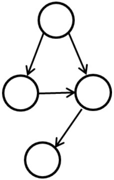
# Inference in Graphical Models

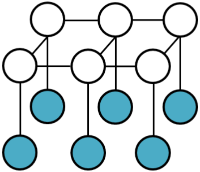
Davide Bacciu

Dipartimento di Informatica
Università di Pisa
bacciu@di.unipi.it

Machine Learning: Neural Networks and Advanced Models (AA2)

# Directed and Undirected Graphical Models





- Represent asymmetric (directed) or symmetric (undirected) relationships between random variables
- Directed Bayesian Networks decompose joint probability as a product of local conditional probabilities

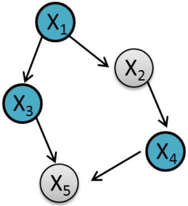
$$P(\mathbf{X}) = \prod_{i=1}^{N} P(X_i | pa(X_i))$$

- Undirected Markov Networks decompose joint probability as a product of clique potentials

$$P(\mathbf{X}) = \frac{1}{Z} \prod_{C} \psi(\mathbf{X}_C)$$

# The Inference Problem

General problem - How to infer the distribution $P(\mathbf{X}_{unk}|\mathbf{X}_{obs})$ of a number of random variables $\mathbf{X}_{unk}$ in the graphical model, given the observed values of other variables $\mathbf{X}_{obs}$




Classical inference problems

- How to query (predict with) a graphical model?
- Probability of unknown $X$ given observations $\mathbf{d}$, $P(X|\mathbf{d})$
- Determine most likely hypothesis

Inference algorithms are fundamental also to address learning in graphical models

# Complexity of Inference

Exact inference of the distribution $P(\mathbf{X}_{unk}|\mathbf{X}_{obs})$ is NP-hard for a general graphical model

- Efficient inference procedure can be defined for particular families of graphical models
  - Exact inference over chains and trees
  - Key idea: variable elimination and message passing on the structure of the graph
- Other families of graphical models require the design of efficient approximate inference algorithms
  - Variational algorithms
  - Sampling methods

# Exact Inference - A Naive Approach

Consider the simple inference problem of computing $P(X_4)$

$$X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4$$

Obtain distribution by marginalization

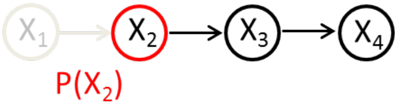
$$P(X_4) = \sum_{X_1} \sum_{X_2} \sum_{X_3} P(X_1, X_2, X_3, X_4)$$

Using the conditional independence assumptions in the model

$$P(X_4) = \sum_{X_1} \sum_{X_2} \sum_{X_3} P(X_1) P(X_2|X_1) P(X_3|X_2) P(X_4|X_3)$$

Naive approach as needs summation over an exponential number of terms

# Key Idea - Variable Elimination




What if we re-arrange the order of terms and summations?

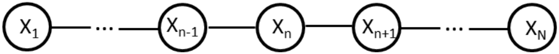
$$P(X_4) = \sum_{X_1} \sum_{X_2} \sum_{X_3} P(X_1)P(X_2|X_1)P(X_3|X_2)P(X_4|X_3)$$

$$= \sum_{X_2} \sum_{X_3} P(X_3|X_2)P(X_4|X_3) \left( \sum_{X_1} P(X_1)P(X_2|X_1) \right)$$

Note that $\sum_{X_1} P(X_1)P(X_2|X_1) = \sum_{X_1} P(X_1, X_2) = P(X_2)$ We can eliminate one variable at the time with a local cost

# Generic Inference on an Undirected Chain

Consider the general case of an undirected chain




with joint distribution

$$
\begin{aligned}
P(\mathbf{X}) &= \frac{1}{Z} \prod_{C} \psi(\mathbf{X}_C) \\
&= \frac{1}{Z} \psi(X_1, X_2)\psi(X_2, X_3) \dots \psi(X_{N-1}, X_N)
\end{aligned}
$$

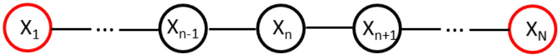
How do we infer $P(X_n)$?

$$
P(X_n) = \sum_{X_1} \cdots \sum_{X_{n-1}} \sum_{X_{n+1}} \cdots \sum_{X_N} P(\mathbf{X})
$$

We do a bit of rearrangements of sum and products to avoid exponential summations

# Step 1 - Variable Elimination

Choose a target for elimination




Bring $X_N$ summation close to the product terms that contains it

$$\sum_{X_N} \psi(X_{N-1}, X_N)$$

Same thing can be done for $X_1$

$$\sum_{X_1} \psi(X_1, X_2)$$

# Step 2 - Re-arrange Summations

Exploit the idea of variable elimination to group potentials and summations

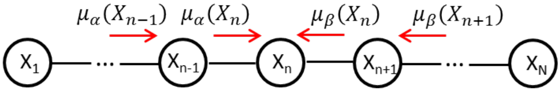
$$P(X_n) = \frac{1}{Z}$$

$$\underbrace{\left[\sum_{X_{n-1}} \psi(X_{n-1}, X_n) \dots \left[\sum_{X_2} \psi(X_2, X_3) \left[\sum_{X_1} \psi(X_1, X_2)\right]\right]\right]}_{\mu_\alpha(X_n)}$$

$$\underbrace{\left[\sum_{X_{n+1}} \psi(X_n, X_{n+1}) \dots \left[\sum_{X_N} \psi(X_{N-1}, X_N)\right]\right]}_{\mu_\beta(X_n)}$$

$$= \frac{1}{Z}\mu_\alpha(X_n)\mu_\beta(X_n)$$

# Step 3 - Message Passing




$P(x_n)$ is efficiently computed by passing local messages on the graph

- $\mu_\alpha(X_n) \rightarrow$ forward message
- $\mu_\beta(X_n) \rightarrow$ backward message

Messages are computed recursively

$$\mu_\alpha(X_n) = \sum_{X_{n-1}} \psi(X_{n-1}, X_n)\mu_\alpha(X_{n-1})$$

$$\mu_\beta(X_n) = \sum_{X_{n+1}} \psi(X_n, X_{n+1})\mu_\beta(X_{n+1})$$

# Computational complexity

Consider each $X_n$ to be a discrete RV taking $K$ values

- Local messages $\mu_\alpha(X_n)$ and $\mu_\beta(X_n)$ are $K$-dimensional vectors
- Computing the local messages is a matrix-vector multiplication with sizes $K^2$ and $K \Rightarrow O(K^2)$
- Local messages are computed for $N-1$ variables, so the total complexity is

$$O(N \cdot K^2)$$

What about the normalization term $Z$? $O(K)$

$$P(X_n) = \frac{1}{Z}\mu_\alpha(X_n)\mu_\beta(X_n) = \frac{\mu_\alpha(X_n)\mu_\beta(X_n)}{\sum_{X_n} \mu_\alpha(X_n)\mu_\beta(X_n)}$$

# Observations

- What if we want to compute $P(X_n)$ for all $n \in [1, N]$?
  - Easy because for different $n$ we will re-use the same local messages
- What if a node $X_{n'}$ is observed?
  - It only means we don't have to perform the summation $\sum_{X_{n'}}(\cdot)$ because we know the value of $X_{n'}$
- How do we compute a joint probability $P(X_n, X_{n+1})$?
  - Similar to the single node case
  - Local message passing until we reach the target nodes, which are not summed out

$$P(X_n, X_{n+1}) = \frac{1}{Z}\mu_\alpha(X_n)\psi(X_n, X_{n+1})\mu_\beta(X_{n+1})$$

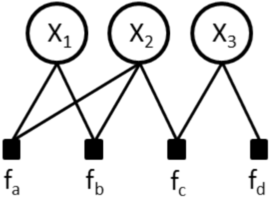
# Inference on a Tree

- How we generalize inference when the graph structure is more complex than a chain?
- Difficult problem in general, but we can solve it if the structure is a tree
- Undirected tree
  - A graph where there is exactly one path between any pair of nodes
  - No loops
- Directed tree
  - A graph where there is exactly one node with no parents while all other nodes have a single parent
  - The corresponding moral graph will be an undirected tree

First we introduce a graphical representation that makes notation easier

# Factor Graphs

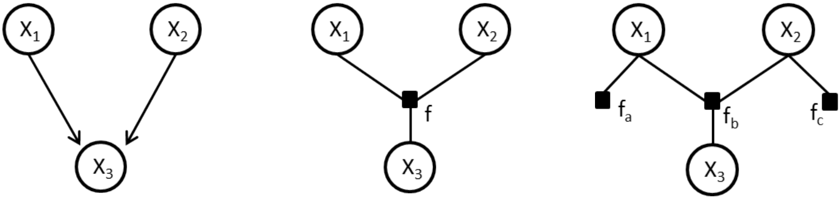
Yet another way to represent how the joint probability of a set of variables factorizes into a product of functions $f_C$ defined over subsets $C$ of the variables




$$P(X_1, X_2, X_3) = f_a(X_1, X_2) f_b(X_1, X_2) f_c(X_2, X_3) f_d(X_3)$$

- Random variables $X_n$ are circular nodes
- Factors $f_C$ are functions of the variables $X_n$ and are denoted as square nodes
- Edges connect a factor to the variables they are functions of, e.g. $f_a(X_1, X_2)$

# From Graphical Models to Factor Graphs
## Directed Models

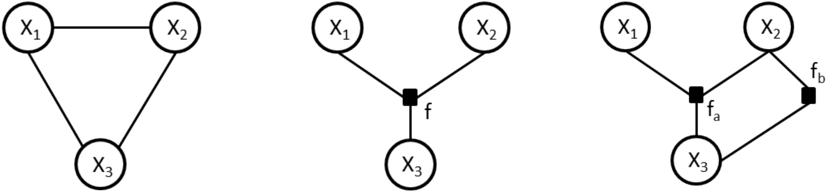



Both factor graphs represent the same distribution, but factorized differently

$$P(X_1, X_2, X_3) = f(X_1, X_2, X_3)$$

$$P(X_1, X_2, X_3) = f_a(X_1) f_b(X_1, X_2, X_3) f_c(X_3)$$

# From Graphical Models to Factor Graphs
## Undirected Models




$$\psi(X_1, X_2, X_3) = f(X_1, X_2, X_3)$$

$$\psi(X_1, X_2, X_3) = f_a(X_1, X_2, X_3) f_b(X_2, X_3)$$

Notice how the loop in the undirected model disappears in the second factor graph

# Sum-Product Algorithm

- A powerful class of efficient, exact inference algorithms for (directed/undirected) tree-structured models
- Use factor graph representation to provide a unique algorithm for directed/undirected models
- Assume all random variables are discrete and hidden
- We begin by computing the marginal $P(X)$ for one particular node $X$ of the graph

# Sum-Product Algorithm

## Overview

- The marginal for a single node $X$ is

$$P(X) = \sum_{\mathbf{X} \setminus X} P(\mathbf{X})$$

where $\mathbf{X} \setminus X$ is the set of all variables except $X$
- Factorize the joint probability $P(\mathbf{X})$ using the graph factorization

$$P(X) = \sum_{\mathbf{X} \setminus X} \prod_{s} f_s(\mathbf{X}_s)$$

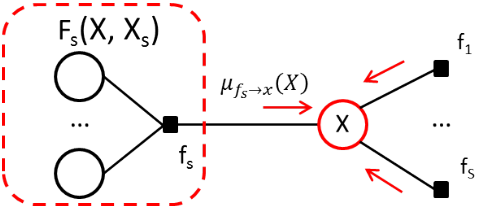
- Re-arrange sum-products to optimize computation

$$P(X) = \prod_{s} \sum_{\mathbf{X}_s \in \mathbf{X} \setminus X} f_s(\mathbf{X}_s)$$

We efficiently due this by message passing on the factor graph

# Sum-Product Algorithm

## Factor to Variable Messages (I)



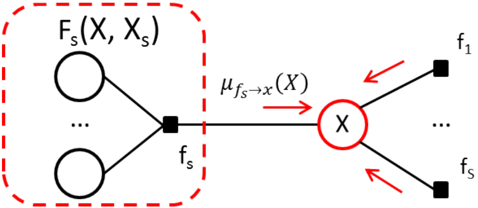

Using the structure of the factor graph locally to $X$ we can express the marginal as

$$P(X) = \prod_{s \in ne(X)} \left[ \sum_{\mathbf{X}_s} F_s(X, \mathbf{X}_s) \right]$$

- $ne(X)$ set of factor nodes $f_s$ neighbor of $X$
- $\mathbf{X}_s$ variable nodes connected to $X$ via $f_s$

# Sum-Product Algorithm

## Factor to Variable Messages (II)




Using the structure of the factor graph locally to $X$ we can express the marginal as

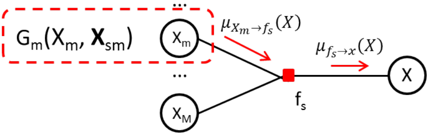
$$P(X) = \prod_{s \in ne(X)} \left[ \sum_{\mathbf{X}_s} F_s(X, \mathbf{X}_s) \right] = \prod_{s \in ne(X)} \mu_{f_s \to x}(X)$$

- $F_s(X, \mathbf{X}_s)$ product of all factors in $\mathbf{X}_s$ reaching factor $f_s$
- $\mu_{f_s \to x}(X)$ message from factor $f_s$ to variable $X$

# Sum-Product Algorithm

## Variable to Factor Messages (I)

How do we compute $F_s(X, \mathbf{X}_s)$?




$F_s$ can be factorized using the variables $X_m$ it depends on

$$F_s(X, \mathbf{X}_s) = \underbrace{f_s(X, X_1, \dots, X_M)}_{\text{local information}} \underbrace{G_1(X_1, \mathbf{X}_{s1}), \dots, G_1(X_M, \mathbf{X}_{sM})}_{\text{information from children } m}$$

$$\mu_{f_s \to x}(X) = \sum_{\mathbf{X}_s} F_s(X, \mathbf{X}_s)$$

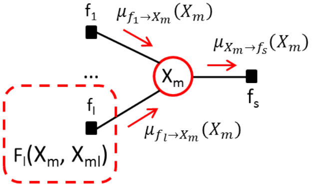
$$\mu_{f_s \to x}(X) = \sum_{X_1} \cdots \sum_{X_M} f_s(X, X_1, \dots, X_M) \prod_{m \in ne(f_s) \backslash X} \overbrace{\left[ \sum_{\mathbf{X}_{sm}} G_m(X_m, \mathbf{X}_{sm}) \right]}^{\mu_{X_m \to f_s}(X_m)}$$

# Sum-Product Algorithm

## Variable to Factor Messages (II)

$$\mu_{f_s \to X}(X) = \sum_{X_1} \cdots \sum_{X_M} f_s(X, X_1, \ldots, X_M) \prod_{m \in ne(f_s) \setminus X} \mu_{X_m \to f_s}(X_m)$$

How do we compute $G_m(X_m, \mathbf{X}_{sm})$?



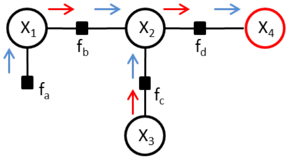

$$G_m(X_m, \mathbf{X}_{sm}) = \prod_{l \in ne(X_m) \setminus f_s} F_l(X_m, \mathbf{X}_{ml})$$

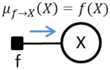
$$\mu_{X_m \to f_s}(X_m) = \prod_{l \in ne(X_m) \setminus f_s} \left[ \sum_{\mathbf{X}_{ml}} F_l(X_m, \mathbf{X}_{ml}) \right]$$

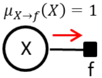
$$= \prod_{l \in ne(X_m) \setminus f_s} \mu_{f_l \to X_m}(X_m)$$

# Sum-Product Algorithm
## Summary (I)

To compute $P(X_i)$ for a given variable $X_i$ using the sum-product algorithm



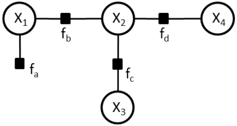



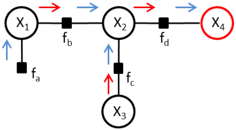

- Pick $X_i$ as the root of the sum-product recursion (i.e. destination of the messages)
- Begin computing messages at the leaves
  - Leaf = factor $\mu_{f\to X} = f(X)$
  - Leaf = variable $\mu_{X\to f} = 1$
- Recursively compute the $\mu_{f\to X}$ and $\mu_{X\to f}$ messages from the leaves until the root is reached

# Sum-Product Algorithm

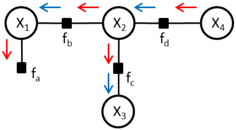
## Summary (II)

What if we want to compute $P(X)$ for all variables?




Pick a node as root and propagate messages as before




Once root has received all information, propagate messages from root to the leaves

The number of messages needed is $2 \cdot |E|$ where $|E|$ is the number of edges in the factor graph

# Observations

- The marginal $P(\mathbf{X}_s)$ of the variables $\mathbf{X}_s$ linked to a factor $f_s$ can be computed as

$$P(\mathbf{X}_s) = f_s(\mathbf{X}_s) \prod_{i \in f_s} \mu_{X_i \to f_s}(X_i)$$

- Normalization
  - If the factor graph derives from a directed model, the marginals are already normalized
  - If derives from an undirected model, we compute the un-normalized marginals $P(X)$ for each $X$ and normalize each marginal separately
- Computing marginal $P(X|\mathbf{X}_e)$ given observed variables $\mathbf{X}_e$
  - Perform sums in messages only for unobserved variables $\mathbf{X}_u$
  - Given an observed variables $X_e = x_e$ with value $x_e$, keep only the summation term corresponding to $x_e$ (set the rest to 0)

# Max-Product Inference

**Sum-Product Algorithm**

Efficiently computes a marginal distribution $P(\mathbf{X})$ from a joint distribution expressed as a factor graph

Another relevant problem in probabilistic models is to determine the most likely assignment of unobserved variables

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{X} = \mathbf{x})$$

**Max-Product Algorithm**

A class of efficient algorithms for finding the $\mathbf{x}^*$ assignment maximizing the joint distribution $P(\mathbf{X})$

We will see an example of this algorithms in Hidden Markov Models

# Exact Inference in General Graphs

- General graphs have loops which lead to messages circling forever
- Can restructure the graph to obtain a tree-like structure representing the same factorization and perform message passing on it (junction tree algorithm)
- Key idea is to triangulate the undirected graph and build a join tree whose nodes are the maximal cliques in the triangulated graph
- The junction tree is a maximum spanning tree condensing edges and nodes
    - Subset cliques are absorbed into larger ones
    - Edges are labeled by the maximum number of variables shared between cliques

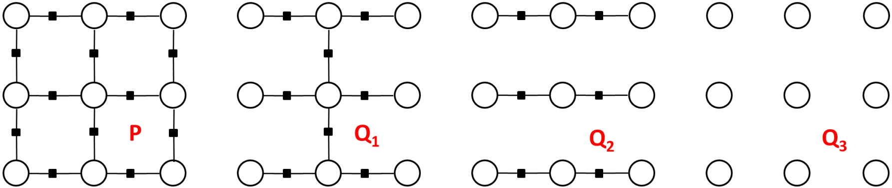
Computational complexity depends on number of variables in the largest clique (e.g. exponential for discrete variables)

# Approximate Inference in General Graphs

- Computational complexity of exact inference can become unfeasible
- Approximate inference algorithms
  - Loopy belief propagation - Treat graphs as trees using sum-product algorithm and a smart message scheduler to handle message loops
  - Variational Methods - Find an analytical approximation of the inference problem that simplifies computational complexity (relaxation of conditional independence relationships)
  - Sampling (Stochastic) Methods - Estimate the sought expectation by sampling from the probability distributions in the graph

# Variational Inference

- Loops cause problems to exact inference on original distribution $Q$
- Variational inference approximates $P$ with a distribution $Q$ corresponding to a simpler graph (tree or simpler)




How do I choose the $Q$ approximation?

- The distribution $Q$ which approximates $P$ more closely
- The simplest distribution $Q$ (less parameters)

# Measuring Goodness of Approximation

Kullback-Leibler Divergence - A measure of difference between distributions

$$DL(Q\|P) = \sum_x Q(x) \ln \frac{Q(x)}{P(x)} \text{ or } \int_{-\infty}^{+\infty} Q(x) \ln \frac{Q(x)}{P(x)} dx$$

Can use it to find the best parameters $\theta^*$ of the approximation $Q(\theta)$

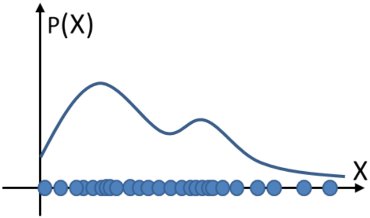
$$\theta^* = \arg\min_\theta DL(Q(\theta)\|P)$$

Variational algorithms are typically iterative

- Compute the factor functions in the approximated graph
- Estimate $Q$ from the factors
- Iterate until $Q$ does not change much between iterations

# Sampling-based Inference

Key Idea - Approximate the expectation of a complex distribution by drawing enough samples from a sufficiently simple distribution (exploiting conditional independence for efficiency)




Sample the local potentials functions/conditional probabilities instead of the joint distribution

- Gibbs sampling - Sample variables conditioned on their Markov blanket

# Take Home Messages

- Exact inference
  - Passing messages (vectors of information) on the structure of the graphical model following a propagation direction
  - A node receives messages from all predecessors (in the propagation order), applies sum-product operation and send out a compact message
- Exact inference is affordable only in certain structures
  - Sum-product and max-product on chains and trees
  - Junction tree on graphs with small cliques
- Approximate inference on general graphs
  - Variational methods: approximate distribution with a simpler one
  - Sampling methods: draw as many instances as needed to estimate the distribution expectation

# Next Lecture

- A probabilistic model for sequences: Hidden Markov Models (HMMs)
- An example of sum-product inference
- Expectation-Maximization algorithm for HMMs parameter learning
- Graphical models with varying structure: Dynamic Bayesian Networks