Introduction to Statistical Machine Learning

Christfried Webers

Statistical Machine Learning Group
NICTA
and
College of Engineering and Computer Science
The Australian National University

Canberra
February – June 2013

(Many figures from C. M. Bishop, "Pattern Recognition and Machine Learning")

Outlines

Part XV

Probabilistic Graphical Models 3

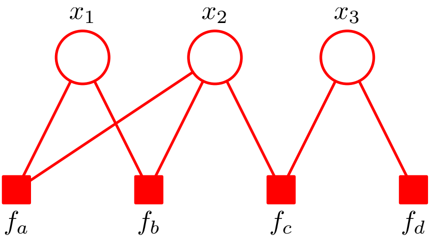
Factor Graphs

- Write $p(\mathbf{x})$ in the form of a product of factors

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

where $\mathbf{x}_s$ denotes a subset of variables.
- Example

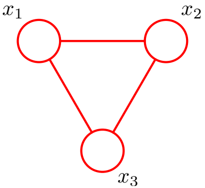
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3).$$



- More information than in MRF, because there $f_a(x_1, x_2) f_b(x_1, x_2)$ would be in one potential function.

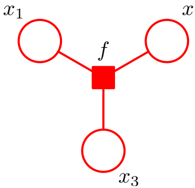
Factor Graphs - MRF example

Example of factor graphs representing the same distribution



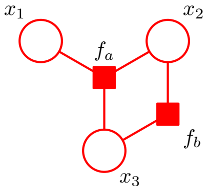
Undirected graph single clique potential

$\psi(x_1, x_2, x_3)$



Factor graph

$$f(x_1, x_2, x_3) = \psi(x_1, x_2, x_3)$$



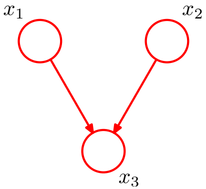
Factor graph factors satisfy

$$f_a(x_1, x_2, x_3) f_b(x_2, x_3) = \psi(x_1, x_2, x_3)$$

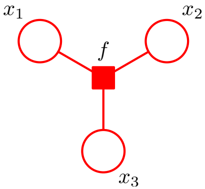
©2013
Christfried Webers
NICTA
The Australian National University

Factor Graphs - BN example

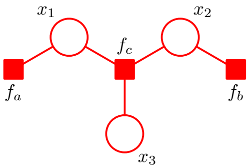
Example of factor graphs representing the same distribution



Directed graph
$p(x_1)\, p(x_2)\, p(x_3 \mid x_1, x_2)$



Factor graph
$$f(x_1, x_2, x_3) = p(x_1)\, p(x_2)\, p(x_3 \mid x_1, x_2)$$



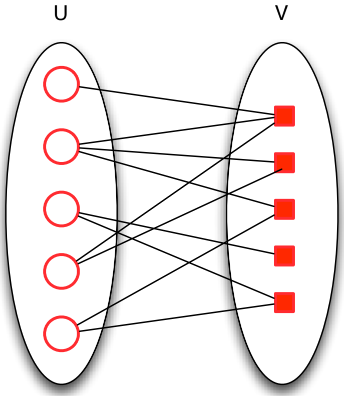
Factor graph factors satisfy
$$f_a(x_1) = p(x_1)$$
$$f_b(x_2) = p(x_2)$$
$$f_c(x_1, x_2, x_3) = p(x_3 \mid x_1, x_2)$$

Factor Graph - Bipartite Graph

- Factor Graphs are bipartite graphs.

Definition (Bipartite Graph)

A bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V.



Markov Random Field → Factor Graph

1. Create variable nodes for each node in the original graph.
2. Create factor nodes corresponding to the maximal cliques $\mathbf{x}_s$.
3. Set the factors $f_s(\mathbf{x}_s)$ to the clique potentials.

Note: There may be several different factor graphs corresponding to the same undirected graph.

Bayesian Network → Factor Graph

1. Create variable nodes for each node in the original graph.
2. Create factor nodes corresponding to the conditional distributions.
3. Add appropriate links.

Note: There may be several different factor graphs corresponding to the same directed graph.

The Sum-Product Algorithm - Overview

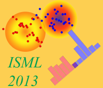
- Assume a tree-structured factor graph.
- Try to find the marginal $p(x)$ for a particular node x. (Assume here that all nodes are hidden.)

$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

- Key idea: Substitute for $p(\mathbf{x})$ using the factor graph and then interchange summations and products in order to obtain an efficient algorithm.
- Partition the factors in the joint distribution into groups, with one group associated with each factor of the nodes that is a neighbour of the variable node x.

$$p(\mathbf{x}) = \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s)$$

where $\mathrm{ne}(x)$ denotes the set of factor nodes which are neighbours of x, and X_s denotes the set of all variables in the subtree connected to the variable node x via the factor node f_s, and $F_s(x, X_s)$ represents the product of all the factors in the group associated with factor f_s.



The Sum-Product Algorithm - In Detail

- Try to find the marginal $p(x)$ for a particular node x.

$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

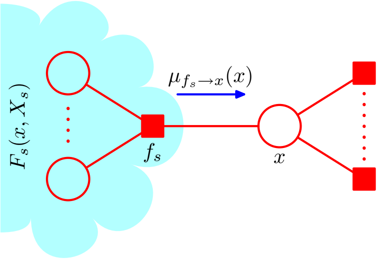
- Note, that x is an element of the set of variables $\mathbf{x}$, $x \in \mathbf{x}$.

The Sum-Product Algorithm - In Detail

- The joint distribution $p(\mathbf{x})$ can be written as a product

$$p(\mathbf{x}) = \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s)$$

- $\mathrm{ne}(x)$ denotes the set of factor nodes that are neighbours of x.
- X_s denotes the set of all variables in the subtree connected to the variable node x via the factor node.



The Sum-Product Algorithm - In Detail

- Goal: Marginal distribution $p(x)$

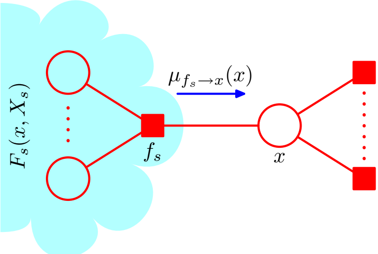
$$p(x) = \sum_{\mathbf{x}\setminus x} p(\mathbf{x})$$

- via joint distribution

$$p(\mathbf{x}) = \prod_{s\in\mathrm{ne}(x)} F_s(x, X_s)$$

- resulting in

$$p(x) = \sum_{\mathbf{x}\setminus x} \prod_{s\in\mathrm{ne}(x)} F_s(x, X_s) = \prod_{s\in\mathrm{ne}(x)} \sum_{X_s} F_s(x, X_s)$$



Example for Exchanging Sum and Product

- Goal: Marginal distribution $p(x)$

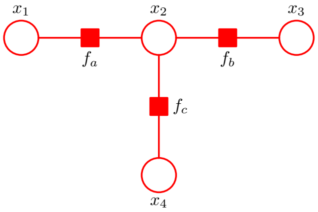
$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

$$p(x_2) = \sum_{\mathbf{x} \setminus x_2} p(\mathbf{x})$$

- via joint distribution

$$p(\mathbf{x}) = \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s)$$

$$p(x_1, x_2, x_3, x_4) = \prod_{s \in \mathrm{ne}(x_2)} F_s(x_2, X_s) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$



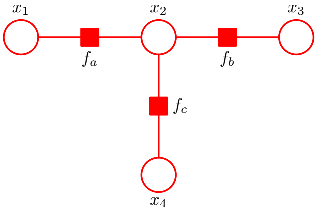
Example for Exchanging Sum and Product

- Goal: Marginal distribution $p(x)$
  (Note: Without normalisation $Z = \sum_{x_2} p(x_2)$)

$$p(x) = \sum_{\mathbf{x}\backslash x} \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s) = \prod_{s \in \mathrm{ne}(x)} \sum_{X_s} F_s(x, X_s)$$

$$p(x_2) = \sum_{x_1, x_3, x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$

$$= \left(\sum_{x_1} f_a(x_1, x_2) \right) \left(\sum_{x_3} f_b(x_2, x_3) \right) \left(\sum_{x_4} f_c(x_2, x_4) \right)$$



The Sum-Product Algorithm - In Detail

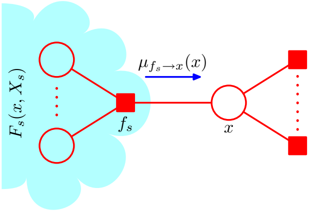
- Goal: Marginal distribution $p(x)$

$$p(x) = \sum_{\mathbf{x}\setminus x} \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s) = \prod_{s \in \mathrm{ne}(x)} \sum_{X_s} F_s(x, X_s)$$

$$= \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x)$$

- with a set of functions which can be view as messages

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s).$$

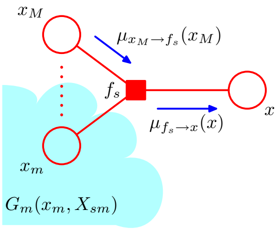


How to evaluate the messages $\mu_{f_s \to x}(x)$?

- Each factor $F_s(x, X_s)$ consists of a subgraph and can therefore be written as

$$F_s(x, X_s) = f_s(x, x_1, \dots, x_M)\, G_1(x_1, X_{s1}) \dots G_M(x_M, X_{sM})$$

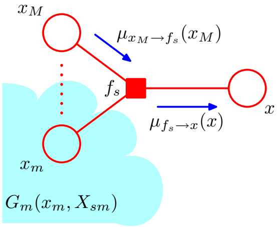
where $\mathbf{x}_s = \{x, x_1, \dots, x_M\}$ is the set of variables on which the factor f_s depends.



How to evaluate the messages $\mu_{f_s \to x}(x)$?

- Using the factorisation of $F_s(x, X_s)$, the message $\mu_{f_s \to x}(x)$ can be written as

$$\begin{aligned}
\mu_{f_s \to x}(x) &= \sum_{X_s} F_s(x, X_s) \\
&= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \left[\sum_{X_{sm}} G_m(x_m, X_{sm}) \right] \\
&= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m).
\end{aligned}$$



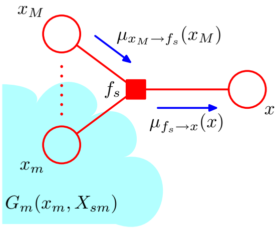
Two kind of messages

- Messages from factor nodes to variable nodes

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$$

- Messages from variable nodes to factor nodes

$$\mu_{x_m \to f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm})$$

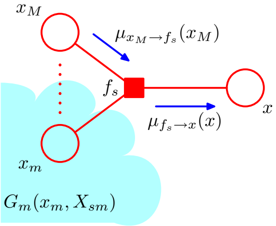


Calculating messages factor → variable nodes

- We already have a formula to calculate messages from factor nodes to variable nodes

$$\mu_{f_s \to x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

- Take the product of all incoming messages, multiply with the factor associated with the node and marginalise over all variables associated with the incoming messages.

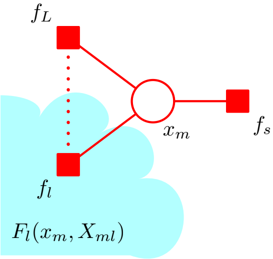


Calculating messages variable → factor nodes

- $G_m(x_m, X_{sm})$ is a product of terms $F_l(x_m, X_{ml}$

$$G_m(x_m, X_{sm}) = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml})$$

where the product is taken over all neighbours of node x_m except for node f_s.

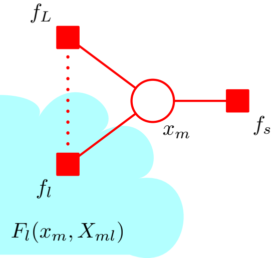


Calculating messages variable → factor nodes

- Therefore

$$\mu_{x_m \to f_s}(x_m) = = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \left[\sum_{X_{ml}} F_l(x_m, X_{ml}) \right]$$
$$= \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m).$$

- Evaluate the message sent by a variable node to an adjacent factor node by taking the product of all incoming messages.

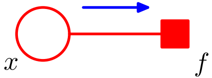


How to start? How to normalise?

- Consider node x as the root node of the factor graph.
- Start at the leaf nodes.

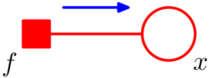
  - If the leaf node is a variable node then

    $$\mu_{x \to f}(x) = 1$$

  - If the leaf node is a factor node then

    $$\mu_{f \to x}(x) = f(x)$$

- Normalisation : Calculate $p(x)$ by message passing. Then

$$Z = \sum_x p(x)$$

Marginals for ALL variable nodes in the graph

- Brute force : Run the algorithm again for each node.
- More efficient
  1. Arbitrarily choose one root in the graph.
  2. Propagate all messages from leafs to root.
  3. Now, root got all messages from its neighbours. Calculate marginal for root.
  4. Root can now send messages to its neighbours.
  5. Calculate their marginals and continue sending messages to the neighbours closer to the leafs.
- More efficient methods needs only twice as many computation to calculate marginals for all nodes than calculating marginal for one node.

Similar Algorithms

- Max-Sum algorithm : Find the values for the variables for which the probability has a maximum.
- Junction Tree Algorithm : Exact inference in general graphs. Computational cost is determined by the number of variables in the largest clique of the graph. Grows exponentially with this number for discrete variables.
- Loopy Belief Propagation : Try Max-Sum algorithm on graphs which are NOT tree-structered. Graph has cycles and therefore information flows several times through the graph. Initialise by assuming a unit message has been sent over each link in each direction. Convergence is NOT longer guaranteed.

Christfried Webers
NICTA
The Australian National University

ISML 2013

Learning the Graph Structure

- Define a space of possible graph structures.
- Define a measure to score each of the graph structures.
- Bayesian viewpoint: Compute the posterior distribution over graph structures.
- If we have a prior $p(m)$ over graphs indexed by m, the posterior is given via Bayes' theorem as

$$p(m \,|\, \mathcal{D}) \propto p(m)\, p(\mathcal{D} \,|\, m)$$

  where $\mathcal{D}$ is the observed data set, and the model evidence $p(\mathcal{D} \,|\, m)$ provides the score for each model.
- Challenge 1: Evaluation of the model evidence involves marginalisation over the latent variables and is computationally very demanding.
- Challenge 2: The umber of different graph structures grows exponentially with the number of nodes. Need to resort to heuristics to find good candidates.