{Z}\sum_{x_1}\psi(x_1,x_3)\sum_{x_2}\psi(x_1,x_2)m_4(x_2)m_5(x_2,x_3)$$

$$p(x_3,\bar{x}_6) = \frac{1}{Z}\sum_{x_1} m_2(x_1,x_3)$$

$$p(x_3,\bar{x}_6) = \frac{1}{Z} m_1(x_3)$$

$$p(x_1,\bar{x}_6) = \frac{1}{Z}\sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5}\sum_{x_6}\psi(x_1,x_2)\psi(x_1,x_3)\psi(x_2,x_4)\psi(x_3,x_5)\psi(x_2,x_5,x_6)\delta(x_6,\bar{x}_6)$$

$$p(x_1,\bar{x}_6) = \frac{1}{Z}\sum_{x_2}\psi(x_1,x_2)\sum_{x_3}\psi(x_1,x_3)\sum_{x_4}\psi(x_2,x_4)\sum_{x_5}\psi(x_3,x_5)\sum_{x_6}\psi(x_2,x_5,x_6)\delta(x_6,\bar{x}_6)$$

$$p(x_1,\bar{x}_6) = \frac{1}{Z}\sum_{x_2}\psi(x_1,x_2)\sum_{x_3}\psi(x_1,x_3)\sum_{x_4}\psi(x_2,x_4)\sum_{x_5}\psi(x_3,x_5)m_6(x_2,x_5)$$

$$p(x_1,\bar{x}_6) = \frac{1}{Z}\sum_{x_2}\psi(x_1,x_2)\sum_{x_3}\psi(x_1,x_3)m_5(x_2,x_3)\sum_{x_4}\psi(x_2,x_4)$$

$$p(x_1,\bar{x}_6) = \frac{1}{Z}\sum_{x_2}\psi(x_1,x_2)m_4(x_2)\sum_{x_3}\psi(x_1,x_3)m_5(x_2,x_3)$$

$$p(x_1,\bar{x}_6) = \frac{1}{Z}\sum_{x_2}\psi(x_1,x_2)m_4(x_2)m_3(x_1,x_2)$$

$$p(x_1,\bar{x}_6) = \frac{1}{Z} m_2(x_1)$$

How to avoid that?

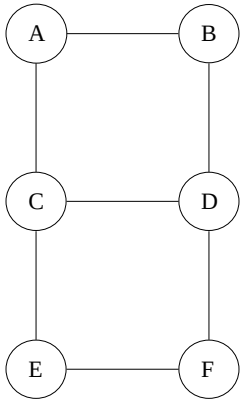
The Junction Tree Algorithm

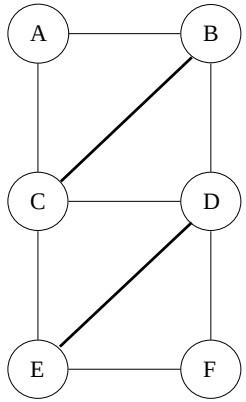
- The Junction Tree Algorithm is a generalization of the belief propagation algorithm for arbitrary graphs
- In theory, it can be applied to any graph (DAG or undirected)
- However, it will be efficient only for certain classes of graphs

Chordal Graphs

Chordal Graphs (also called triangulated graphs)

- The JT algorithm runs on chordal graphs
- A chord in a cycle is an edge connecting two nodes in the cycle but which does not belong to the cycle (i.e. a shortcut in the cycle)
- A graph is chordal if every cycle of length greater than 3 has a chord.

not chordal 



chordal

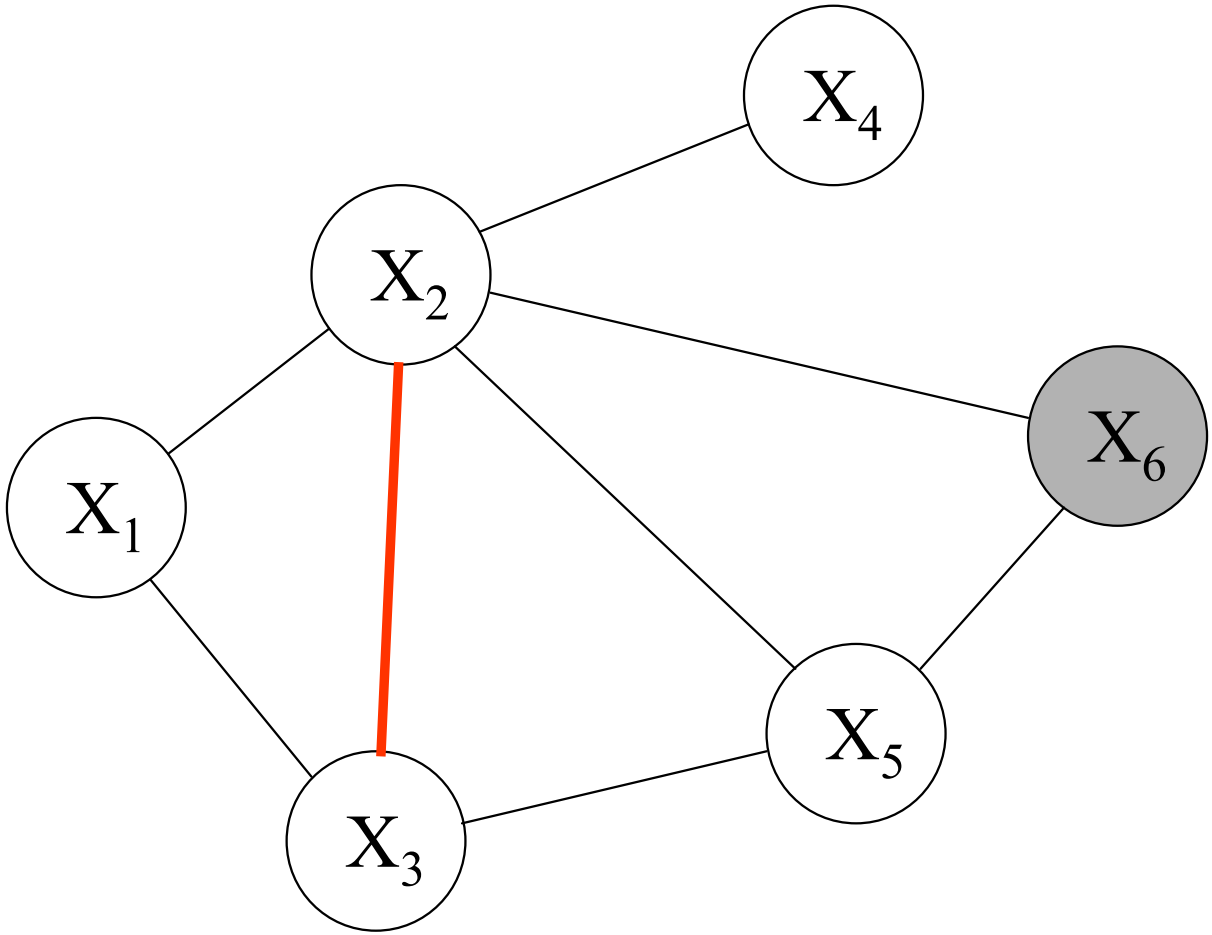
Chordal Graphs

What if a graph is not chordal?

- Add edges until it becomes chordal
- This will change the graph
- Exercise: Why is this not a problem?

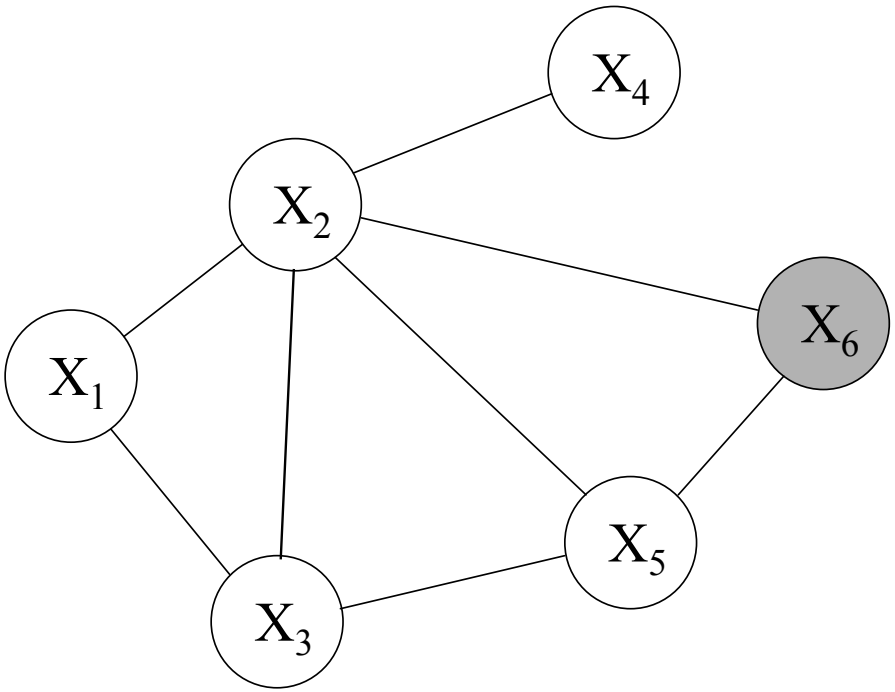
Triangulation Step

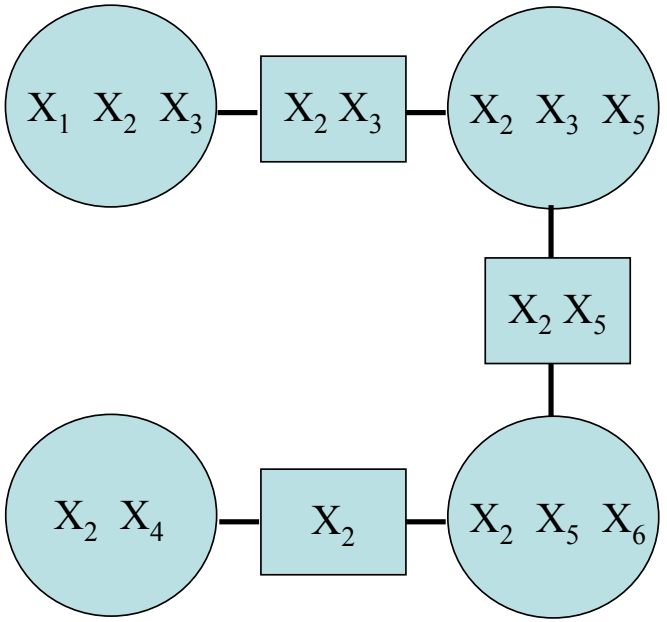
(1) *Triangulate* the graph (if it's not triangulated)



Junction Tree Construction

(2) Create a Junction Tree



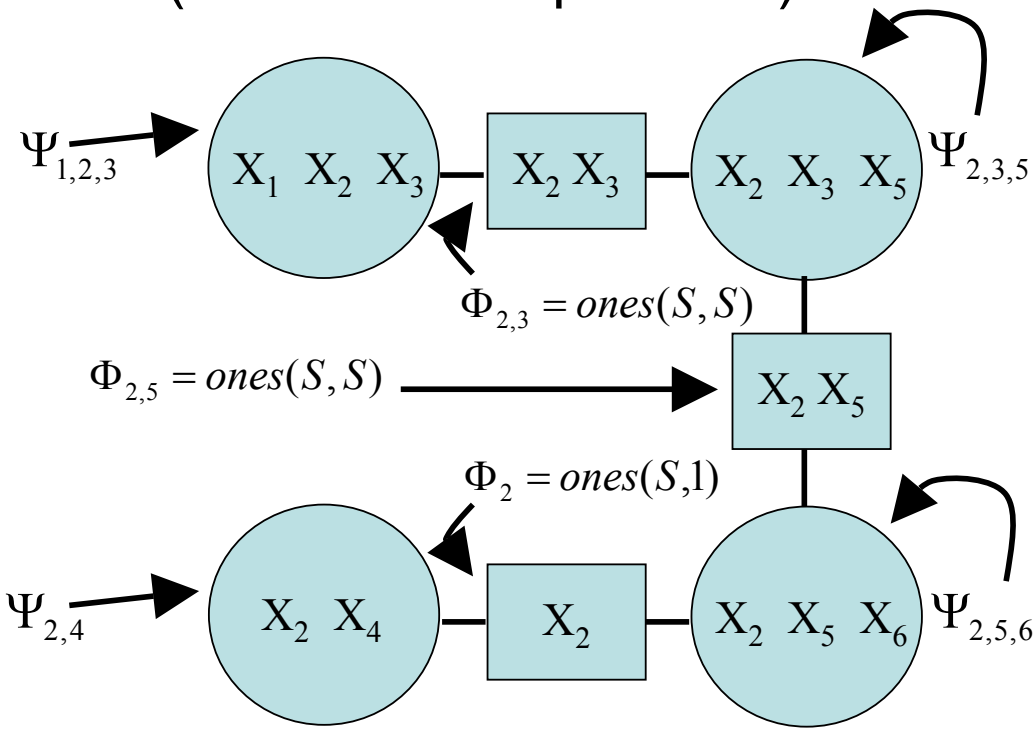


Initialization

(3) Initialize clique potentials (nodes and separators)

Ψ_c Directly introduced

Φ_s Initialized to 1



Propagation

(4) Message passing

$$\Psi^{**}_{1,2,3} = \frac{\Phi^{**}_{2,3}}{\Phi^{*}_{2,3}} \Psi_{1,2,3} \qquad \Phi^{**}_{2,3} = \sum_{x_5} \Psi^{**}_{2,3,5}$$

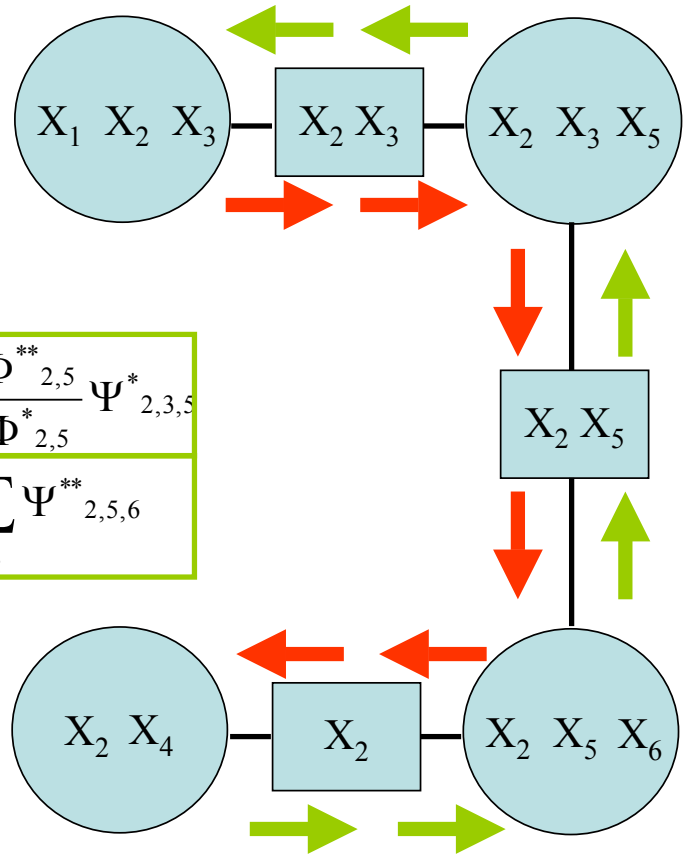
$$\Phi^{*}_{2,3} = \sum_{x_1} \Psi_{1,2,3} \qquad \Psi^{*}_{2,3,5} = \frac{\Phi^{*}_{2,3}}{\Phi_{2,3}} \Psi_{2,3,5}$$

$$\Phi^{*}_{2,5} = \sum_{x_3} \Psi^{*}_{2,3,5} \qquad \Psi^{**}_{2,3,5} = \frac{\Phi^{**}_{2,5}}{\Phi^{*}_{2,5}} \Psi^{*}_{2,3,5}$$

$$\Psi^{*}_{2,5,6} = \frac{\Phi^{*}_{2,5}}{\Phi_{2,5}} \Psi_{2,5,6} \qquad \Phi^{**}_{2,5} = \sum_{x_6} \Psi^{**}_{2,5,6}$$

$$\Psi^{*}_{2,4} = \frac{\Phi^{*}_{2}}{\Phi_{2}} \Psi_{2,4} \qquad \Phi^{*}_{2} = \sum_{x_5 x_6} \Psi^{*}_{2,5,6}$$

$$\Phi^{**}_{2} = \sum_{x_4} \Psi^{*}_{2,4} \qquad \Psi^{*}_{2,5,6} = \frac{\Phi^{**}_{2}}{\Phi^{*}_{2}} \Psi_{2,5,6}$$



Graphical Models
(Lecture 6 - Learning)

Tibério Caetano

tiberiocaetano.com
Statistical Machine Learning Group
NICTA Canberra

LLSS, Canberra, 2009

Learning

We saw that

- Given $p(x;\theta)$, Probabilistic Inference consists of computing
  - Marginals of $p(x;\theta)$
  - Conditional distributions
  - MAP configurations
  - etc.
- However, what is $p(x;\theta)$ in the first place?
- Finding $p(x;\theta)$ from data is called Learning or Estimation or Statistical Inference.

Maximum Likelihood Estimation

In the case of Graphical Models, we've seen that

$$p(x;\theta) = \frac{1}{Z}\prod_{s\in S} f_s(x_s;\theta_s)$$

where $\{s\}$ are subsets of random variables and $\{f_s\}$ are non-negative real-valued functions.

We can re-write that as

$$p(x;\theta) = \exp(\sum_{s\in S} \log f_s(x_s;\theta_s) - g(\theta))$$

where $g(\theta) = \log \sum_x \exp(\sum_{s\in S} \log f_s(x_s;\theta_s))$

Data

IID assumption

- We observe data $X = \{X^1, \ldots, X^m\}$
- We assume every X_i is a sample from the same unknown distribution $p(x; \theta_S)$ (identical assumption)
- We assume X^i and X^j, $i \neq j$, to be drawn independently from $p(x; \theta_S)$ (independence assumption)
- This is the *iid* setting (independently and identically distributed)

Maximum Likelihood Estimation

The joint probability of observing the data X is thus

$$p(X;\theta) = \prod_i p(x^i;\theta) = \prod_i \exp(\sum_s \log f_s(x_s^i;\theta_s) - g(\theta))$$

Seen as a function of θ this is the likelihood function.

The negative log-likelihood is

$$-\log p(X;\theta) = mg(\theta) - \sum_{i=1}^{m} \sum_s \log f_s(x_s^i;\theta_s)$$

Maximum Likelihood Estimation

Maximum likelihood estimation consists of finding θ^* that maximizes the likelihood function, or minimizes the negative log-likelihood:

$$\theta^* = \text{argmin}_\theta \underbrace{\left[mg(\theta) - \sum_{i=1}^{m} \sum_{s} \log f_s(x_s^i; \theta_s) \right]}_{:=\ell(\theta; X)}$$

In order to minimize it we must have $\nabla_\theta \ell(\theta; X) = 0$ and therefore each $\nabla_{\theta_s} \ell(\theta; X) = 0, \forall s$.

What happens for both BNs and MRFs?

ML Estimation in BNs

For BNs, $g(\theta) = 0$ (Exercise) and $\sum_{x_s} f_s(x_s; \theta_s) = 1 \ \ \forall s$ so

$$\nabla_{\theta_{s'}} \left[mg(\theta) - \sum_{i=1}^{m} \sum_{s} \log f_s(x_s^i; \theta_s) + \sum_{s} \lambda_s (1 - \sum_{x_s} f_s(x_s; \theta_s)) \right] =$$

$$\sum_{i=1}^{m} \nabla_{\theta_{s'}} f_{s'}(x_{s'}^i; \theta_{s'}) = \lambda_{s'} \sum_{x_s} \nabla_{\theta_{s'}} f_{s'}(x_s; \theta_{s'}), \forall s'$$

Therefore we have $2|S|$ equations where every pair can be solved independently for $\theta_{s'}$ and $\lambda_{s'}$

So the ML estimation problem decouples on local ML estimation problems involving only the variables in each individual set s.

ML Estimation in MRFs

For MRFs, $g(\theta) \neq 0$ so

$$\nabla_{\theta_s} \left[m g(\theta) - \sum_{i=1}^{m} \sum_{s} \log f_s(x_s^i; \theta_s) \right] = 0 \quad \Rightarrow$$

$$m \nabla_{\theta_s} g(\theta) - \sum_{i=1}^{m} \sum_{s} \nabla_{\theta_s} f_s(x_s^i; \theta_s) = 0, \forall s$$

Therefore we have $|S|$ equations that cannot be solved independently since $g(\theta)$ involves all s. This may give rise to a complex non-linear system of equations.

So, learning in MRFs is more difficult than in BNs.

Exponential Families

Consider the parameterized family of distributions

$$p(x;\theta) = \exp(\langle \Phi(x), \theta \rangle - g(\theta))$$

Such a family of distributions is called an Exponential Family

$\Phi(x)$ is the sufficient statistics

θ is the natural parameter

$g(\theta) = \log \sum_x \exp(\langle \Phi(x), \theta \rangle)$ is the log-partition function

This is the form of several distributions of interest, like Gaussian, binomial, multinomial, Poisson, gamma, Rayleigh, beta, etc.

Exponential Families

If we assume that our $p(x;\theta)$ is an exponential family, the learning problem becomes particularly convenient because it becomes convex (Why?)

Recall the form of $p(x;\theta)$ for a graphical model

$$p(x;\theta) = \exp(\sum_{s\in S} \log f_s(x_s;\theta_s) - g(\theta))$$

for it to be an exponential family we need

$$\sum_{s\in S} \log f_s(x_s;\theta_s) = \langle \Phi(x), \theta \rangle = \sum_s \langle \Phi_s(x_s), \theta_s \rangle$$

Exponential Families for MRFs

For MRFs, the negative log-likelihood now becomes

$$\begin{aligned} -\log p(X;\theta) =& mg(\theta) - \sum_{i=1}^{m} \sum_{s} \left\langle \Phi_s(x_s^i), \theta_s \right\rangle \\ =& mg(\theta) - m \sum_{s} \left\langle \mu_s(x_s), \theta_s \right\rangle \end{aligned}$$

where we defined $\mu_s(x_s) := \sum_{i=1}^{m} \Phi_s(x_s)/m$

Taking the gradient and setting to zero we have

$$\nabla_{\theta_s} mg(\theta) - m \sum_{s} \left\langle \mu_s(x_s), \theta_s \right\rangle = 0 \Rightarrow$$

$$\nabla_{\theta_s} g(\theta) = \mu_s(x_s), \text{ but}$$

$\nabla_{\theta_s} g(\theta) = \mathbb{E}_{x \sim p(x;\theta)}[\Phi_s(x_s)]$ (?), so $\mathbb{E}_{x \sim p(x;\theta)}[\Phi_s(x_s)] = \mu_s(x_s)$

Exponential Families for MRFs

In other words:

- The ML estimate θ^* must be such that the expected value of the sufficient statistics under $p(x;\theta^*)$ for every clique has to match the sample average for the clique.

Why is the problem convex? (Exercise)

Exponential Families for BNs

For BNs, the negative log-likelihood now becomes

$$
\begin{aligned}
-\log p(X;\theta) &= -\sum_{i=1}^{m}\sum_{s}\left\langle \Phi_s(x_s^i),\theta_s\right\rangle \\
&= -m\sum_{s}\left\langle \mu_s(x_s),\theta_s\right\rangle
\end{aligned}
$$

where we also defined $\mu_s(x_s) := \sum_{i=1}^{m}\Phi_s(x_s^i)/m$.
Constructing the Lagrangian corresponding to the constraints $\sum_{x_s}\exp(\langle\Phi(x_s),\theta_s\rangle) = 1, \forall s$, and taking the gradient equal to zero we have

$$
m\cdot\mu_{s'}(x_{s'}) = \lambda_{s'}\mathbb{E}_{x_{s'}\sim p(x_{s'};\theta_{s'})}[\Phi_{s'}(x_{s'})], \forall s'
$$

which can be solved for $\theta_{s'}$ and $\lambda_{s'}$ using

$$
\sum_{x_{s'}}\exp(\langle\Phi(x_{s'}),\theta_{s'}\rangle) = 1
$$

Example: Discrete BNs

Multinomial random variables

- Tabular representation for $p(x_v|x_{pa(v)})$ (define $\phi_v := v \cup pa(v)$)
- One parameter θ_v associated to each $p(x_v|x_{pa(v)})$, i.e. $\theta_v(x_{\phi_v}) := p(x_v|x_{pa(v)};\theta_v)$
- Note that there are no constraints beyond the normalization constraint
- The joint is $p(x_{\mathcal{V}}|\theta) = \prod_v \theta_v(x_{\phi_v})$
- The likelihood is then $\log p(X;\theta) = \log \prod_n p(x_{\mathcal{V},n}|\theta)$
- continuing...

Example: Discrete BNs

$$
\begin{aligned}
\log p(X;\theta) &= \log \prod_n p(x_{\mathcal{V},n}|\theta) \\
&= \sum_n \log \prod_{x_\mathcal{V}} p(x_\mathcal{V};\theta)^{\delta(x_\mathcal{V},x_{\mathcal{V},n})} \\
&= \sum_n \sum_{x_\mathcal{V}} \delta(x_\mathcal{V}, x_{\mathcal{V},n}) \log p(x_\mathcal{V};\theta) \\
&= \sum_{x_\mathcal{V}} m(x_\mathcal{V}) \log p(x_\mathcal{V};\theta) \\
&= \sum_{x_\mathcal{V}} m(x_\mathcal{V}) \log \prod_v \theta_v(x_{\phi_v}) \\
&= \sum_{x_\mathcal{V}} m(x_\mathcal{V}) \sum_v \log \theta_v(x_{\phi_v}) \\
&= \sum_v \sum_{x_{\phi_v}} \left(\sum_{x_{\mathcal{V}\setminus\phi_v}} m(x_\mathcal{V}) \right) \log \theta_v(x_{\phi_v}) \\
&= \sum_v \sum_{x_{\phi_v}} m(x_{\phi_v}) \log \theta_v(x_{\phi_v})
\end{aligned}
$$

Example: Discrete BNs

The Lagrangian is

$$\mathcal{L}(\theta, \lambda) = \sum_{v} \sum_{x_{\phi_v}} m(x_{\phi_v}) \log \theta_v(x_{\phi_v}) + \sum_{v} \lambda_v (1 - \sum_{x_v} \theta_v(x_{\phi_v}))$$

and

$$\nabla_{\theta_{v'}(x_{\phi_{v'}})}(\mathcal{L}(\theta, \lambda)) = \frac{m(x_{\phi_{v'}})}{\theta_{v'}(x_{\phi_{v'}})} - \lambda_{v'} = 0 \quad \Rightarrow \quad \lambda_{v'} = \frac{m(x_{\phi_{v'}})}{\theta_{v'}(x_{\phi_{v'}})}$$

but since $\sum_{x_{v'}} \theta_{v'}(x_{\phi_{v'}}) = 1, \lambda_{v'} = m(x_{pa(v')})$, so

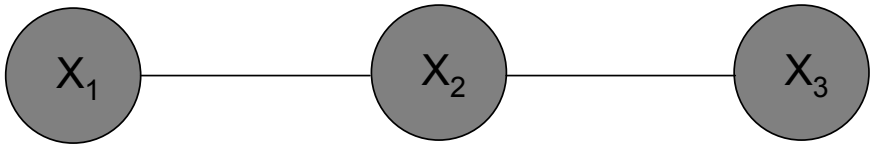
$$\theta_{v'}(x_{\phi_{v'}}) = \frac{m(x_{\phi_{v'}})}{m(x_{pa(v')})} \qquad \text{(Matches intuition)}$$

Learning the Potentials

First – How to learn the potential functions when we have **observed data for all variables** in the model?

Second – How to learn the potential functions when **there are latent (hidden) variables**, i.e., we do not observe data for them?

Learning the Potentials



$$p(x_V) = \frac{1}{Z}\Psi_{1,2}(x_1, x_2)\Psi_{2,3}(x_2, x_3)$$

Assume we observe N instances of this model

For IID sampling, the sufficient statistics are the empirical marginals

$$\tilde{p}(x_1, x_2) \quad \text{and} \quad \tilde{p}(x_2, x_3)$$

How do we estimate $\Psi_{1,2}(x_1, x_2)$ and $\Psi_{2,3}(x_2, x_3)$ from the sufficient statistics?

Learning the Potentials

Let's make a guess:

$$\hat{p}_{ML}(x_1, x_2, x_3) = \frac{\widetilde{p}(x_1, x_2)\widetilde{p}(x_2, x_3)}{\widetilde{p}(x_2)}$$

, so that

$$\begin{cases} \hat{\Psi}^{ML}_{1,2}(x_1, x_2) = \widetilde{p}(x_1, x_2) \\ \hat{\Psi}^{ML}_{2,3}(x_2, x_3) = \dfrac{\widetilde{p}(x_2, x_3)}{\widetilde{p}(x_2)} \end{cases}$$

We can verify that our "guess" is good, because:

$$\hat{p}_{ML}(x_1, x_2) = \sum_{x_3} \frac{\widetilde{p}(x_1, x_2)\widetilde{p}(x_2, x_3)}{\widetilde{p}(x_2)} = \widetilde{p}(x_1, x_2)$$

$$\hat{p}_{ML}(x_2, x_3) = \sum_{x_1} \frac{\widetilde{p}(x_1, x_2)\widetilde{p}(x_2, x_3)}{\widetilde{p}(x_2)} = \widetilde{p}(x_2, x_3)$$

Learning the Potentials

The general recipe is:

(1) For every maximal clique C, set the clique potential to its empirical marginal

(2) For every intersection S between maximal cliques, associate an empirical marginal with that intersection and divide it into the potential of **ONE** of the cliques that form the intersection

This will give ML estimates for decomposable Graphical Models

Decomposable Graphs

A graph is *complete* if E contains all pairs of distinct elements of V.
A graph $G = (V, E)$ is *decomposable* if either

1. G is complete, or
2. We can express V as $V = A \cup B \cup C$ where
   (a) A, B and C are disjoint,
   (b) A and C are non-empty,
   (c) B is complete,
   (d) B separates A and C in G, and
   (e) $A \cup B$ and $B \cup C$ are decomposable.

for decomposable graphs, the derivative of the log-partition function $g(\theta)$ decouples over the cliques (Exercise) $\Rightarrow$ MRF learning easy.

Learning the Potentials

Non-decomposable Graphical Models:

An iterative procedure must be used: Iterative Proportional Fitting (IPF):

$$\frac{p(x_C)}{\Psi_C(x_C)} = \frac{\tilde{p}(x_C)}{\Psi_C(x_C)}$$

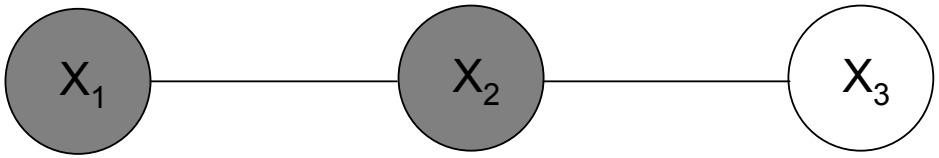
$$\Psi_C^{(t+1)}(x_C) = \Psi_C^{(t)}(x_C)\frac{\tilde{p}(x_C)}{p^{(t)}(x_C)}$$

Where it can be shown that:

$$p^{(t+1)}(x_C) = \tilde{p}(x_C)$$

EM Algorithm

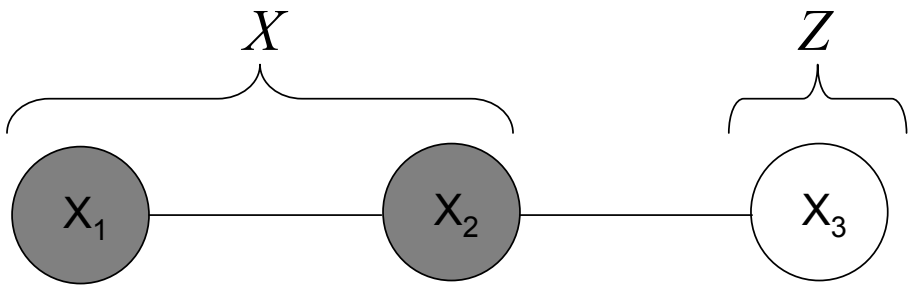
How to estimate the potentials when there are unobserved variables?



Answer: EM algorithm

EM Algorithm

Denote the observed variables by X and the hidden variables by Z



If we knew Z, the problem would reduce to maximizing the complete log-likelihood:

$$l_c(\theta; x, z) = \log p(x, z \mid \theta)$$

EM Algorithm

However, we **don't** observe Z, so the probability of the data X is

$$l(\theta; x) = \log p(x \mid \theta) = \log \sum_z p(x, z \mid \theta)$$

Which is the incomplete log-likelihood

This is the quantity we really want to maximize

Note that now the logarithm cannot transform the product into a sum, since it is "blocked" by the sum over Z, and the optimization does not "decouple"

EM Algorithm

The basic idea of the EM algorithm is:

Given that Z is not observed, we may try to optimize an "averaged" version, over all possible values of Z, of the complete log-likelihood

We do that through an "averaging distribution" q:

$$\left\langle l_c(\theta, x, z) \right\rangle_q = \sum_z q(z \mid x, \theta) \log p(x, z \mid \theta)$$

And obtain the expected complete log-likelihood

The hope then is that maximizing this should at least improve the current estimate for the parameters (so that iteration would eventually maximize the log-likelihood)

EM Algorithm

In order to present the algorithm, we first note that:

$$l(\theta; x) = \log p(x \mid \theta)$$

$$l(\theta; x) = \log \sum_z p(x, z \mid \theta)$$

$$l(\theta; x) = \log \sum_z q(z \mid x) \frac{p(x, z \mid \theta)}{q(z \mid x)}$$

$$\geq \sum_z q(z \mid x) \log \frac{p(x, z \mid \theta)}{q(z \mid x)}$$

$$:= L(q, \theta)$$

Where L is the auxiliary function. The EM algorithm is coordinate-ascent on L

EM Algorithm

The EM algorithm

E - step $$q^{(t+1)} = \arg\max_{q} L(q, \theta^{(t)})$$

M - step $$\theta^{(t+1)} = \arg\max_{\theta} L(q^{(t+1)}, \theta)$$

EM Algorithm

Note that the "M step" is equivalent to maximizing the expected complete log-likelihood:

$$L(q,\theta) = \sum_z q(z \mid x) \log \frac{p(x,z \mid \theta)}{q(z \mid x)}$$

$$L(q,\theta) = \sum_z q(z \mid x) \log p(x,z \mid \theta) - \sum_z q(z \mid x) \log q(z \mid x)$$

$$L(q,\theta) = \left\langle l_c(\theta; x, z) \right\rangle_q - \sum_z q(z \mid x) \log q(z \mid x)$$

Because the second term does not depend on θ

EM Algorithm

The general solution to the "E step" turns out to be

$$q^{(t+1)}(z \mid x) = p(z \mid x, \theta^{(t)})$$

Because

$$L(p(z \mid x, \theta^{(t)}), \theta^{(t)}) = \sum_z p(z \mid x, \theta^{(t)}) \log \frac{p(x, z \mid \theta^{(t)})}{p(z \mid x, \theta^{(t)})}$$

$$L(p(z \mid x, \theta^{(t)}), \theta^{(t)}) = \sum_z p(z \mid x, \theta^{(t)}) \log p(x \mid \theta^{(t)})$$

$$L(p(z \mid x, \theta^{(t)}), \theta^{(t)}) = \log p(x \mid \theta^{(t)})$$

$$L(p(z \mid x, \theta^{(t)}), \theta^{(t)}) = l(\theta^{(t)}; x)$$