Probabilistic Inference (CO-493)

Imperial College London

Graphical Models

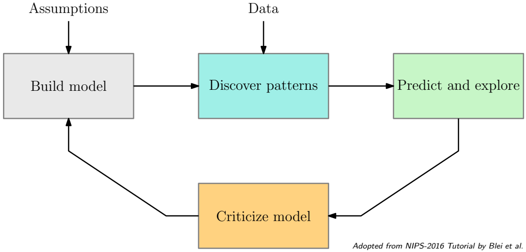
Marc Deisenroth

Department of Computing
Imperial College London

m.deisenroth@imperial.ac.uk

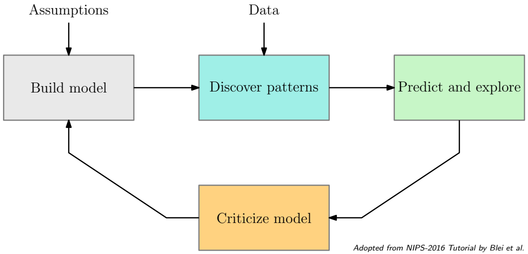
January 15, 2019

Probabilistic Pipeline



- Use knowledge and assumptions about the data to build a model
- Use model and data to discover patterns
- Predict and explore
- Criticize/revise the model

Probabilistic Pipeline



- Use knowledge and assumptions about the data to build a model
- Use model and data to discover patterns
- Predict and explore
- Criticize/revise the model

➤➤ **Inference is the key algorithmic problem**:
What does the model say about the data?

➤➤ **Goal: general and scalable approaches to inference**

Probabilistic Machine Learning

- **Probabilistic model:** Joint distribution of latent variables $\boldsymbol{z}$ and observed variables $\boldsymbol{x}$ (data):

$$p(\boldsymbol{x}, \boldsymbol{z})$$

Probabilistic Machine Learning

- **Probabilistic model:** Joint distribution of latent variables $\boldsymbol{z}$ and observed variables $\boldsymbol{x}$ (data):

$$p(\boldsymbol{x}, \boldsymbol{z})$$

- **Inference**: Learning about the unknowns $\boldsymbol{z}$ through the posterior distribution

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{x})}, \qquad p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}$$

Probabilistic Machine Learning

- **Probabilistic model:** Joint distribution of latent variables $\boldsymbol{z}$ and observed variables $\boldsymbol{x}$ (data):

$$p(\boldsymbol{x}, \boldsymbol{z})$$

- **Inference**: Learning about the unknowns $\boldsymbol{z}$ through the posterior distribution

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{x})}, \qquad p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}$$

- Normally: Denominator (marginal likelihood/evidence) intractable (i.e., we cannot compute the integral analytically)
  ⏩ Approximate inference to get the posterior

Some Options for Posterior Inference

- Exact inference (in some cases)
  - Conjugate models (see CO-496 for some examples)
  - Belief propagation and sum-product algorithm (Lauritzen & Spiegelhalter, 1988; Kschischang et al., 2001)
- Approximate inference
  - Sampling and Markov Chain Monte Carlo (to sample from the posterior)
  - Laplace approximation
  - Expectation propagation (Minka, 2001)
  - Variational inference (Jordan et al., 1999)

Graphical Models

Reading Material

Bishop: Pattern Recognition and Machine Learning, Chapter 8

Probabilistic Models

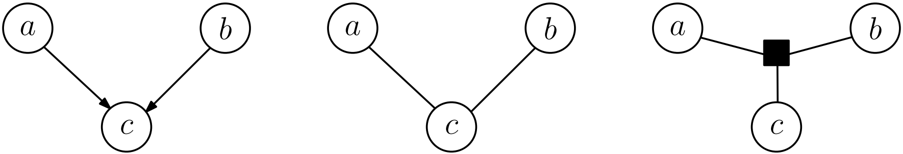
- Quantity of interest: Joint distribution $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})$ of all observed $\boldsymbol{x}$ and unobserved (latent) $\boldsymbol{z}$ random variables
  ▶▶ Probabilistic model
- Comprises information about the prior, the likelihood and the posterior

Probabilistic Models

- Quantity of interest: Joint distribution $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})$ of all observed $\boldsymbol{x}$ and unobserved (latent) $\boldsymbol{z}$ random variables
  ⏩ Probabilistic model
- Comprises information about the prior, the likelihood and the posterior
- Joint distribution $p(\boldsymbol{x}, \boldsymbol{z})$ itself can be complicated
- Does not tell us anything about structural properties of the probabilistic model (e.g., factorization, independence)

⏩ **Probabilistic graphical models**

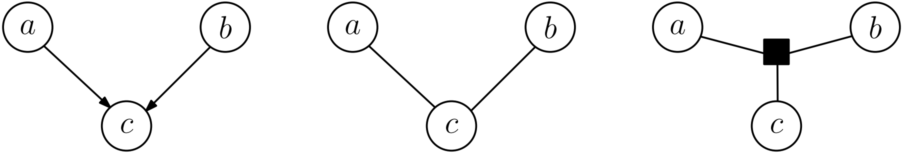
Probabilistic Graphical Models



Three types of probabilistic graphical models:

- Bayesian networks (directed graphical models)
- Markov random fields (undirected graphical models)
- Factor graphs

Probabilistic Graphical Models

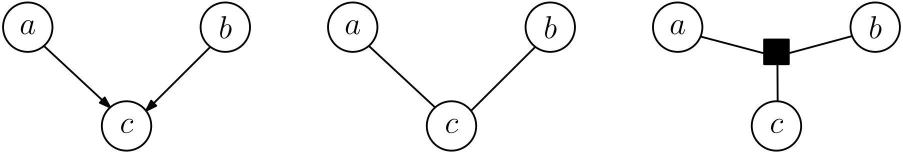


Three types of probabilistic graphical models:

- Bayesian networks (directed graphical models)
- Markov random fields (undirected graphical models)
- Factor graphs

- **Nodes:** (Sets of) random variables
- **Edges:** Probabilistic/functional relations between variables

Probabilistic Graphical Models



Three types of probabilistic graphical models:

- Bayesian networks (directed graphical models)
- Markov random fields (undirected graphical models)
- Factor graphs

- **Nodes:** (Sets of) random variables
- **Edges:** Probabilistic/functional relations between variables

▶▶ Graph captures the way in which the joint distribution over all random variables can be decomposed into a product of factors depending only on a subset of these variables

Why are they useful?

- Simple way to visualize the structure of a probabilistic model

Why are they useful?

- Simple way to visualize the structure of a probabilistic model
- Insights into properties of the model (e.g., conditional independence) by inspection of the graph

Why are they useful?

- Simple way to visualize the structure of a probabilistic model
- Insights into properties of the model (e.g., conditional independence) by inspection of the graph
- Can be used to design/motivate new models

Why are they useful?

- Simple way to visualize the structure of a probabilistic model
- Insights into properties of the model (e.g., conditional independence) by inspection of the graph
- Can be used to design/motivate new models
- Complex computations for inference and learning can be expressed in terms of graphical manipulations

Importance of Visualization

$$Pr(\{y_g, \gamma_g, t_{gk}, \beta_{gk}, l_d, f_g, z_n, i_{ng}\} | \{w_{nd}\}) = \prod_g^G p(y_g|\rho) p(\gamma_g|\sigma) p(f_g|\alpha) \cdot$$

$$[\prod_k^K p(t_{gk}|\gamma_g) p(\beta_{gk}|t_{gk}, y_g)] p(\kappa|\alpha) \prod_d^D p(l_d|\kappa) p(\pi|\alpha) \prod_n^N p(z_n|\pi)$$

$$\prod_n^N \prod_g^G p(i_{ng}|\beta, z_n) \prod_n^N \prod_d^D p(w_{nd}|i_{ng}, f, l_d)]$$

From Kim et al. (NIPS, 2015)

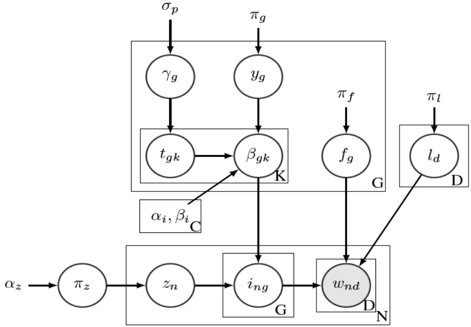
Importance of Visualization

$$Pr(\{y_g, \gamma_g, t_{gk}, \beta_{gk}, l_d, f_g, z_n, i_{ng}\}|\{w_{nd}\}) = \prod_g^G p(y_g|\rho)p(\gamma_g|\sigma)p(f_g|\alpha)\cdot$$

$$[\prod_k^K p(t_{gk}|\gamma_g)p(\beta_{gk}|t_{gk}, y_g)]p(\kappa|\alpha)\prod_d^D p(l_d|\kappa)p(\pi|\alpha)\prod_n^N p(z_n|\pi)$$

$$\prod_n^N \prod_g^G p(i_{ng}|\beta, z_n)\prod_n^N \prod_d^D p(w_{nd}|i_{ng}, f, l_d)]$$

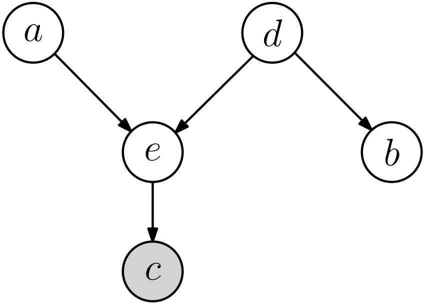
From Kim et al. (NIPS, 2015)



From Kim et al. (NIPS, 2015)

Bayesian Networks (Directed Graphical Models)

Directed Graphical Models



- Nodes: Random variables
- Shaded nodes: Observed random variables
- Unshaded nodes: Unobserved (latent) random variables
- Directed arrow from a to b: Conditional distribution $p(b|a)$.

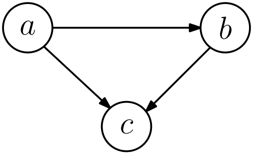
From Joints to Graphs

Consider the joint distribution

$$p(a, b, c) = p(c|a, b)p(b|a)p(a)$$

Building the corresponding graphical model:

1. Create a node for all random variables



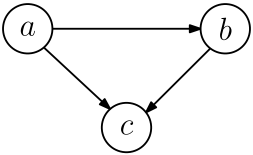
From Joints to Graphs

Consider the joint distribution

$$p(a, b, c) = p(c|a, b)p(b|a)p(a)$$

Building the corresponding graphical model:

1. Create a node for all random variables
2. For each conditional distribution, we add a directed link (arrow) to the graph from the nodes corresponding to the variables on which the distribution is conditioned on



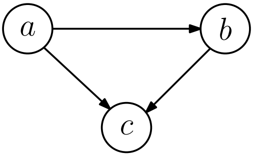
From Joints to Graphs

Consider the joint distribution

$$p(a, b, c) = p(c|a, b)p(b|a)p(a)$$

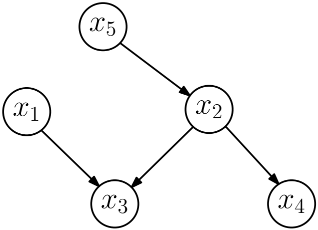
Building the corresponding graphical model:

1. Create a node for all random variables
2. For each conditional distribution, we add a directed link (arrow) to the graph from the nodes corresponding to the variables on which the distribution is conditioned on



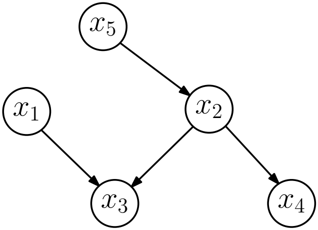
▶▶ Graph layout depends on the choice of factorization

From Graphs to Joints



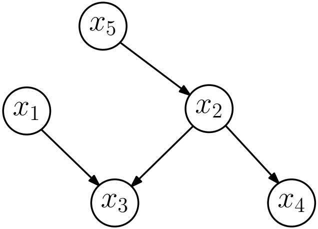
- Joint distribution is the product of a set of conditionals, one for each node in the graph

From Graphs to Joints



- Joint distribution is the product of a set of conditionals, one for each node in the graph
- Each conditional depends only on the parents of the corresponding node in the graph

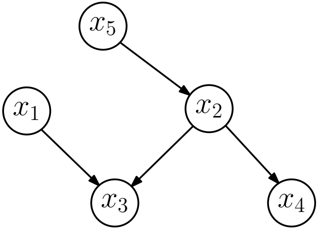
From Graphs to Joints



- Joint distribution is the product of a set of conditionals, one for each node in the graph
- Each conditional depends only on the parents of the corresponding node in the graph

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_5)p(x_2|x_5)p(x_3|x_1, x_2)p(x_4|x_2)$$

From Graphs to Joints

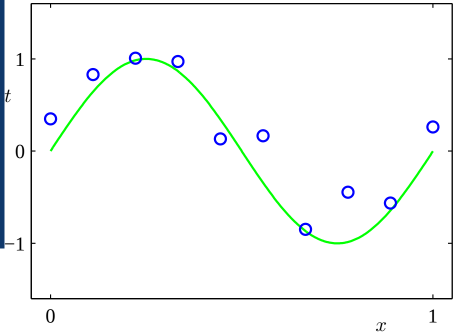


- Joint distribution is the product of a set of conditionals, one for each node in the graph
- Each conditional depends only on the parents of the corresponding node in the graph

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_5)p(x_2|x_5)p(x_3|x_1, x_2)p(x_4|x_2)$$

In general: $p(\boldsymbol{x}) = p(x_1, \ldots, x_K) = \prod_{k=1}^{K} p(x_k|\text{parents}(x_k))$

Graphical Model for (Bayesian) Linear Regression



From PRML (Bishop, 2006)

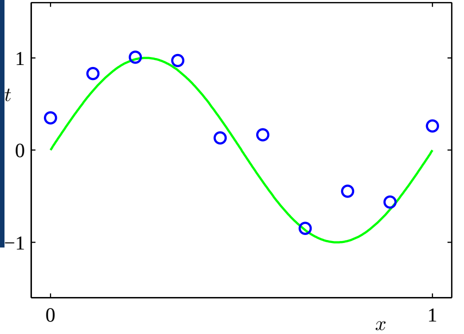
We are given a data set $(x_1, y_1), \dots, (x_N, y_N)$ where

$$y_i = f(x_i) + \varepsilon, \quad \varepsilon \sim \mathcal{N}\big(0,\, \sigma^2\big)$$

with f unknown.

▶▶ Find a (regression) model that explains the data

Graphical Model for (Bayesian) Linear Regression



From PRML (Bishop, 2006)

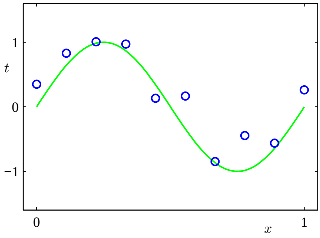
We are given a data set $(x_1, y_1), \ldots, (x_N, y_N)$ where

$$y_i = f(x_i) + \varepsilon, \quad \varepsilon \sim \mathcal{N}\big(0, \sigma^2\big)$$

with f unknown.
▶▶ Find a (regression) model that explains the data

- Consider polynomials $f(x) = \sum_{j=0}^{M} w_j x^j$ with parameters $\boldsymbol{w} = [w_0, \ldots, w_M]^\top$.
- Bayesian linear regression: Place a conjugate Gaussian prior on the parameters: $p(\boldsymbol{w}) = \mathcal{N}\big(\boldsymbol{0}, \alpha^2 \boldsymbol{I}\big)$

Graphical Model for Linear Regression

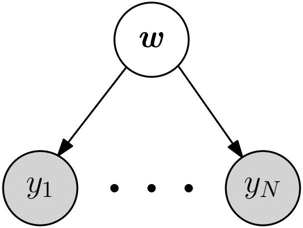


From PRML (Bishop, 2006)

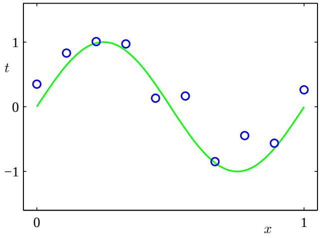
$$p(y|x) = \mathcal{N}\big(y \,|\, f(x),\, \sigma^2\big)$$

$$f(x) = \sum_{j=0}^{M} w_j x^j$$

$$p(\boldsymbol{w}) = \mathcal{N}\big(\boldsymbol{0},\, \alpha^2 \boldsymbol{I}\big)$$



Graphical Model for Linear Regression

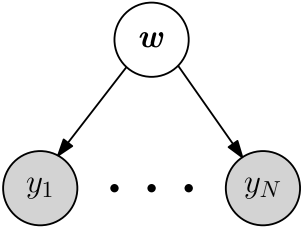


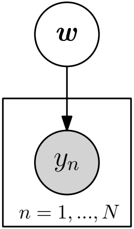
From PRML (Bishop, 2006)

$$p(y|x) = \mathcal{N}\big(y \,|\, f(x),\, \sigma^2\big)$$

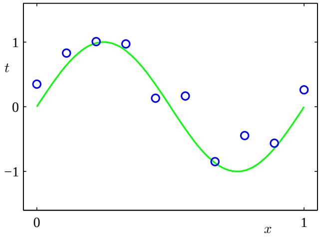
$$f(x) = \sum_{j=0}^{M} w_j x^j$$

$$p(\boldsymbol{w}) = \mathcal{N}\big(\boldsymbol{0},\, \alpha^2 \boldsymbol{I}\big)$$





Graphical Model for Linear Regression

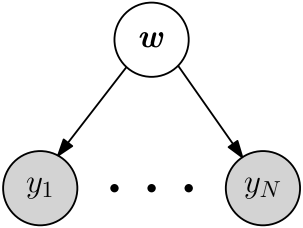


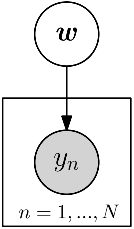
From PRML (Bishop, 2006)

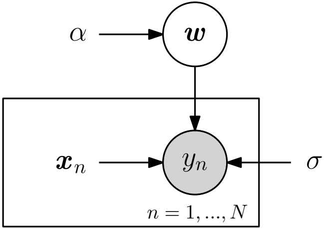
$$p(y|x) = \mathcal{N}\big(y \,|\, f(x),\, \sigma^2\big)$$

$$f(x) = \sum_{j=0}^{M} w_j x^j$$

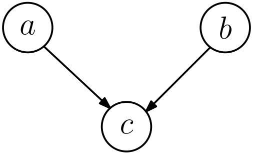
$$p(\boldsymbol{w}) = \mathcal{N}\big(\boldsymbol{0},\, \alpha^2 \boldsymbol{I}\big)$$







Conditional Independence

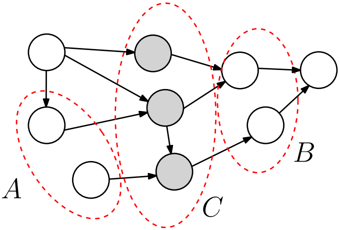


$$a \perp\!\!\!\perp b|c \iff p(a|b,c) = p(a|c)$$
$$\iff p(a,b|c) = p(a|c)p(b|c)$$

- (Conditional) independence allows for a factorization of the joint distribution ▸▸ More efficient inference
- Conditional independence properties of the joint distribution can be read directly from the graph
- No analytical manipulations required.

▸▸ **d-separation** (Pearl, 1988)

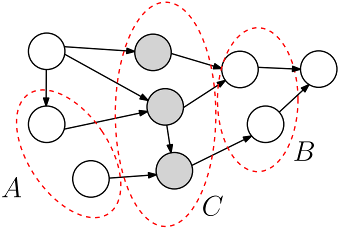
D-Separation (Directed Graphs)



Directed, acyclic graph in which A, B, C are arbitrary, non-intersecting sets of nodes. Does $A \perp\!\!\!\perp B | C$ hold?

Note: C is observed if we condition on it (and the nodes in the GM are shaded)

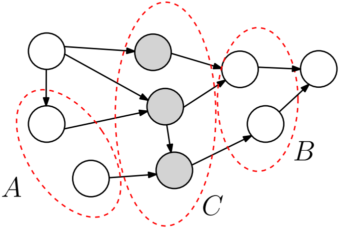
D-Separation (Directed Graphs)



Directed, acyclic graph in which A, B, C are arbitrary, non-intersecting sets of nodes. Does $A \perp\!\!\!\perp B | C$ hold?
Note: C is observed if we condition on it (and the nodes in the GM are shaded)

- Consider all possible paths from any node in A to any node in B.

D-Separation (Directed Graphs)

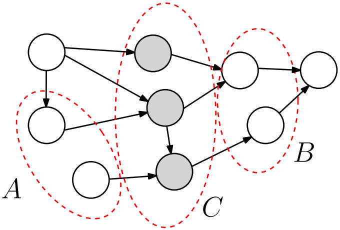


Directed, acyclic graph in which A, B, C are arbitrary, non-intersecting sets of nodes. Does $A \perp\!\!\!\perp B|C$ hold?
Note: C is observed if we condition on it (and the nodes in the GM are shaded)

▶▶ Consider all possible paths from any node in A to any node in B.
Any such **path is blocked** if it includes a node such that either

- Arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C or

D-Separation (Directed Graphs)

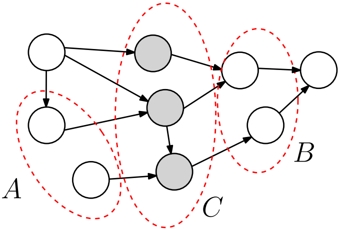


Directed, acyclic graph in which A, B, C are arbitrary, non-intersecting sets of nodes. Does $A \perp\!\!\!\perp B|C$ hold?
Note: C is observed if we condition on it (and the nodes in the GM are shaded)

▶▶ Consider all possible paths from any node in A to any node in B.
Any such **path is blocked** if it includes a node such that either

- Arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C or
- Arrows meet head-to-head at the node and neither the node nor any of its descendants is in the set C

D-Separation (Directed Graphs)



Directed, acyclic graph in which A, B, C are arbitrary, non-intersecting sets of nodes. Does $A \perp\!\!\!\perp B|C$ hold?
Note: C is observed if we condition on it (and the nodes in the GM are shaded)

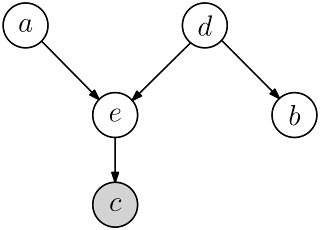
▸▸ Consider all possible paths from any node in A to any node in B.

Any such **path is blocked** if it includes a node such that either

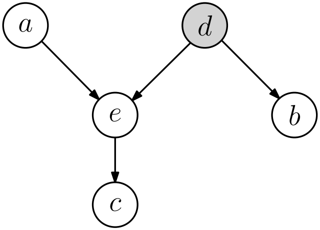
- Arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C or
- Arrows meet head-to-head at the node and neither the node nor any of its descendants is in the set C

If all paths are blocked, then A is d-separated (conditionally indep.) from B by C, and the joint distribution satisfies $A \perp\!\!\!\perp B|C$.

Example



(a) $a \perp\!\!\!\perp b | c$?



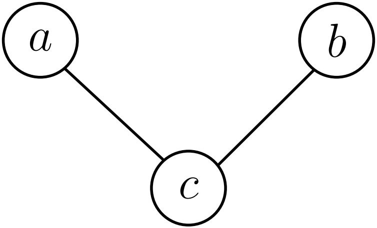
(b) $a \perp\!\!\!\perp b | d$?

A path is **blocked** if it includes a node such that either

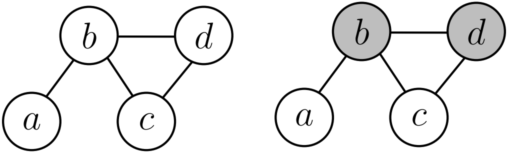
- The arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C (observed) or
- The arrows meet head-to-head at the node, and neither the node nor any of its descendants is in the set C (observed)

Markov Random Fields (Undirected Graphical Models)

Markov Random Fields

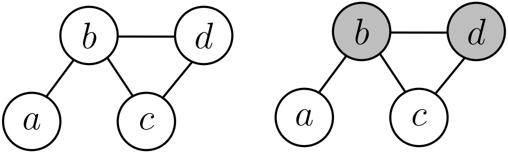


Joint Distribution



- Express joint distribution $p(x_1, \ldots, x_n) =: p(\boldsymbol{x})$ as a product of functions defined on subsets of variables that are local to the graph

Joint Distribution



- Express joint distribution $p(x_1, \ldots, x_n) =: p(\boldsymbol{x})$ as a product of functions defined on subsets of variables that are local to the graph
- If x_i, x_j are not connected directly by a link then $x_i \perp\!\!\!\perp x_j | \boldsymbol{x} \backslash \{x_i, x_j\}$ (conditionally independent given everything else)

Factorization of the Joint Distribution

- If $x_i \perp\!\!\!\perp x_j | \boldsymbol{x} \backslash \{x_i, x_j\}$ then x_i, x_j never appear in a common factor in the factorization of the joint

  ⏩ Joint distribution as a product of cliques (fully connected subgraphs)

Factorization of the Joint Distribution

- If $x_i \perp\!\!\!\perp x_j | \boldsymbol{x} \backslash \{x_i, x_j\}$ then x_i, x_j never appear in a common factor in the factorization of the joint

  ⏩ Joint distribution as a product of cliques (fully connected subgraphs)
- Define factors in the decomposition of the joint to be functions of the variables in (maximum) cliques:

$$p(\boldsymbol{x}) \propto \prod_C \psi_C(\boldsymbol{x}_C)$$

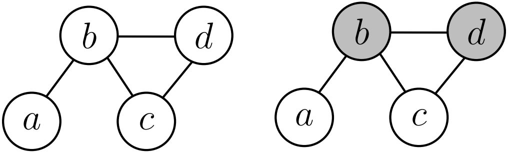
Factorization of the Joint Distribution

- If $x_i \perp\!\!\!\perp x_j | \boldsymbol{x} \backslash \{x_i, x_j\}$ then x_i, x_j never appear in a common factor in the factorization of the joint

  ⏩ Joint distribution as a product of cliques (fully connected subgraphs)
- Define factors in the decomposition of the joint to be functions of the variables in (maximum) cliques:

$$p(\boldsymbol{x}) \propto \prod_C \psi_C(\boldsymbol{x}_C)$$

Example: $p(a, b, c, d) \propto \psi_1(a, b)\psi_2(b, c, d)$



Factorization of the Joint Distribution

More generally:

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_C \psi_C(\boldsymbol{x}_C)$$

- C: maximal clique
- $\boldsymbol{x}_C$: all variables in this clique
- $\psi_C(\boldsymbol{x}_C)$: clique potential
- $Z = \sum_x \prod_C \psi_C(\boldsymbol{x}_C)$: normalization constant

Clique Potentials

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_C \psi_C(\boldsymbol{x}_C)$$

Clique potentials $\psi_C(\boldsymbol{x}_C)$:

- $\psi_C(\boldsymbol{x}_C) \geqslant 0$

Clique Potentials

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_C \psi_C(\boldsymbol{x}_C)$$

Clique potentials $\psi_C(\boldsymbol{x}_C)$:

- $\psi_C(\boldsymbol{x}_C) \geqslant 0$
- Unlike directed graphs, no probabilistic interpretation necessary (e.g., marginal or conditional)
  ▶▶ Greater flexibility but computational challenges

Clique Potentials

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_C \psi_C(\boldsymbol{x}_C)$$

Clique potentials $\psi_C(\boldsymbol{x}_C)$:

- $\psi_C(\boldsymbol{x}_C) \geqslant 0$
- Unlike directed graphs, no probabilistic interpretation necessary (e.g., marginal or conditional)
  ⏩ Greater flexibility but computational challenges
- If we convert a directed graph into an MRF, the clique potentials do have a probabilistic interpretation

Normalization Constant

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C} \psi_C(\boldsymbol{x}_C)$$

- Flexibility in the definition the factorization in an MRF
- Normalization constant (also: partition function) Z is required for parameter learning (not covered in here) and model selection

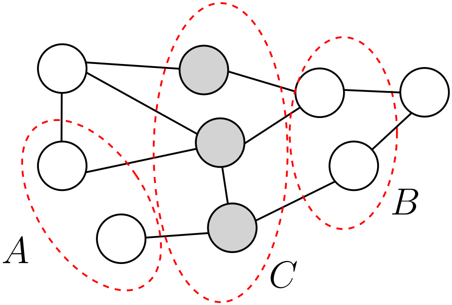
Normalization Constant

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_C \psi_C(\boldsymbol{x}_C)$$

- Flexibility in the definition the factorization in an MRF
- Normalization constant (also: partition function) Z is required for parameter learning (not covered in here) and model selection
- In a discrete model with M discrete nodes each having K states, the evaluation Z requires summing over K^M states
  ▶▶ Exponential in the size of the model
- In a continuous model, we need to solve integrals
  ▶▶ Intractable in many cases

▶▶ Major limitation of MRFs

Conditional Independence



Two easy checks for conditional independence:

- $A \perp\!\!\!\perp B|C$ if and only if all paths from A to B pass through C. (Then, all paths are blocked)
- Alternative: Remove all nodes in C from the graph. If there is a path from A to B then $A \perp\!\!\!\perp B|C$ does not hold

Potentials as Energy Functions

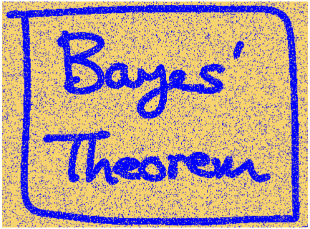
- Look only at potential functions with $\psi_C(\boldsymbol{x}_C) > 0$
  - ⏩ $\psi_C(\boldsymbol{x}_C) = \exp(-E(\boldsymbol{x}_C))$ for some energy function E

Potentials as Energy Functions

- Look only at potential functions with $\psi_C(\boldsymbol{x}_C) > 0$
  ⏩ $\psi_C(\boldsymbol{x}_C) = \exp(-E(\boldsymbol{x}_C))$ for some energy function E
- Joint distribution is the product of clique potentials
  ⏩ Total energy is the sum of the energies of the clique potentials

$$-\log p(\boldsymbol{x}) = -\log \prod_C \underbrace{\exp(-E(\boldsymbol{x}_C))}_{=\psi_C(\boldsymbol{x}_C)} = \sum_C E(\boldsymbol{x}_C)$$

Example: Image De-Noising

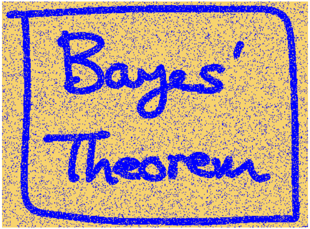


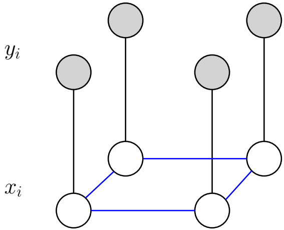
From PRML (Bishop, 2006)

- Binary image, corrupted by 10% binary noise (pixel values flip with probability 0.1).
- Objective: Restore noise-free image

▶▶ Pairwise MRF that has all its variables joined in cliques of size 2

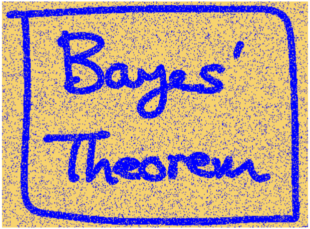
Example: Image De-Noising (2)

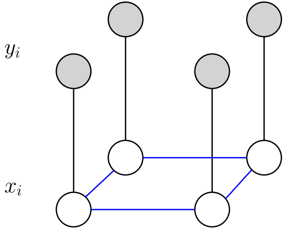




- MRF-based approach
- Latent variables $x_i \in \{-1, +1\}$ are the binary noise-free pixel values that we wish to recover

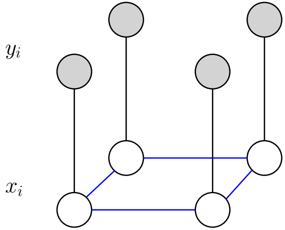
Example: Image De-Noising (2)





- MRF-based approach
- Latent variables $x_i \in \{-1, +1\}$ are the binary noise-free pixel values that we wish to recover
- Observed variables $y_i \in \{-1, +1\}$ are the noise-corrupted pixel values

Clique Potentials

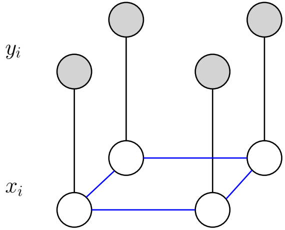


Two types of clique potentials:

- $-\log \psi_{xy}(x_i, y_i) = E(x_i, y_i) = -\eta x_i y_i\,, \quad \eta > 0$

  ▶▶ Strong correlation between observed and latent variables

Clique Potentials



Two types of clique potentials:

- $-\log \psi_{xy}(x_i, y_i) = E(x_i, y_i) = -\eta x_i y_i \,, \quad \eta > 0$

  ▶▶ Strong correlation between observed and latent variables

- $-\log \psi_{xx}(x_i, x_j) = E(x_i, x_j) = -\beta x_i x_j \,, \quad \beta > 0$

  for neighboring pixels x_i, x_j

  ▶▶ Favor similar labels for neighboring pixels (smoothness prior)

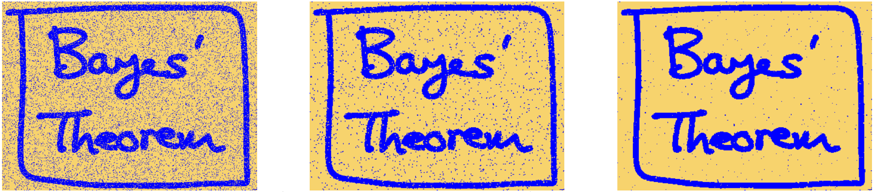
Energy Function

Total energy:

$$E(\boldsymbol{x}, \boldsymbol{y}) = \underbrace{-\eta \sum_i x_i y_i}_{\text{latent-observed}} \underbrace{-\beta \sum_{\{i,j\}} x_i x_j}_{\text{latent-latent}} + \underbrace{\gamma \sum_i x_i}_{\text{bias}}$$

- Bias term places a prior on the latent pixel values, e.g., $+1$.
- Joint distribution $p(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{Z} \exp(-E(\boldsymbol{x}, \boldsymbol{y}))$
- Fix y-values to the observed ones ⏩ Implicitly define $p(\boldsymbol{x}|\boldsymbol{y})$
- Example of an Ising model ⏩ Statistical physics

ICM Algorithm for Image De-Noising



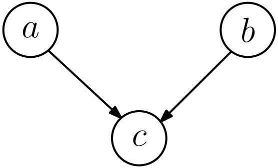
Noise-corrupted image, ICM, Graph-cut (From PRML (Bishop, 2006))

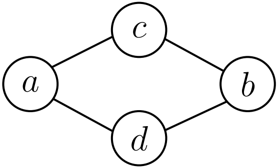
Iterated Conditional Modes (ICM, Kittler & Föglein, 1984)

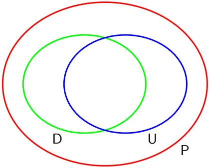
1. Initialize all $x_i = y_i$
2. Pick any x_j: Evaluate total energy
   $E(\boldsymbol{x}^{\setminus j} \cup \{+1\}, \boldsymbol{y}), \quad E(\boldsymbol{x}^{\setminus j} \cup \{-1\}, \boldsymbol{y})$
3. Set x_j to whichever state (± 1) has the lower energy
4. Repeat

▶▶ Local optimum

Relation to Directed Graphs







- Directed and undirected graphs express different conditional independence properties
- Left: $a \perp\!\!\!\perp b|\emptyset,\ a \not\!\perp\!\!\!\perp b|c$ has no MRF equivalent
- Center: $a \not\!\perp\!\!\!\perp b|\emptyset,\ c \perp\!\!\!\perp d|a \cup b,\ a \perp\!\!\!\perp b|c \cup d$ has no Bayesnet equivalent

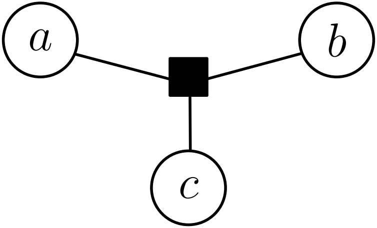
Factor Graphs

Good references:

Kschischang et al.: Factor Graphs and the Sum-Product Algorithm. IEEE Transactions on Information Theory (2001)

Loeliger: An Introduction to Factor Graphs. IEEE Signal Processing Magazine, (2004)

Factor Graphs



- (Un)directed graphical models express a global function of several variables as a product of factors over subsets of those variables
- Factor graphs make this decomposition explicit by introducing additional nodes for the factors themselves

Factorizing the Joint

The joint distribution is a product of factors:

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

- $\boldsymbol{x} = (x_1, \ldots, x_n)$
- $\boldsymbol{x}_s$: Subset of variables
- f_s: Factor; non-negative function of the variables $\boldsymbol{x}_s$

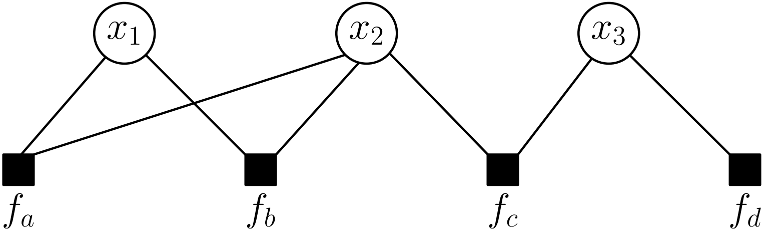
Factorizing the Joint

The joint distribution is a product of factors:

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

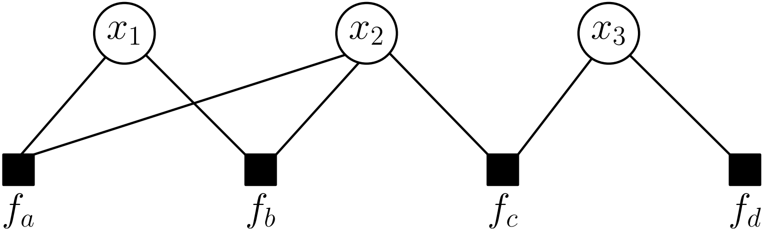
- $\boldsymbol{x} = (x_1, \dots, x_n)$
- $\boldsymbol{x}_s$: Subset of variables
- f_s: Factor; non-negative function of the variables $\boldsymbol{x}_s$
- Building a factor graph as a bipartite graph:
  - Nodes for all random variables (same as in (un)directed graphical models)
  - Additional nodes for factors (black squares) in the joint distribution
- Undirected links connecting each factor node to all of the variable nodes the factor depends on

Example



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

Example

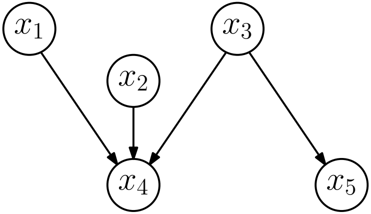


$$p(\boldsymbol{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

▶▶ Efficient inference algorithms for factor graphs (e.g., sum-product algorithm)

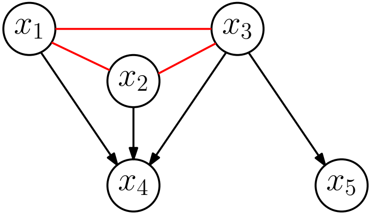
Converting Graphs

Directed Graph → MRF



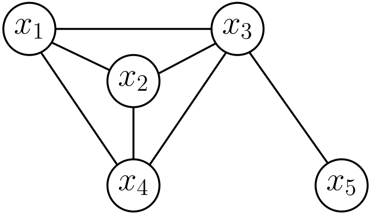
1. Moralization:

Directed Graph → MRF



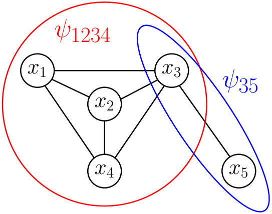
1. Moralization:
   - Add additional undirected links between all pairs of parents for each node in the graph

Directed Graph → MRF



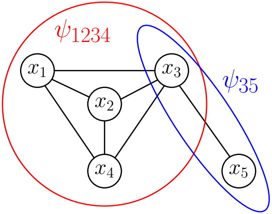
1. Moralization:
   - Add additional undirected links between all pairs of parents for each node in the graph
   - Drop arrows on original links

Directed Graph → MRF



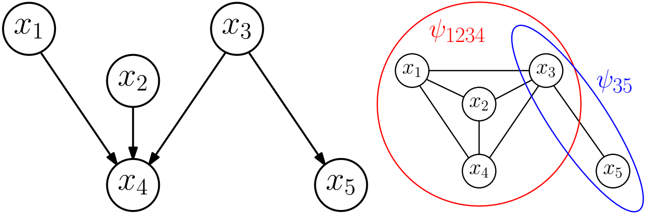
1. Moralization:
   - Add additional undirected links between all pairs of parents for each node in the graph
   - Drop arrows on original links
2. Identify (maximum) cliques

Directed Graph → MRF



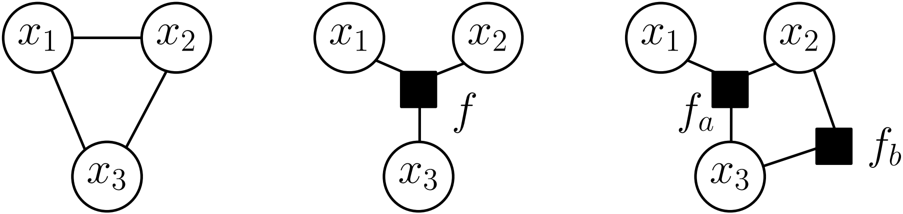
1. Moralization:
   - Add additional undirected links between all pairs of parents for each node in the graph
   - Drop arrows on original links
2. Identify (maximum) cliques
3. Initialize all clique potentials to 1

Directed Graph → MRF



1. Moralization:
   - Add additional undirected links between all pairs of parents for each node in the graph
   - Drop arrows on original links
2. Identify (maximum) cliques
3. Initialize all clique potentials to 1
4. Take each conditional distribution factor in the directed graph, multiply it into one of the clique potentials

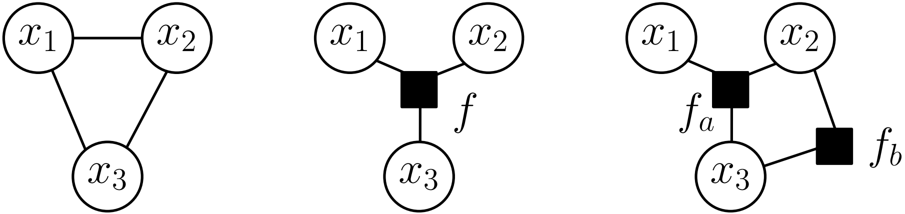
MRF → Factor Graph



1. Take variable nodes from MRF
2. Create additional factor nodes corresponding to the maximal cliques $\boldsymbol{x}_s$
3. The factors $f_s(\boldsymbol{x}_s)$ equal the clique potentials
4. Add appropriate links

Multiple factor graphs may correspond to the same undirected graph

Example: MRF → Factor Graph



Multiple factor graphs may correspond to the same undirected graph

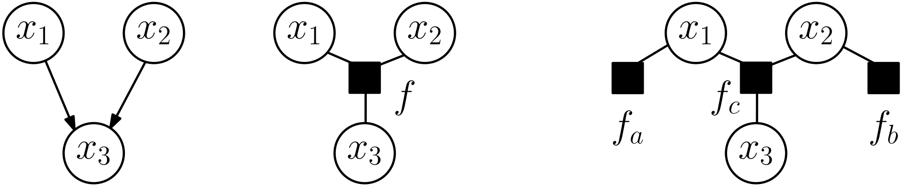
- MRF with clique potential $\psi(x_1, x_2, x_3)$
- Factor graph with factor $f(x_1, x_2, x_3) = \psi(x_1, x_2, x_3)$
- Factor graph with factors, such that $f_a(x_1, x_2, x_3) f_b(x_2, x_3) = \psi(x_1, x_2, x_3)$

Directed Graphical Model → Factor Graph

1. Take variable nodes from Bayesian network
2. Create additional factor nodes corresponding to the conditional distributions
3. Add appropriate links

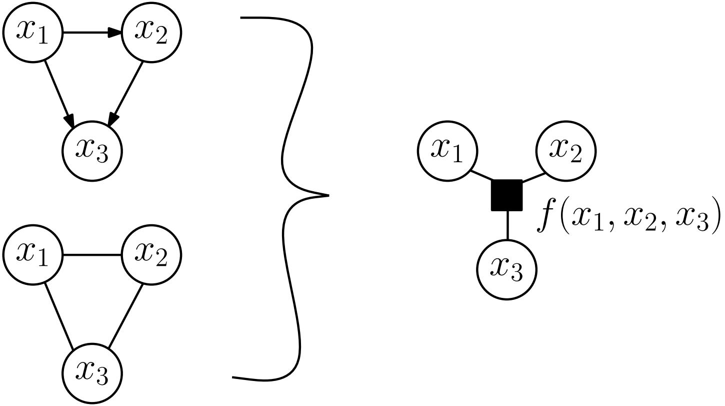
Not unique

Example: Directed Graph → Factor Graph



- Directed graph with factorization $p(x_1)p(x_2)p(x_3|x_1, x_2)$
- Factor graph with factor $f(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$
- Factor graph with factors $f_a = p(x_1),\ f_b = p(x_2),\ f_c = p(x_3|x_1, x_2)$

Removing Cycles



- Local cycles in an (un)directed graph (due to links connecting parents of a node) can be removed on conversion to a factor graph

Exact Inference in Factor Graphs

Sum-Product Algorithm for Factor Graphs

- Factor graphs give a uniform treatment to message passing, which is used for inference in graphs
- Inference: Find (marginal) posterior distributions

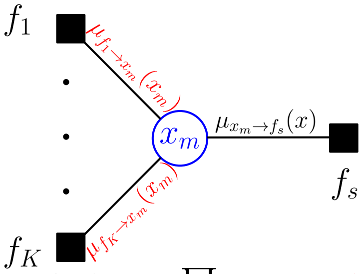
Sum-Product Algorithm for Factor Graphs

- Factor graphs give a uniform treatment to message passing, which is used for inference in graphs
- Inference: Find (marginal) posterior distributions
- Idea: Local message passing between nodes and factors
- Two different types of messages:
  - Messages $\mu_{x\to f}(x)$ from variable nodes to factors
  - Messages $\mu_{f\to x}(x)$ from factors to variable nodes

Sum-Product Algorithm for Factor Graphs

- Factor graphs give a uniform treatment to message passing, which is used for inference in graphs
- Inference: Find (marginal) posterior distributions
- Idea: Local message passing between nodes and factors
- Two different types of messages:
  - Messages $\mu_{x\to f}(x)$ from variable nodes to factors
  - Messages $\mu_{f\to x}(x)$ from factors to variable nodes
- Repeated sending of these messages through the graph converges
- Factors transform messages into evidence for the receiving node

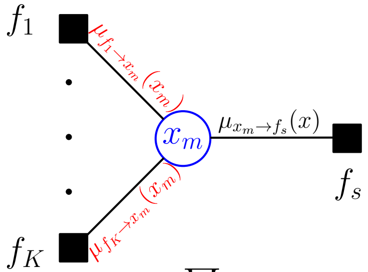
Variable-to-Factor Message



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

- Take the product of all incoming messages along all other links

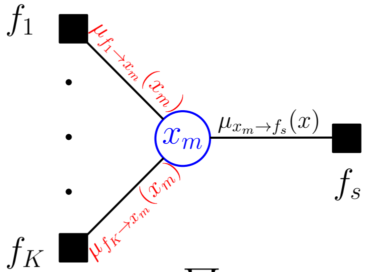
Variable-to-Factor Message



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

- Take the product of all incoming messages along all other links
- A variable node can send a message to a factor node once it has received messages from all other neighboring factors

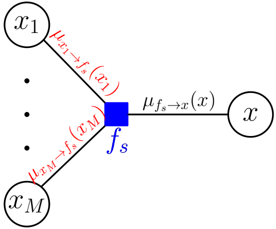
Variable-to-Factor Message



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

- Take the product of all incoming messages along all other links
- A variable node can send a message to a factor node once it has received messages from all other neighboring factors
- The message that a node sends to a factor is made up of the messages that it receives from all other factors.

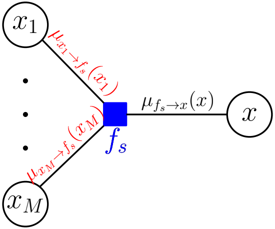
Factor-to-Variable Message



$$\mu_{f_s \to x}(x) = \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

- Take the product of the incoming messages along all other links coming into the factor node

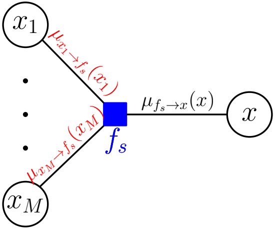
Factor-to-Variable Message



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

- Take the product of the incoming messages along all other links coming into the factor node
- Multiply by the factor associated with that node

Factor-to-Variable Message



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

- Take the product of the incoming messages along all other links coming into the factor node
- Multiply by the factor associated with that node
- Marginalize over all variables associated with the incoming messages

Initialization

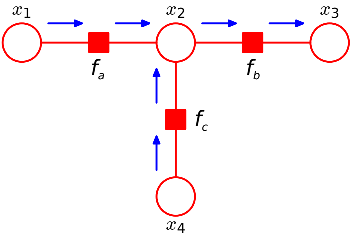
- If the leaf node is a variable node, initialize the corresponding messages to 1:

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

- If the leaf node is a factor node, the message should be

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

Example (1)



From PRML (Bishop, 2006)

$$\mu_{x_1 \to f_a}(x_1) = 1$$

$$\mu_{f_a \to x_2}(x_2) = \sum_{x_1} f_a(x_1, x_2) \cdot 1$$

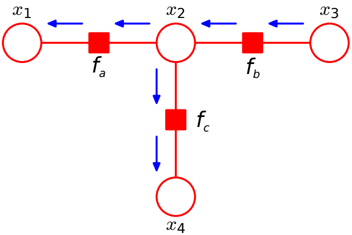
$$\mu_{x_4 \to f_c}(x_4) = 1$$

$$\mu_{f_c \to x_2}(x_2) = \sum_{x_4} f_c(x_2, x_4) \cdot 1$$

$$\mu_{x_2 \to f_b}(x_2) = \mu_{f_a \to x_2}(x_2)\mu_{f_c \to x_2}(x_2)$$

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

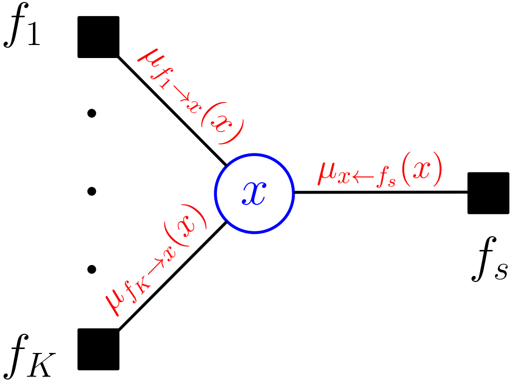
Example (2)



From PRML (Bishop, 2006)

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

Marginals



For a single variable node the marginal is given as the product of all incoming messages:

$$p(x) = \prod_{f_i \in \text{ne}(x)} \mu_{f_i \to x}(x)$$

Observed Variables ⏩ Posterior

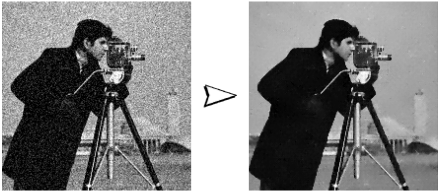
- Thus far, we have focused on the case where all variables are unobserved.
- Posterior is always conditioned on observations
- Partition $\boldsymbol{x} = \boldsymbol{h} \cup \boldsymbol{v}$, $\boldsymbol{h}$: hidden variables, $\boldsymbol{v}$: visible variables with observations $\hat{\boldsymbol{v}}$
- $p(\boldsymbol{v} = \hat{\boldsymbol{v}}) = \prod_i I(v_i = \hat{v}_i)$
- $p(\boldsymbol{x})p(\boldsymbol{v} = \hat{\boldsymbol{v}}) = p(\boldsymbol{h}, \boldsymbol{v} = \hat{\boldsymbol{v}}) \propto p(\boldsymbol{h}|\boldsymbol{v} = \hat{\boldsymbol{v}})$
- Marginal posteriors $p(h_i|\boldsymbol{v} = \hat{\boldsymbol{v}})$ can be obtained via sum-product algorithm and some local computations

  ⏩ (Koller & Friedman, 2009)

Exact Inference in (Un)Directed Graphical Models

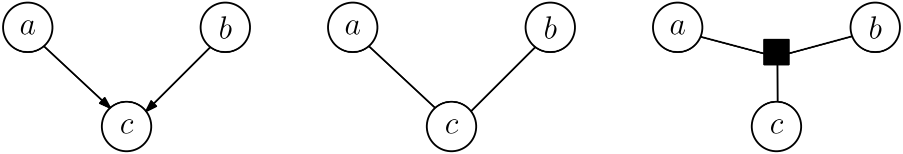
- Loops are possible ⏩ **Junction Tree Algorithm** (Lauritzen & Spiegelhalter, 1988)
- Alternative: **Loopy Belief Propagation** (Frey & MacKay 1998)

Applications of Inference in Graphical Models



- Ranking: TrueSkill (Herbrich et al., 2007)
- Computer vision: de-noising, segmentation, semantic labeling, ... (e.g., Sucar & Gillies, 1994; Shotton et al., 2006; Szeliski et al., 2008)
- Coding theory: Low-density parity-check codes, turbo codes, ... (e.g., McEliece et al., 1998)
- Linear algebra: Solve linear equation systems (Shental et al., 2008)
- Signal processing: Iterative state estimation (e.g., Bickson et al., 2007; Deisenroth & Mohamed, 2012)

Summary



- Three types of graphical models: directed, undirected, factor graphs
- Conditional independence
- Sum-product algorithm for exact inference in factor graphs

References I

[1] D. Bickson, D. Dolev, O. Shental, P. H. Siegel, and J. K. Wolf. Linear Detection via Belief Propagation. In *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*, 2007.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag, 2006.

[3] M. P. Deisenroth and S. Mohamed. Expectation Propagation in Gaussian Process Dynamical Systems. In *Advances in Neural Information Processing Systems*, pages 2618–2626, 2012.

[4] B. J. Frey and D. J. C. MacKay. A Revolution: Belief Propagation in Graphs with Cycles. In *Advances in Neural Information Processing Systems*, 1998.

[5] R. Herbrich, T. Minka, and T. Graepel. TrueSkill(TM): A Bayesian Skill Rating System. In *Advances in Neural Information Processing Systems*, pages 569–576. MIT Press, 2007.

[6] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37:183–233, 1999.

[7] B. Kim, J. A. Shah, and F. Doshi-Velez. Mind the Gap: A Generative Approach to Interpretable Feature Selection and Extraction. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 2260–2268, 2015.

[8] J. Kittler and J. Föglein. Contextual Classification of Multispectral Pixel Data. *IMage and Vision Computing*, 2(1):13–29, 1984.

[9] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, 47:498–519, 2001.

[10] S. L. Lauritzen and D. J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. *Journal of the Royal Statistical Society*, 50:157–224, 1988.

[11] H.-A. Loeliger. An Introduction to Factor Graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, 2004.

[12] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng. Turbo Decoding as an Instance of Pearl's "Belief Propagation" Algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140–152, 1998.

[13] T. P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, Jan. 2001.

References II

[14] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[15] O. Shental, D. Bickson, J. K. W. P. H. Siegel and, and D. Dolev. Gaussian Belief Propagatio Solver for Systems of Linear Equations. In *IEEE International Symposium on Information Theory*, 2008.

[16] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint Appearance, Shape and Context Modeling for Mulit-Class Object Recognition and Segmentation. In *Proceedings of the European Conference on Computer Vision*, 2006.

[17] L. E. Sucar and D. F. Gillies. Probabilistic Reasoning in High-Level Vision. *Image and Vision Computing*, 12(1):42–60, 1994.

[18] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, A. A. Vladimir Kolmogorov, M. Tappen, and C. Rother. A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-based Priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1068–1080, 2008.