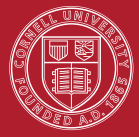


Cornell University

Lecture 24 - Graphical Models and Markov Random Fields

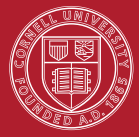
ECE 5260 – ORIE 5735

Anna Scaglione

Cornell Tech

April 30, 2025

Content



Cornell University

Review

In the last lecture we talked about:

- Bayesian Estimation, and MAP as well as MMSE estimators (or classifiers)
- Bayesian Networks (BN) models, when applicable, can simplify the computations associated to MAP and MMSE estimates when we deal with multivariate experiments.
- Markov Random Fields (MRF) are, ultimately, the structure that is used for computations
- More specifically, MRF are mapped in another graph type called Factor Graph (FG) that aids the description of the MAP and MMSE algorithms

Review

In the last lecture we talked about:

- Bayesian Estimation, and MAP as well as MMSE estimators (or classifiers)
- Bayesian Networks (BN) models, when applicable, can simplify the computations associated to MAP and MMSE estimates when we deal with multivariate experiments.
- Markov Random Fields (MRF) are, ultimately, the structure that is used for computations
- More specifically, MRF are mapped in another graph type called Factor Graph (FG) that aids the description of the MAP and MMSE algorithms

Review

In the last lecture we talked about:

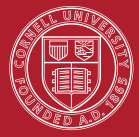
- Bayesian Estimation, and MAP as well as MMSE estimators (or classifiers)
- Bayesian Networks (BN) models, when applicable, can simplify the computations associated to MAP and MMSE estimates when we deal with multivariate experiments.
- Markov Random Fields (MRF) are, ultimately, the structure that is used for computations
- More specifically, MRF are mapped in another graph type called Factor Graph (FG) that aids the description of the MAP and MMSE algorithms

Review

In the last lecture we talked about:

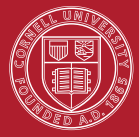
- Bayesian Estimation, and MAP as well as MMSE estimators (or classifiers)
- Bayesian Networks (BN) models, when applicable, can simplify the computations associated to MAP and MMSE estimates when we deal with multivariate experiments.
- Markov Random Fields (MRF) are, ultimately, the structure that is used for computations
- More specifically, MRF are mapped in another graph type called Factor Graph (FG) that aids the description of the MAP and MMSE algorithms

Overview



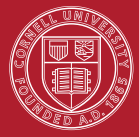
- We concluded by introducing the "Sum-Product" algorithm using Factor Graphs to calculate marginal distributions and posteriors
- After some basic examples to understand the mechanism we will discuss applications to
  - Dynamic systems, via Hidden Markov Models
  - Knowledge Networks
  - Biological Networks

Overview



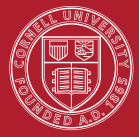
- We concluded by introducing the "Sum-Product" algorithm using Factor Graphs to calculate marginal distributions and posteriors
- After some basic examples to understand the mechanism we will discuss applications to
  - Dynamic systems, via Hidden Markov Models
  - Knowledge Networks
  - Biological Networks

Overview



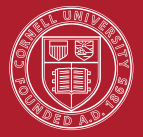
- We concluded by introducing the "Sum-Product" algorithm using Factor Graphs to calculate marginal distributions and posteriors
- After some basic examples to understand the mechanism we will discuss applications to
  - Dynamic systems, via Hidden Markov Models
  - Knowledge Networks
  - Biological Networks

Overview



- We concluded by introducing the "Sum-Product" algorithm using Factor Graphs to calculate marginal distributions and posteriors
- After some basic examples to understand the mechanism we will discuss applications to
  - Dynamic systems, via Hidden Markov Models
  - Knowledge Networks
  - Biological Networks

Outline



Cornell University

Bayesian estimation

- Maximum A-posteriori Probability estimation approach:

$$\text{MAP Bayesian}: \Rightarrow \quad \hat{\boldsymbol{\theta}}_{\text{MAP}}(X) := \underset{\boldsymbol{\theta}}{\text{argmax}}\, p(\boldsymbol{\theta}|X)$$

$$\equiv \underset{\boldsymbol{\theta}}{\text{argmax}} \{\overbrace{L(\boldsymbol{X}, \boldsymbol{\theta})}^{\ln p(X|\boldsymbol{\theta})} + \ln p(\boldsymbol{\theta})\}$$

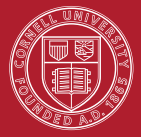
- Optimizing Bayesian risk/loss considering the distribution of Θ:

$$\hat{\boldsymbol{\theta}}(X) = \underset{f(\mathrm{X})}{\text{argmin}}\, \mathbb{E}[\mathcal{L}(f(\mathrm{X}), \Theta)]$$

- **Special case** $\mathcal{L}(f(\mathrm{X}), \Theta) = \|f(X) - \boldsymbol{\theta}\|^2$ Mean Squared Error (MSE) and the minimum of $MSE(\hat{\boldsymbol{\theta}}(X)) = \mathbb{E}[\|(\hat{\boldsymbol{\theta}}(\mathrm{X}) - \Theta\|^2]$ is:

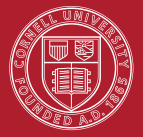
$$\text{MMSE Bayesian} \Rightarrow \quad \hat{\boldsymbol{\theta}}_{\text{MMSE}}(\mathrm{X}) = \mathbb{E}[\Theta|X]$$

Outline



Cornell University

We introduced the general idea....

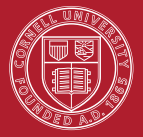


- Conditional independence allows to express the joint distribution (probability density/mass function) in terms of products of factors that depends on a **subset** of the random variables.
- This can be expressed by writing the joint distribution as,

$$p\Big(x_1, x_2, \cdots, x_l\Big) = \prod_{i=1}^{l} p\Big(x_i | \mathrm{Pa}_i\Big),$$

where Pa_i denotes the subset of variables associated with the random variable x_i (the "parents").

We introduced the general idea....



- Conditional independence allows to express the joint distribution (probability density/mass function) in terms of products of factors that depends on a **subset** of the random variables.
- This can be expressed by writing the joint distribution as,

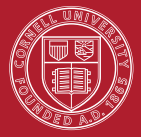
$$p\Big(x_1, x_2, \cdots, x_l\Big) = \prod_{i=1}^{l} p\Big(x_i | \mathrm{Pa}_i\Big),$$

where Pa_i denotes the subset of variables associated with the random variable x_i (the "parents").

Undirected Graphical Models

- Modeling and computations favor undirected graphical models or Markov Networks
- Each node of the graph is associated with a random variable. Edges connecting nodes are **undirected**.
- Statistical interactions are expressed via functions known as potential functions or compatibility functions or factors nonnegative functions of their arguments.
- The global PDF is the product of these local potential functions like in BN with conditional probabilities. In fact, a BN can always be converted into an MRF.

Undirected Graphical Models



- Modeling and computations favor undirected graphical models or Markov Networks
- Each node of the graph is associated with a random variable. Edges connecting nodes are **undirected**.
- Statistical interactions are expressed via functions known as potential functions or compatibility functions or factors nonnegative functions of their arguments.
- The global PDF is the product of these local potential functions like in BN with conditional probabilities. In fact, a BN can always be converted into an MRF.

Undirected Graphical Models

- Modeling and computations favor undirected graphical models or Markov Networks
- Each node of the graph is associated with a random variable. Edges connecting nodes are **undirected**.
- Statistical interactions are expressed via functions known as potential functions or compatibility functions or factors nonnegative functions of their arguments.
- The global PDF is the product of these local potential functions like in BN with conditional probabilities. In fact, a BN can always be converted into an MRF.

Markov Random Fields: potential functions

- The random variables $\mathrm{x}_1, \ldots, \mathrm{x}_N$ are divided in K groups, $\mathbf{x}_1, \ldots, \mathbf{x}_K$; each random vector, $\mathbf{x}_k$, $k = 1, 2, \ldots, K$, involves a subset of the random variables.

Gibbs distribution

A distribution is called Gibbs distribution if it can expressed in terms of a set of potential functions, $\psi_1, \ldots, \psi_K$ as:

$$p(x_1, \ldots, x_l) = \frac{1}{Z} \prod_{k=1}^{K} \psi_k(\boldsymbol{x}_k).$$

The constant Z is the normalizing constant to guarantee that $p(x_1, \ldots, x_l)$ is a distribution, i.e.:

$$Z = \int \cdots \int \prod_{k=1}^{K} \psi_k(\boldsymbol{x}_k) dx_1 \ldots dx_l,$$

which becomes a summation for the case of probabilities.

Examples of parametric MRF

- **The Ising model**: Each random variable takes binary values in $\{-1, 1\}$ and that the joint probability distribution is given by the following model,

$$p(x_1, \ldots, x_l) := p(\boldsymbol{x}) = \frac{1}{Z} \exp\Big(-\sum_i \Big(\sum_{j>i} \theta_{ij} x_i x_j + \theta_{i0} x_i \Big) \Big),$$

  where $\theta_{ij} = 0$ if the respective nodes are not connected.

- The Potts model: This is a generalization of the Ising model:

$$P(\boldsymbol{x}) = \frac{1}{Z} \exp\Big(-\sum_i g_i(x_i) + \sum_{j>i} f_{ij}(x_i, x_j) \Big),$$

  it is applicable to encode *network interactions* through the functions $f_{ij}(x_i, x_j)$

Examples of parametric MRF

- **The Ising model**: Each random variable takes binary values in $\{-1, 1\}$ and that the joint probability distribution is given by the following model,

$$p(x_1, \ldots, x_l) := p(\boldsymbol{x}) = \frac{1}{Z} \exp\Big(-\sum_i \Big(\sum_{j>i} \theta_{ij} x_i x_j + \theta_{i0} x_i \Big) \Big),$$

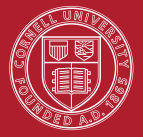
  where $\theta_{ij} = 0$ if the respective nodes are not connected.

- The Potts model: This is a generalization of the Ising model:

$$P(\boldsymbol{x}) = \frac{1}{Z} \exp\Big(-\sum_i g_i(x_i) + \sum_{j>i} f_{ij}(x_i, x_j) \Big),$$

  it is applicable to encode *network interactions* through the functions $f_{ij}(x_i, x_j)$

Examples of parametric MRF



- **The Ising model**: Each random variable takes binary values in $\{-1, 1\}$ and that the joint probability distribution is given by the following model,

$$p(x_1, \ldots, x_l) := p(\boldsymbol{x}) = \frac{1}{Z} \exp\Big(-\sum_i \Big(\sum_{j>i} \theta_{ij} x_i x_j + \theta_{i0} x_i \Big) \Big),$$

  where $\theta_{ij} = 0$ if the respective nodes are not connected.

- The Potts model: This is a generalization of the Ising model:

$$P(\boldsymbol{x}) = \frac{1}{Z} \exp\Big(-\sum_i g_i(x_i) + \sum_{j>i} f_{ij}(x_i, x_j) \Big),$$

  it is applicable to encode *network interactions* through the functions $f_{ij}(x_i, x_j)$

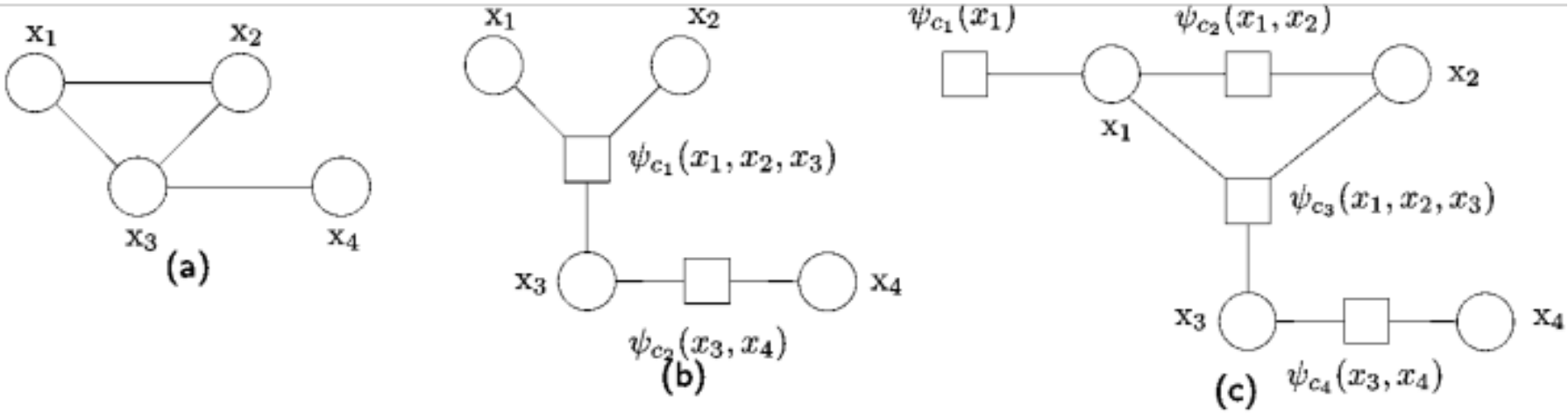
Outline

Cornell University

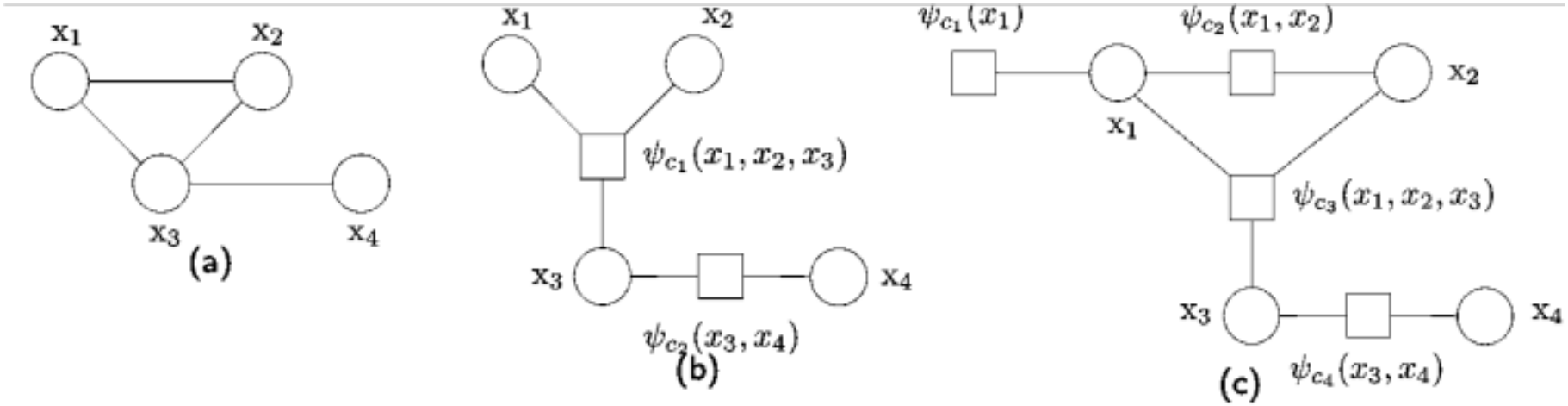
Factor Graphs

Factor graphs

A factor graph is an undirected **bipartite** graph involving two types of nodes (thus the term bipartite): one that corresponds to the random variables, denoted by circles, and one to the potential functions, denoted by squares. Edges exist **only** between two different types of nodes, i.e., between "potential function" nodes and "variable" nodes.



From MRF Networks to Factor Graphs



- Starting from the MRF in Figure (a) We have either:

Figure (b) $$p(x_1, x_2, x_3, x_4) = \frac{1}{Z}\psi_{c_1}(x_1, x_2, x_3)\psi_{c_2}(x_3, x_4).$$

Figure (c) $$p(x_1,x_2,x_3,x_4) = \frac{1}{Z}\psi_{c_1}(x_1)\psi_{c_2}(x_1, x_2)\psi_{c_3}(x_1, x_2, x_3)\psi_{c_4}(x_3, x_4).$$

Moralization of BN Directed Graphs

- One can convert a **Bayesian Network** (BN) to an MRF
- For this conversion, the conditional distributions will play the role of the potential factors.
- To ensure that edges do exist among **all** the involved variables in each one of these factors since edges from the parents to children exist, we must:
  a) retain these edges and make them undirected
  b) add edges between parents of a common child.
- Step (b) is known as moralization ("parents of a child are forced to be married") and the resulting undirected graph is called the moral graph.

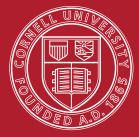
Moralization of BN Directed Graphs

- One can convert a **Bayesian Network** (BN) to an MRF
- For this conversion, the conditional distributions will play the role of the potential factors.
- To ensure that edges do exist among **all** the involved variables in each one of these factors since edges from the parents to children exist, we must:
  a) retain these edges and make them undirected
  b) add edges between parents of a common child.
- Step (b) is known as moralization ("parents of a child are forced to be married") and the resulting undirected graph is called the moral graph.

Moralization of BN Directed Graphs

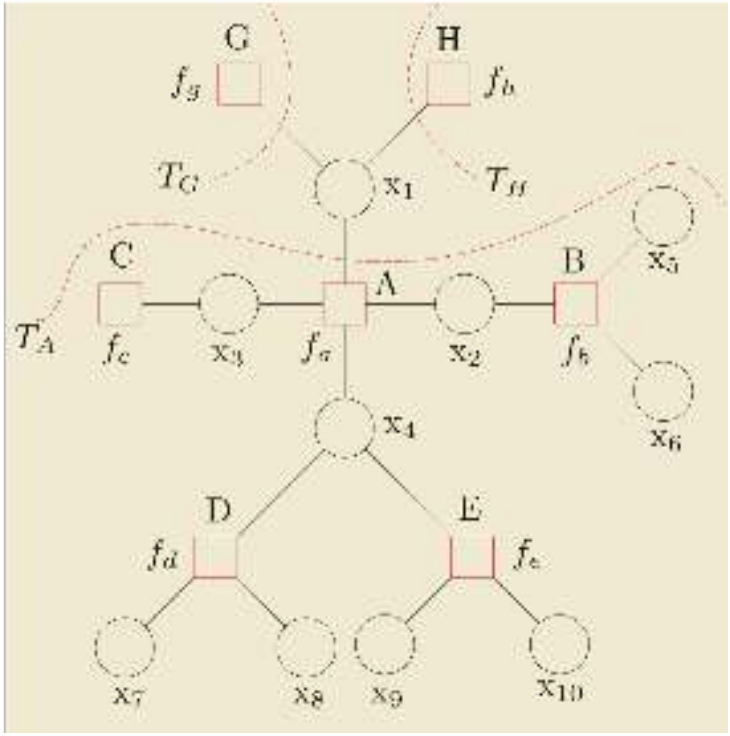
- One can convert a **Bayesian Network** (BN) to an MRF
- For this conversion, the conditional distributions will play the role of the potential factors.
- To ensure that edges do exist among **all** the involved variables in each one of these factors since edges from the parents to children exist, we must:
  a) retain these edges and make them undirected
  b) add edges between parents of a common child.
- Step (b) is known as moralization ("parents of a child are forced to be married") and the resulting undirected graph is called the moral graph.

Outline



Cornell University

The Sum–Product Algorithm



Factor nodes are capital letters and squares and are associated with a potential function f_x. The rest are variable nodes, denoted by circles.

Assume that we want to compute the marginal $P(x_1)$. Node x_1, being a variable node, is connected to factor nodes **only**.

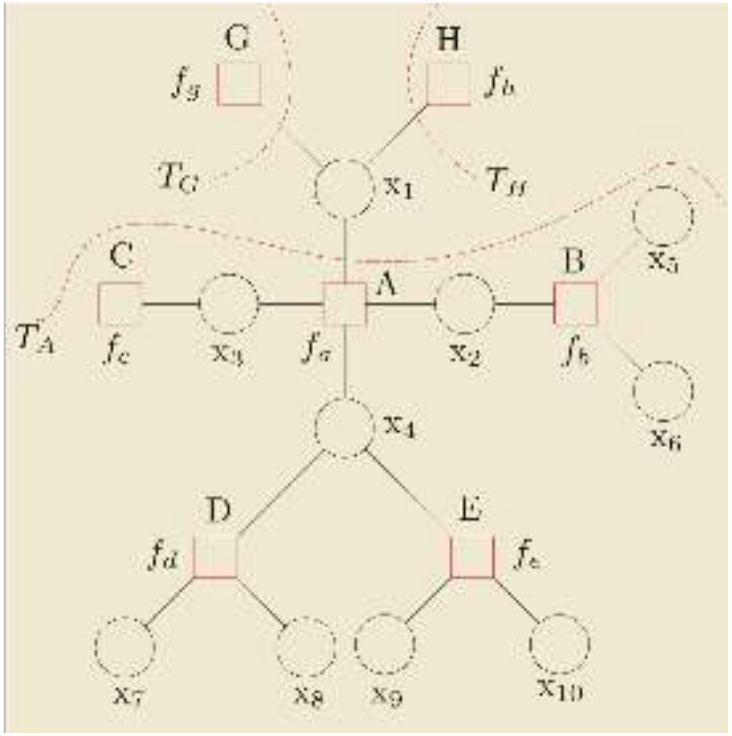
We split the graph in as many subgraphs as the factor nodes connected to x_1 (three in the figure, each encircled having as roots the nodes A, H and G, respectively.

- Focusing on the node of interest, x_1, the factorization is:

$$P(\boldsymbol{x}) = \frac{1}{Z}\psi_A(x_1, \boldsymbol{x}_A)\psi_H(x_1, \boldsymbol{x}_H)\psi_G(x_1, \boldsymbol{x}_G),$$

where $\boldsymbol{x}_A$ is the vector corresponding to all the variables in the tree T_A; the vectors $\boldsymbol{x}_H$ and $\boldsymbol{x}_G$ are similarly defined.

The Sum–Product Algorithm



Factor nodes are capital letters and squares and are associated with a potential function f_x. The rest are variable nodes, denoted by circles. Assume that we want to compute the marginal $P(x_1)$. Node x_1, being a variable node, is connected to factor nodes **only**.

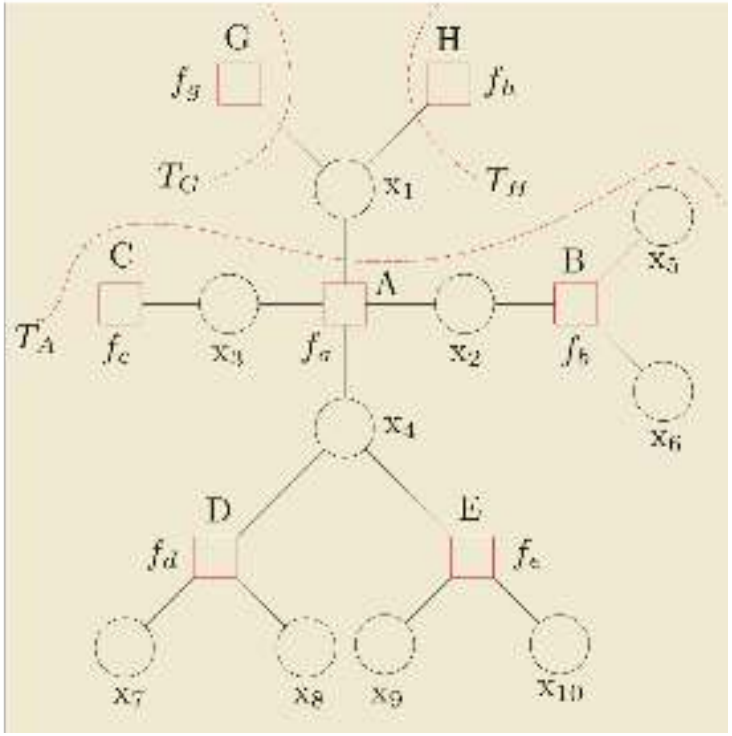
We split the graph in as many subgraphs as the factor nodes connected to x_1 (three in the figure, each encircled having as roots the nodes A, H and G, respectively.

- Focusing on the node of interest, x_1, the factorization is:

$$P(\boldsymbol{x}) = \frac{1}{Z}\psi_A(x_1, \boldsymbol{x}_A)\psi_H(x_1, \boldsymbol{x}_H)\psi_G(x_1, \boldsymbol{x}_G),$$

where $\boldsymbol{x}_A$ is the vector corresponding to all the variables in the tree T_A; the vectors $\boldsymbol{x}_H$ and $\boldsymbol{x}_G$ are similarly defined.

The Sum–Product Algorithm



Factor nodes are capital letters and squares and are associated with a potential function f_x. The rest are variable nodes, denoted by circles. Assume that we want to compute the marginal $P(x_1)$. Node x_1, being a variable node, is connected to factor nodes **only**.

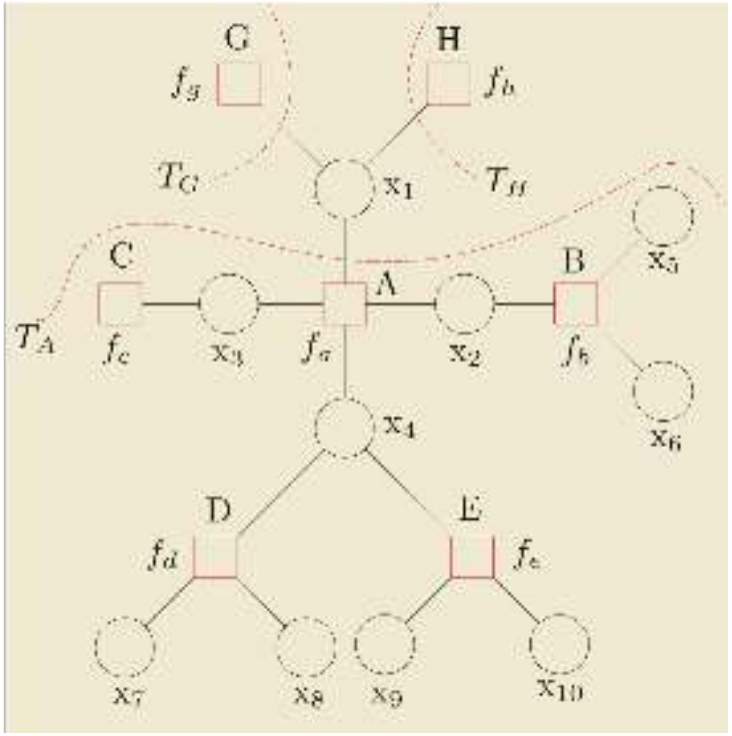
We split the graph in as many subgraphs as the factor nodes connected to x_1 (three in the figure, each encircled having as roots the nodes A, H and G, respectively.

- Focusing on the node of interest, x_1, the factorization is:

$$P(\boldsymbol{x}) = \frac{1}{Z}\psi_A(x_1, \boldsymbol{x}_A)\psi_H(x_1, \boldsymbol{x}_H)\psi_G(x_1, \boldsymbol{x}_G),$$

where $\boldsymbol{x}_A$ is the vector corresponding to all the variables in the tree T_A; the vectors $\boldsymbol{x}_H$ and $\boldsymbol{x}_G$ are similarly defined.

The Sum–Product Algorithm



Factor nodes are capital letters and squares and are associated with a potential function f_x. The rest are variable nodes, denoted by circles. Assume that we want to compute the marginal $P(x_1)$. Node x_1, being a variable node, is connected to factor nodes **only**.

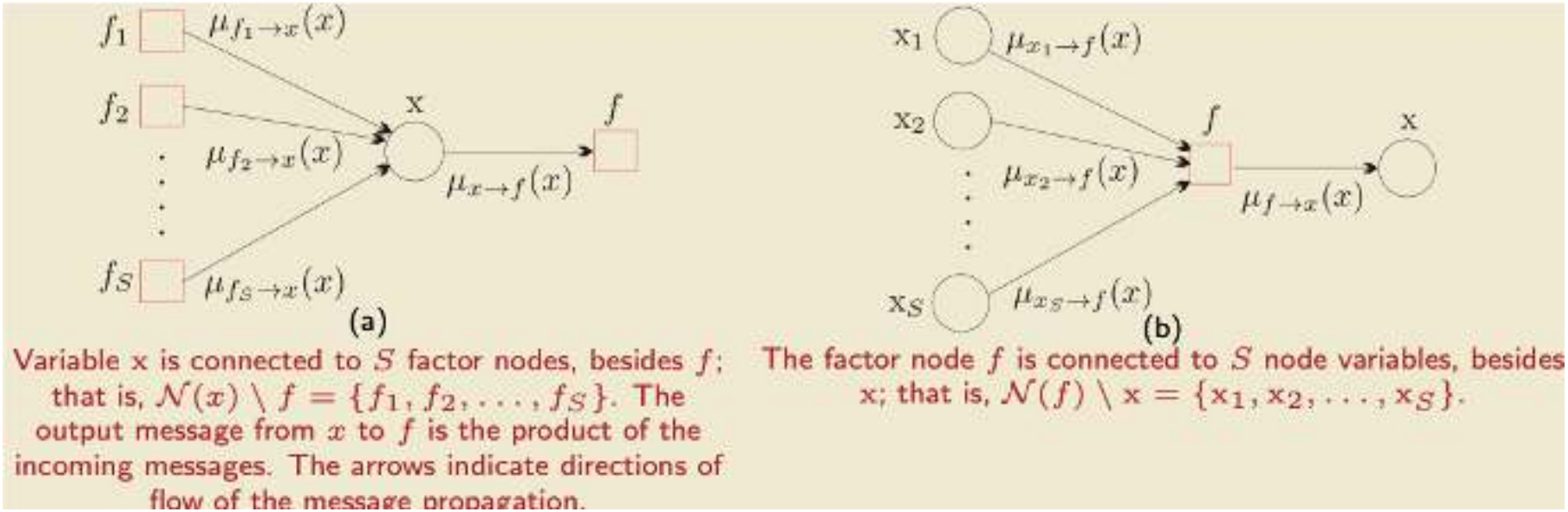
We split the graph in as many subgraphs as the factor nodes connected to x_1 (three in the figure, each encircled having as roots the nodes A, H and G, respectively.

- Focusing on the node of interest, x_1, the factorization is:

$$P(\boldsymbol{x}) = \frac{1}{Z}\psi_A(x_1, \boldsymbol{x}_A)\psi_H(x_1, \boldsymbol{x}_H)\psi_G(x_1, \boldsymbol{x}_G),$$

where $\boldsymbol{x}_A$ is the vector corresponding to all the variables in the tree T_A; the vectors $\boldsymbol{x}_H$ and $\boldsymbol{x}_G$ are similarly defined.

Sum-Product Operations summary



Variable x is connected to S factor nodes, besides f; that is, $\mathcal{N}(x) \setminus f = \{f_1, f_2, \ldots, f_S\}$. The output message from x to f is the product of the incoming messages. The arrows indicate directions of flow of the message propagation.

The factor node f is connected to S node variables, besides x; that is, $\mathcal{N}(f) \setminus \mathrm{x} = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_S\}$.

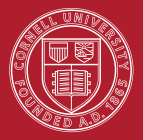
- Variable to factor node messages (Figure (a)):

$$\mu_{x \to f}(x) = \prod_{s: f_s \in \mathcal{N}(\mathrm{x}) \setminus f} \mu_{f_s \to x}(x),$$

- Factor to variable node messages (Figure (b)):

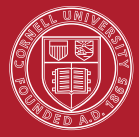
$$\mu_{f \to x}(x) = \sum_{x_i \in \mathcal{N}(f) \setminus \mathrm{x}} f(\boldsymbol{x}^f) \prod_{i: x_i \in \mathcal{N}(f) \setminus \mathrm{x}} \mu_{x_i \to f}(x_i),$$

The Sum–Product Algorithm



- The message passing algorithm starts by initializing the leafs, as:
  - If the leaf node is a variable node, x, connected to a factor node, f, then,
$$\mu_{x \to f}(x) = 1.$$
  - If the leaf node is a factor node, f, connected to variable node, x then,
$$\mu_{f \to x}(x) = f(x).$$
- Recall that our original goal was to compute
$$P(x_1) = \frac{1}{Z} \sum_{\boldsymbol{x}_A \in V_A} \psi_A(x_1, \boldsymbol{x}_A) \sum_{\boldsymbol{x}_H \in V_H} \psi_H(x_1, \boldsymbol{x}_H) \sum_{\boldsymbol{x}_G \in V_G} \psi_G(x_1, \boldsymbol{x}_G).$$

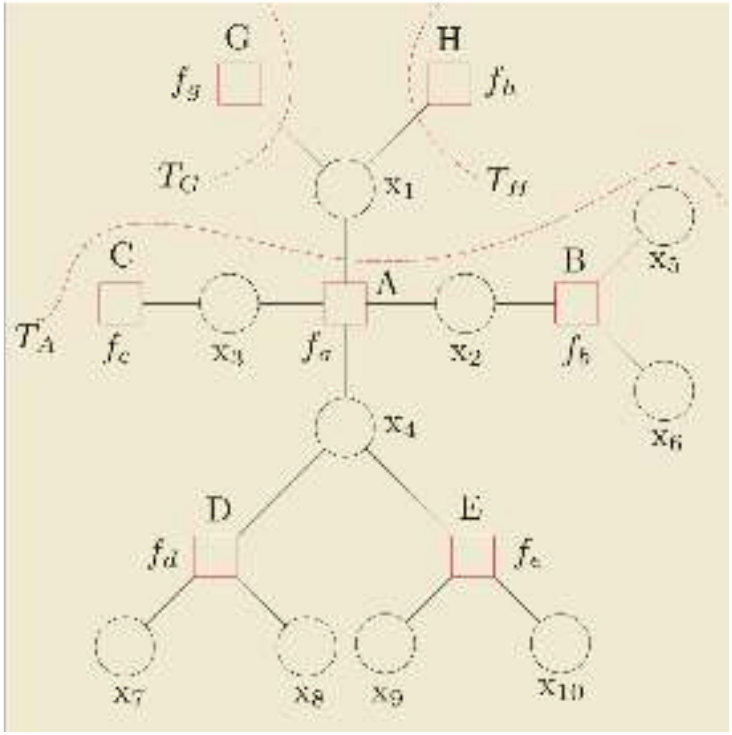
The Sum–Product Algorithm



- The message passing algorithm starts by initializing the leafs, as:
  - If the leaf node is a variable node, x, connected to a factor node, f, then,
$$\mu_{x \to f}(x) = 1.$$
  - If the leaf node is a factor node, f, connected to variable node, x then,
$$\mu_{f \to x}(x) = f(x).$$
- Recall that our original goal was to compute

$$P(x_1) = \frac{1}{Z} \sum_{\boldsymbol{x}_A \in V_A} \psi_A(x_1, \boldsymbol{x}_A) \sum_{\boldsymbol{x}_H \in V_H} \psi_H(x_1, \boldsymbol{x}_H) \sum_{\boldsymbol{x}_G \in V_G} \psi_G(x_1, \boldsymbol{x}_G).$$

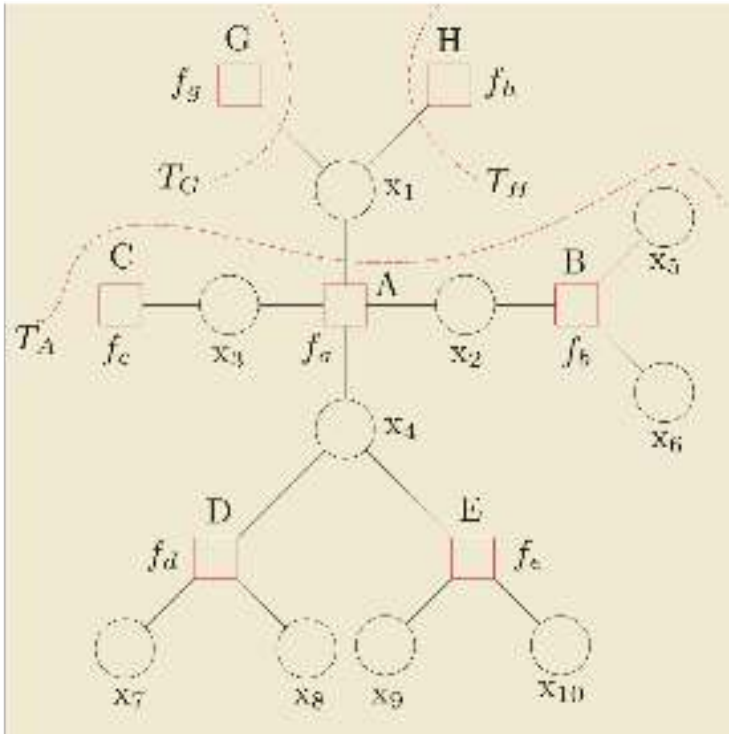
The Sum-Product algorithm



$$\sum_{\boldsymbol{x}_A \in V_A} \psi_A(x_1, \boldsymbol{x}_A) = \sum_{x_2,x_3,x_4} f_a(x_1, x_2, x_3, x_4)\mu_{x_2 \to f_a}(x_2)\mu_{x_3 \to f_a}(x_3)\mu_{x_4 \to f_a}(x_4)$$

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

The Sum-Product algorithm



Working similarly for the other two subtrees, T_G and T_H

$$P(x_1) = \frac{1}{Z}\mu_{f_a \to x_1}(x_1)\mu_{f_g \to x_1}(x_1)\mu_{f_h \to x_1}(x_1).$$

The algorithmic steps

The sum-product algorithm

1. Pick the variable, x, whose marginal, $P(x)$, will be computed.
2. Divide the tree to as **many subtrees** as the **number of factor nodes** the variable node, x, is connected to.
3. For each one of these subtrees, identify the **leaf nodes**.
4. Start message passing, towards x, by **initializing the leaf nodes**.
5. Compute the marginal according to

$$P(x) = \frac{1}{Z} \prod_{s: f_s \in \mathcal{N}(\mathrm{x})} \mu_{f_s \to x}(x).$$

6. Adding both sides of the last equation, over all possible values of x we obtain the normalizing constant, Z.

Markov Random Fields: potential functions

- The random variables $\mathrm{x}_1, \ldots, \mathrm{x}_N$ are divided in K groups, $\mathbf{x}_1, \ldots, \mathbf{x}_K$; each random vector, $\mathbf{x}_k$, $k = 1, 2, \ldots, K$, involves a subset of the random variables.

Gibbs distribution

A distribution is called Gibbs distribution if it can expressed in terms of a set of potential functions, $\psi_1, \ldots, \psi_K$ as:

$$p(x_1, \ldots, x_l) = \frac{1}{Z} \prod_{k=1}^{K} \psi_k(\boldsymbol{x}_k).$$

The constant Z is the normalizing constant to guarantee that $p(x_1, \ldots, x_l)$ is a distribution, i.e.:

$$Z = \int \cdots \int \prod_{k=1}^{K} \psi_k(\boldsymbol{x}_k) dx_1 \ldots dx_l,$$

which becomes a summation for the case of probabilities.

The algorithm when observations are available

- If some variables are observed, then we replace the summation over the instantiated variables by a single term evaluated on the observed value.
- For instance, we we know $x_3 = 0$, instead of $\sum_{x_2,x_3,x_4} f_a(x_1, x_2, x_3, x_4)$ we compute $\sum_{x_2,x_4} f_a(x_1, x_2, 0, x_4)$
- We can then compute joint distributions and conditional distributions using the message passing algorithm we described.
- The sparser is the factor graph associated to the MRF the easier are the computations of the messages

The algorithm when observations are available

- If some variables are observed, then we replace the summation over the instantiated variables by a single term evaluated on the observed value.
- For instance, we we know $x_3 = 0$, instead of $\sum_{x_2,x_3,x_4} f_a(x_1, x_2, x_3, x_4)$ we compute $\sum_{x_2,x_4} f_a(x_1, x_2, 0, x_4)$
- We can then compute joint distributions and conditional distributions using the message passing algorithm we described.
- The sparser is the factor graph associated to the MRF the easier are the computations of the messages

The algorithm when observations are available

- If some variables are observed, then we replace the summation over the instantiated variables by a single term evaluated on the observed value.
- For instance, we we know $x_3 = 0$, instead of $\sum_{x_2,x_3,x_4} f_a(x_1, x_2, x_3, x_4)$ we compute $\sum_{x_2,x_4} f_a(x_1, x_2, 0, x_4)$
- We can then compute joint distributions and conditional distributions using the message passing algorithm we described.
- The sparser is the factor graph associated to the MRF the easier are the computations of the messages

The algorithm when observations are available

- If some variables are observed, then we replace the summation over the instantiated variables by a single term evaluated on the observed value.
- For instance, we we know $x_3 = 0$, instead of $\sum_{x_2,x_3,x_4} f_a(x_1, x_2, x_3, x_4)$ we compute $\sum_{x_2,x_4} f_a(x_1, x_2, 0, x_4)$
- We can then compute joint distributions and conditional distributions using the message passing algorithm we described.
- The sparser is the factor graph associated to the MRF the easier are the computations of the messages

The Max-Product and Max-Sum Algorithms

- Next turn our attention from marginals to modes of distributions. That is, given a distribution, $P(\boldsymbol{x})$, that factorizes over a tree (factor) graph, the task is to compute efficiently the quantity,

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}).$$

- This is what are called the Max-Product and Max-Sum Algorithms and are tied to MAP esitmation or detection

The Max-Product and Max-Sum Algorithms

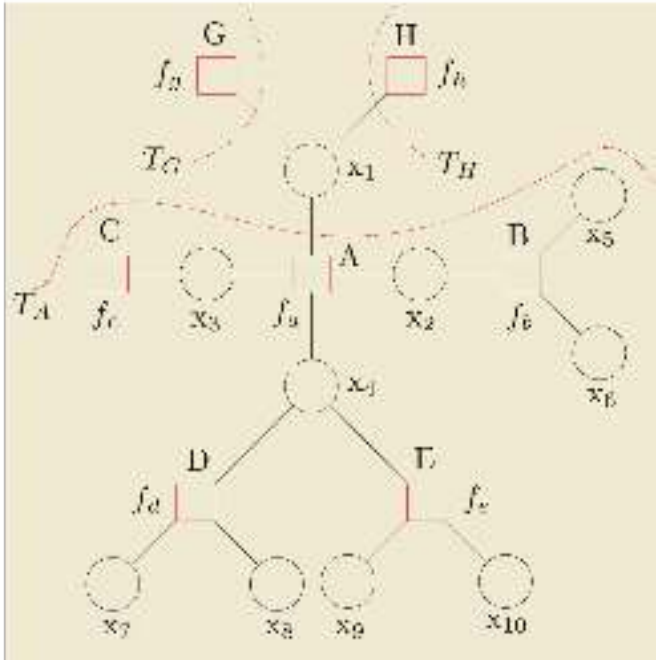
- Next turn our attention from marginals to modes of distributions. That is, given a distribution, $P(\boldsymbol{x})$, that factorizes over a tree (factor) graph, the task is to compute efficiently the quantity,

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}).$$

- This is what are called the Max-Product and Max-Sum Algorithms and are tied to MAP esitmation or detection

The Max-Product and Max-Sum Algorithms

- Following similar arguments as before, one can readily write:



$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x_1, \boldsymbol{x}_A, \boldsymbol{x}_H, \boldsymbol{x}_G} \Big\{ \psi_A(x_1, \boldsymbol{x}_A) \times \psi_H(x_1, \boldsymbol{x}_H) \psi_G(x_1, \boldsymbol{x}_G) \Big\}.$$

Exploiting the property of the max operator, i.e.,

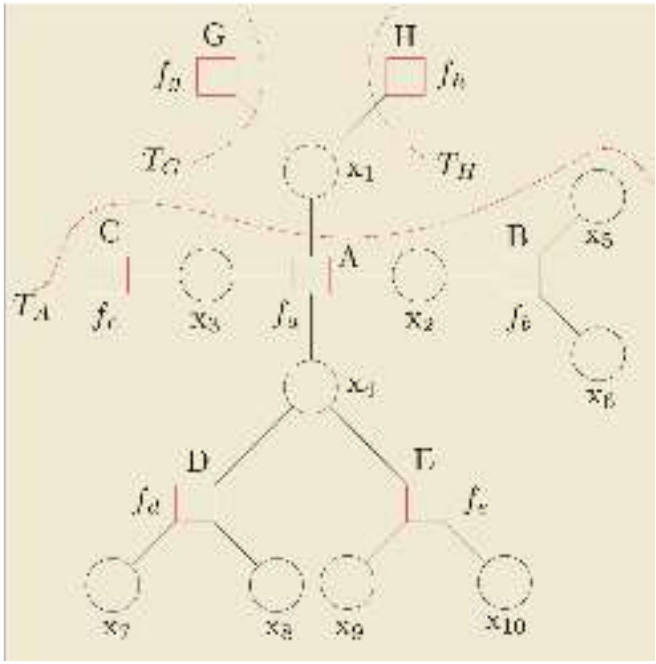
$$\max_{b,c}(ab, ac) = a \max(b, c) \ , \quad a \geq 0,$$

we can rewrite,

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x_1} \max_{\boldsymbol{x}_A \in V_A} \psi_A(x_1, \boldsymbol{x}_A) \times \max_{\boldsymbol{x}_H \in V_H} \psi_H(x_1, \boldsymbol{x}_H) \max_{\boldsymbol{x}_G \in V_G} \psi_G(x_1, \boldsymbol{x}_G).$$

The Max-Product and Max-Sum Algorithms

Following similar steps as in the sum-product algorithm:



$$\max_{x_1} \max_{\boldsymbol{x}_A \in V_A} \psi_A(x_1, \boldsymbol{x}_A) =$$

$$\max_{x_1} \max_{x_2,x_3,x_4} f_a(x_1, x_2, x_3, x_4) f_c(x_3) \times$$

$$\max_{x_7,x_8} f_d(x_4, x_7, x_8) \max_{x_9,x_{10}} f_e(x_4, x_9, x_{10})$$

$$\max_{x_5,x_6} f_b(x_2, x_6, x_5).$$

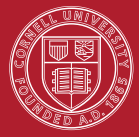
The Max-Product and Max-Sum Algorithms

- Everything said before for the sum-product message passing algorithm holds true if replace summations with the max operator; the definitions of the messages passed between nodes and factors change to,

$$\mu_{x \to f}(x) = \prod_{s: f_s \in \mathcal{N}(\mathrm{x}) \setminus f} \mu_{f_s \to x}(x),$$

$$\mu_{f \to x}(x) = \max_{x_i: x_i \in \mathcal{N}(f) \setminus \mathrm{x}} f(\boldsymbol{x}^f) \prod_{i: x_i \in \mathcal{N}(f) \setminus \mathrm{x}} \mu_{x_i \to f}(x_i).$$

Finding the mode



- Then, the mode of $P(\boldsymbol{x})$ is given by

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x_1} \mu_{f_a \to x_1}(x_1) \mu_{f_g \to x_1}(x_1) \mu_{f_h \to x_1}(x_1),$$

  or in general

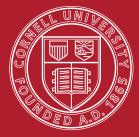
$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x} \prod_{s: f_s \in \mathcal{N}(\mathrm{x})} \mu_{f_s \to x}(x),$$

  where x plays the role of the root, towards which the flow of the messages is directed, starting from the leaves. This procedure is called max-product algorithm.

- Often, a large number of product terms is involved it may lead to arithmetic inaccuracies. We can use a log likelihood instead:

$$\boldsymbol{x}_* := \arg\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}} \ \ln P(\boldsymbol{x}).$$

Finding the mode



- Then, the mode of $P(\boldsymbol{x})$ is given by

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x_1} \mu_{f_a \to x_1}(x_1) \mu_{f_g \to x_1}(x_1) \mu_{f_h \to x_1}(x_1),$$

  or in general

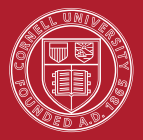
$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x} \prod_{s: f_s \in \mathcal{N}(\mathrm{x})} \mu_{f_s \to x}(x),$$

  where x plays the role of the root, towards which the flow of the messages is directed, starting from the leaves. This procedure is called max-product algorithm.

- Often, a large number of product terms is involved it may lead to arithmetic inaccuracies. We can use a log likelihood instead:

$$\boldsymbol{x}_* := \arg\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}} \ \ln P(\boldsymbol{x}).$$

Using the log-likelihood



- The message passage recursions are,

$$\mu_{x\to f}(x) = \sum_{s:f_s\in\mathcal{N}(\mathrm{x})\setminus f} \mu_{f_s\to x}(x),$$

$$\mu_{f\to x}(x) = \max_{x_i:x_i\in\mathcal{N}(f)\setminus\mathrm{x}} \left\{ \ln f(\boldsymbol{x}^f) + \sum_{i:x_i\in\mathcal{N}(f)\setminus\mathrm{x}} \mu_{x_i\to f}(x_i) \right\}.$$

- For the initial messages, sent by the leaf nodes, we now define,

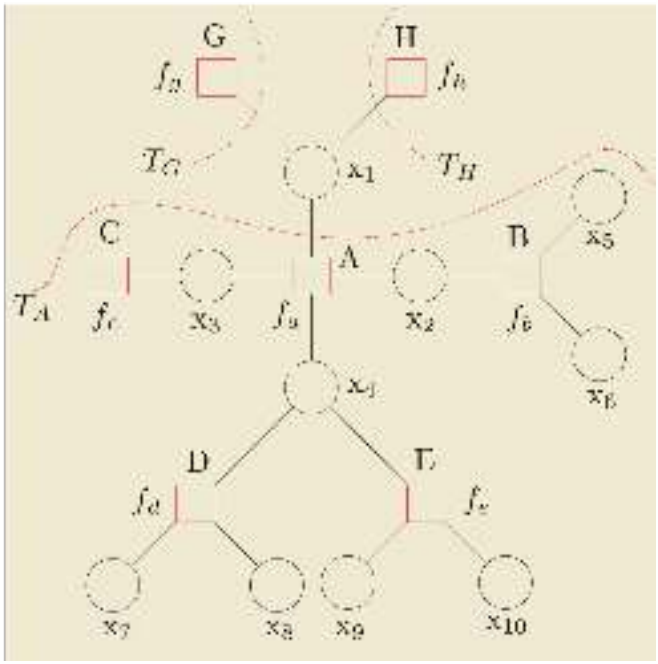
$$\mu_{x\to f}(x) = 0, \text{ and } \mu_{f\to x}(x) = \ln f(x).$$

- Note that after one pass of the message flow, the maximum value of $P(\boldsymbol{x})$ has been obtained. However, one also wants the corresponding value $\boldsymbol{x}_*$ at the maximum

$$\boldsymbol{x}_* = \arg\max_{\boldsymbol{x}} P(\boldsymbol{x}).$$

This is achieved by a reverse message passing a.k.a. back-tracking.

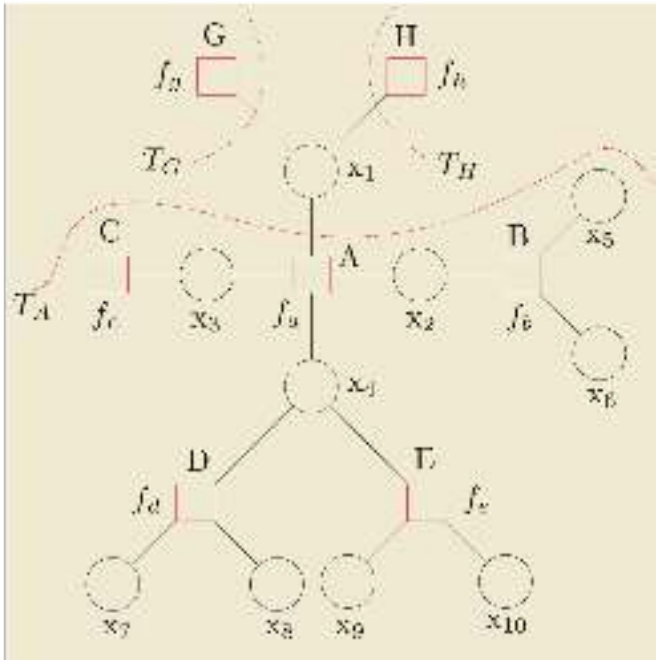
Back-tracking



- Back-tracking: Assume that x_1 is the chosen node to play the role of the root, where the flow of messages "converge". We have already obtained the value

$$x_{1*} = \arg\max_{x_1} \left\{ \mu_{f_a \to x_1}(x_1) \cdot \mu_{f_g \to x_1}(x_1) \mu_{f_f \to x_1}(x_1) \right\}.$$

Back-tracking

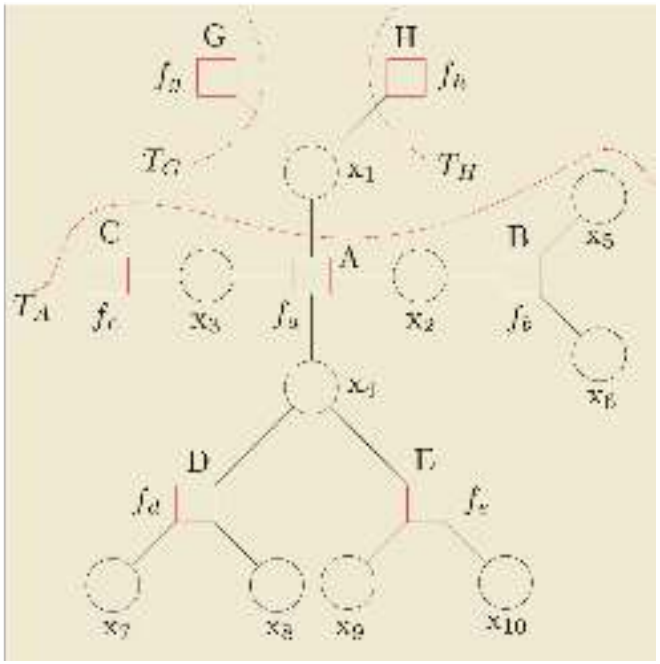


- Back-tracking: Assume that x_1 is the chosen node to play the role of the root, where the flow of messages "converge". We have already obtained the value

$$x_{1*} = \arg\max_{x_1} \left\{ \mu_{f_a \to x_1}(x_1) \cdot \mu_{f_g \to x_1}(x_1) \mu_{f_f \to x_1}(x_1) \right\}.$$

- A new message passing flow now starts, and the root node, x_1, passes the obtained value to the connected factor nodes.

Back-tracking

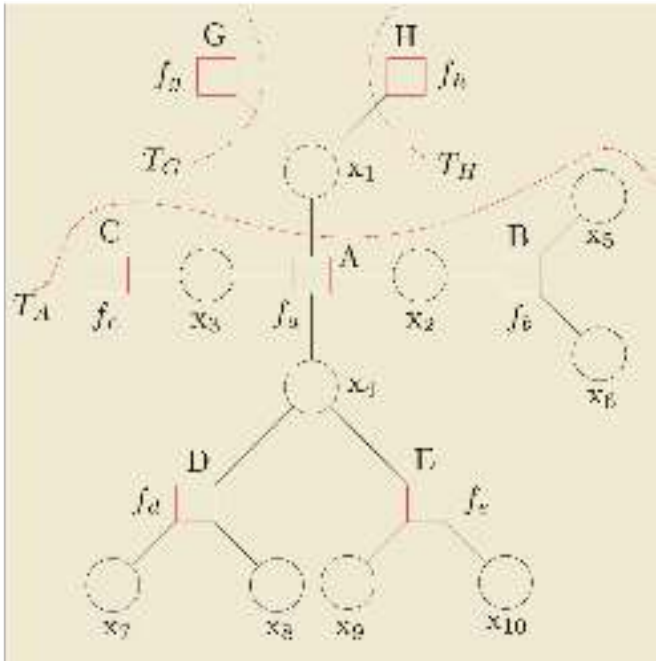


- Node A: It receives x_{1*} from x_1.
  Selection of the optimal values:
  Recall that

$$\mu_{f_a \to x_1}(x_1) = \max_{x_2,x_3,x_4} f_a(x_1, x_2, x_3, x_4) \times \mu_{x_4 \to f_a}(x_4)\mu_{x_3 \to f_a}(x_3)\mu_{x_2 \to f_a}(x_2).$$

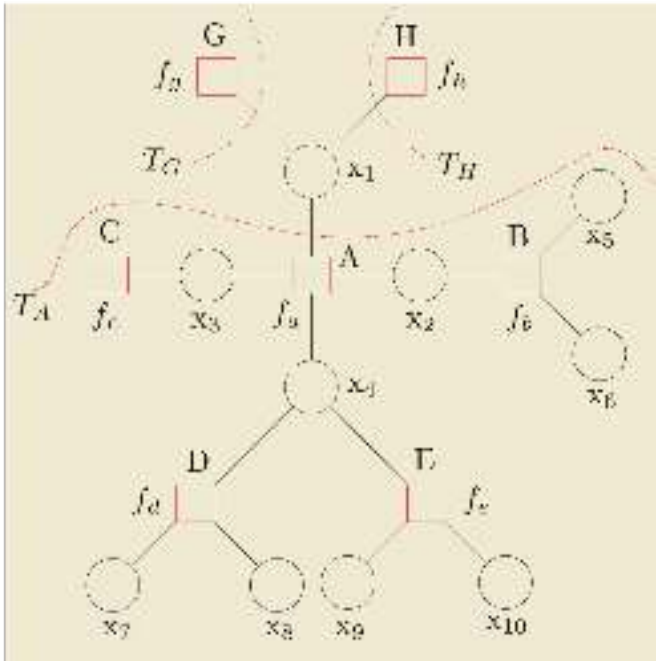
  For **different** values of x_1, there are **different optimal** (x_2, x_3, x_4).
  Having obtained a **specific** optimal value for x_{1*}, one has to use the corresponding optimal values for the triplet. Thus, during the first pass, the obtained optimal values in node A have to be stored to be used during the second (backward) pass

Back-tracking



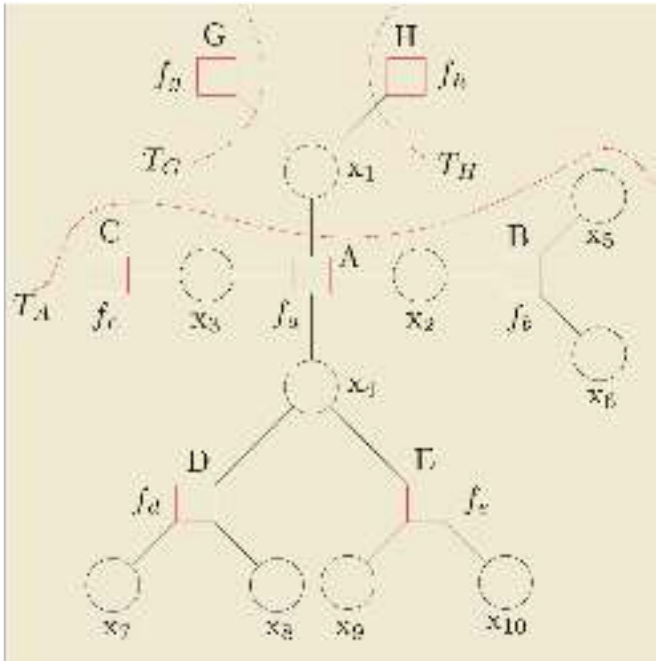
- Node A: It receives x_{1*} from node x_1 and selects the corresponding x_{4*}, x_{2*}, x_{3*}.
  - Message Passing: Node A passes x_{4*} to node x_4, x_{2*} to x_2 and x_{3*} to x_3.

Back-tracking



- Node x_4 passes x_{4*} to D and E.

Back-tracking

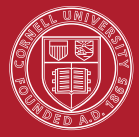


- Node D
  - Selection of the optimal values:
    Select (x_{7*}, x_{8*}) such as,

$$(x_{7*}, x_{8*}) = \arg\max_{x_7, x_8} f_d(x_{4*}, x_7, x_8) \times \mu_{x_7 \to f_d}(x_7) \times \mu_{x_8 \to f_d}(x_8).$$

  - Message Passing: Node D passes (x_{7*}, x_{8*}) to nodes x_7, x_8, respectively.
- Similarly for nodes B and E

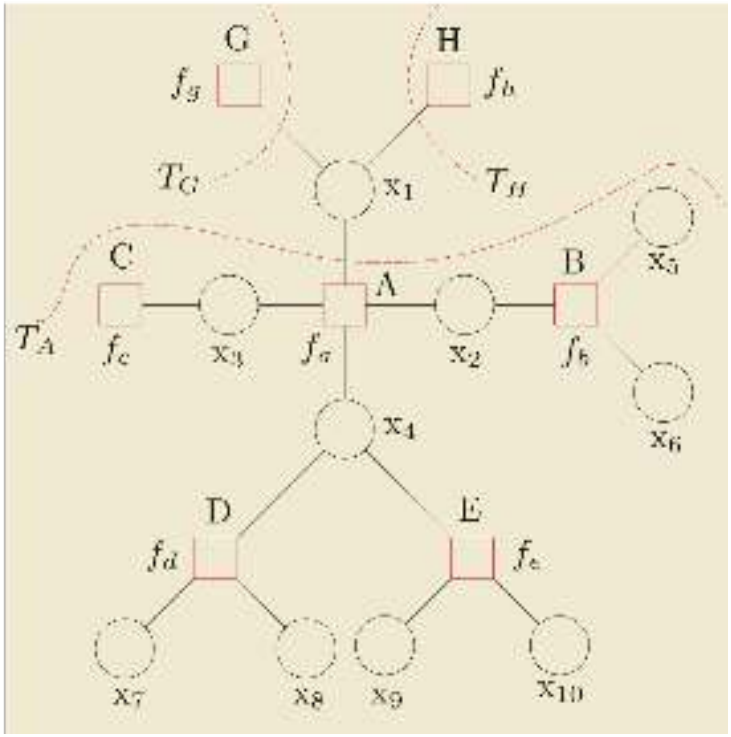
Outline



Cornell University

The general case

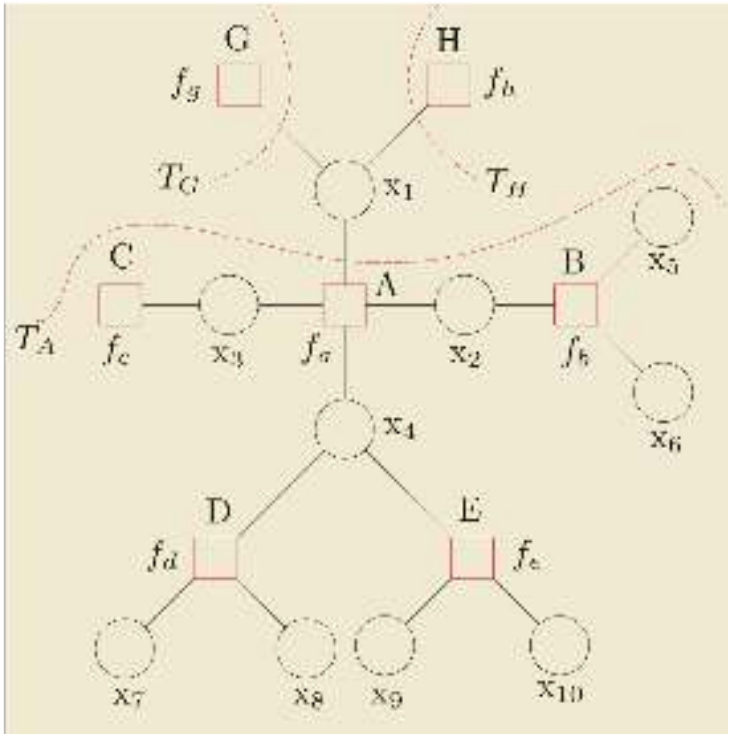
- In general, MRF are not trees like the one in the example:



- They always allow simplifications, but are highly dependent on how the factors are processed

The general case

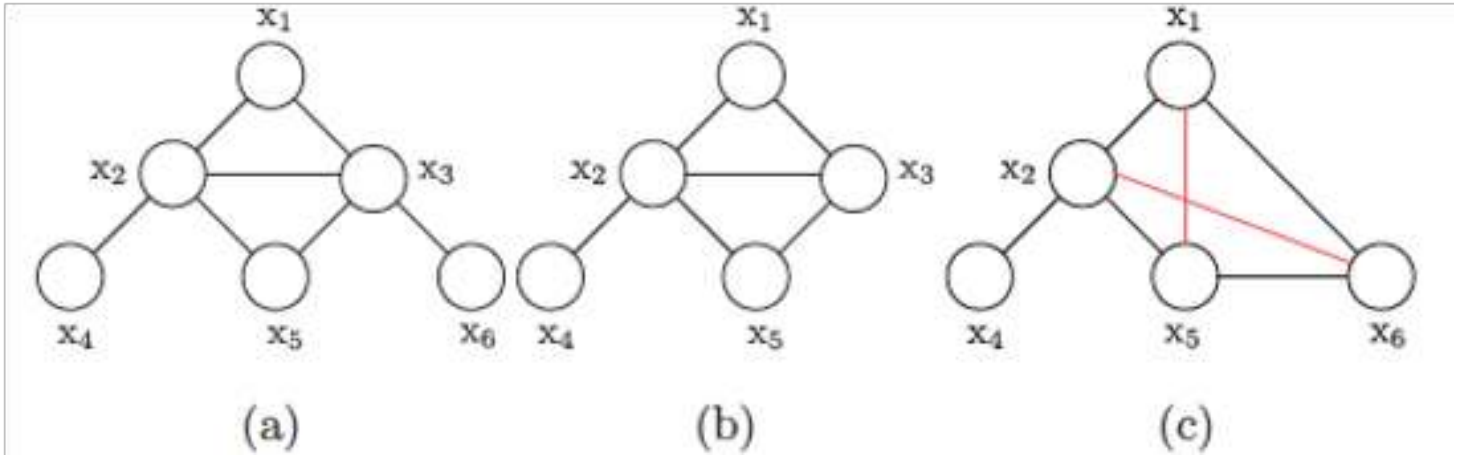
- In general, MRF are not trees like the one in the example:



- They always allow simplifications, but are highly dependent on how the factors are processed

Example

- For example, the MRF shown has $P(\boldsymbol{x}) = \frac{1}{Z}\psi_1(x_1)\psi_2(x_1,x_2)\psi_3(x_1,x_3)\psi_4(x_2,x_4)\psi_5(x_2,x_3,x_5)\psi_6(x_3,x_6)$



- If we eliminate x_6 first we get (b):

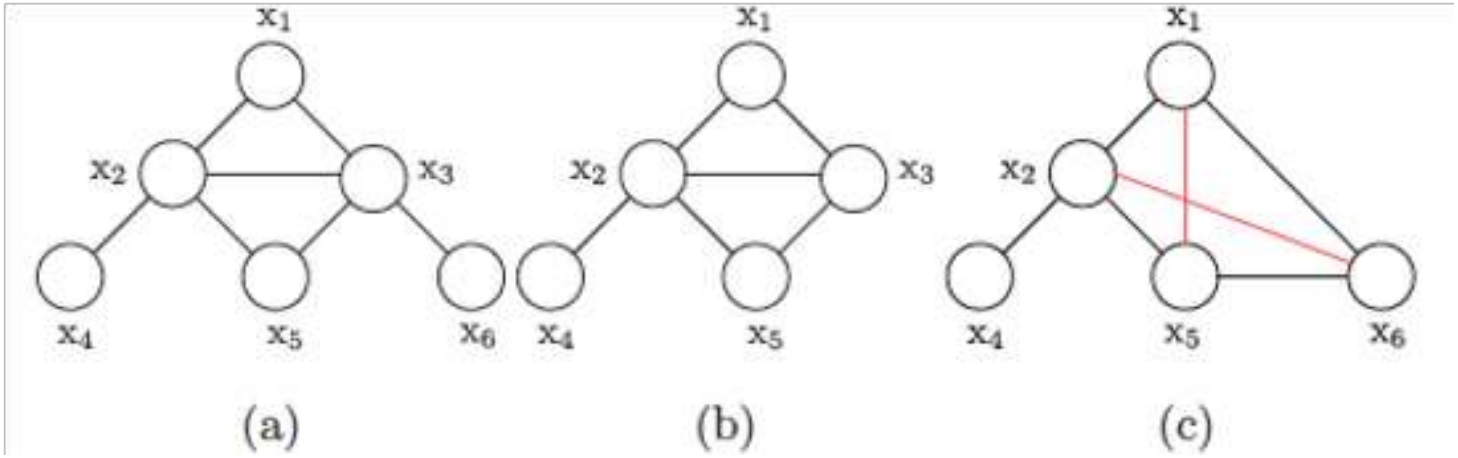
$$\sum_{x_6} P(\boldsymbol{x}) = \psi'(x_1,\ldots,x_5)\sum_{x_6}\psi_6(x_3,x_6)$$

- If we eliminate x_3 first we get (c):

$$\sum_{x_6} P(\boldsymbol{x}) = \psi''(x_1,x_2,x_4)\sum_{x_3}\psi_3(x_1,x_3)\psi_5(x_2,x_3,x_5)\psi_6(x_3,x_6)$$

Example

- For example, the MRF shown has $P(\boldsymbol{x}) = \frac{1}{Z}\psi_1(x_1)\psi_2(x_1,x_2)\psi_3(x_1,x_3)\psi_4(x_2,x_4)\psi_5(x_2,x_3,x_5)\psi_6(x_3,x_6)$



- If we eliminate x_6 first we get (b):

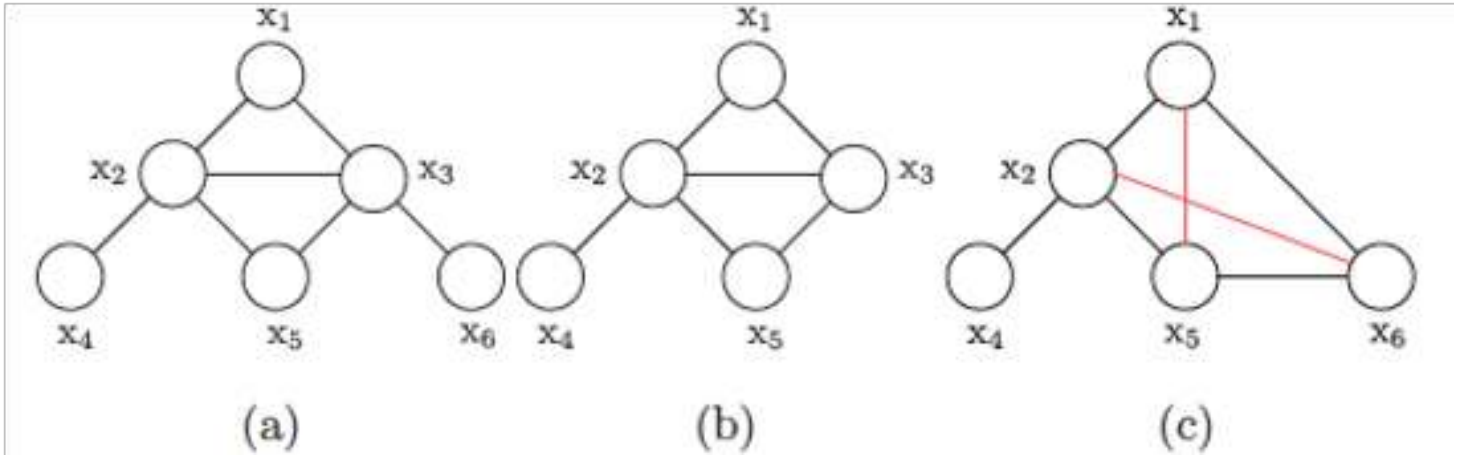
$$\sum_{x_6} P(\boldsymbol{x}) = \psi'(x_1,\ldots,x_5)\sum_{x_6}\psi_6(x_3,x_6)$$

- If we eliminate x_3 first we get (c):

$$\sum_{x_6} P(\boldsymbol{x}) = \psi''(x_1,x_2,x_4)\sum_{x_3}\psi_3(x_1,x_3)\psi_5(x_2,x_3,x_5)\psi_6(x_3,x_6)$$

Example

- For example, the MRF shown has $P(\boldsymbol{x}) = \frac{1}{Z}\psi_1(x_1)\psi_2(x_1,x_2)\psi_3(x_1,x_3)\psi_4(x_2,x_4)\psi_5(x_2,x_3,x_5)\psi_6(x_3,x_6)$



- If we eliminate x_6 first we get (b):

$$\sum_{x_6} P(\boldsymbol{x}) = \psi'(x_1,\ldots,x_5)\sum_{x_6}\psi_6(x_3,x_6)$$

- If we eliminate x_3 first we get (c):

$$\sum_{x_6} P(\boldsymbol{x}) = \psi''(x_1,x_2,x_4)\sum_{x_3}\psi_3(x_1,x_3)\psi_5(x_2,x_3,x_5)\psi_6(x_3,x_6)$$

General approaches for inference

- There are two main approaches: (a) Constructing a Junction Tree; (b) Applying what is called Loopy Belief Propagation
- (b) Loopy BP is exactly like the case described for trees, but does not end in two passes. We cannot guarantee it will converge
- The junction tree approach (a) is more reliable, but requires preprocessing and works if and only if the original graph is triangulated

Triangulated graph

An undirected graph is said to be triangulated if and only if for every cycle of length greater than three the graph has a chord. A chord is an edge joining two nonconsecutive nodes in the cycle.

General approaches for inference

- There are two main approaches: (a) Constructing a Junction Tree; (b) Applying what is called Loopy Belief Propagation
- (b) Loopy BP is exactly like the case described for trees, but does not end in two passes. We cannot guarantee it will converge
- The junction tree approach (a) is more reliable, but requires preprocessing and works if and only if the original graph is triangulated

Triangulated graph

An undirected graph is said to be triangulated if and only if for every cycle of length greater than three the graph has a chord. A chord is an edge joining two nonconsecutive nodes in the cycle.

General approaches for inference

- There are two main approaches: (a) Constructing a Junction Tree; (b) Applying what is called Loopy Belief Propagation
- (b) Loopy BP is exactly like the case described for trees, but does not end in two passes. We cannot guarantee it will converge
- The junction tree approach (a) is more reliable, but requires preprocessing and works if and only if the original graph is triangulated

Triangulated graph

An undirected graph is said to be triangulated if and only if for every cycle of length greater than three the graph has a chord. A chord is an edge joining two nonconsecutive nodes in the cycle.

General approaches for inference

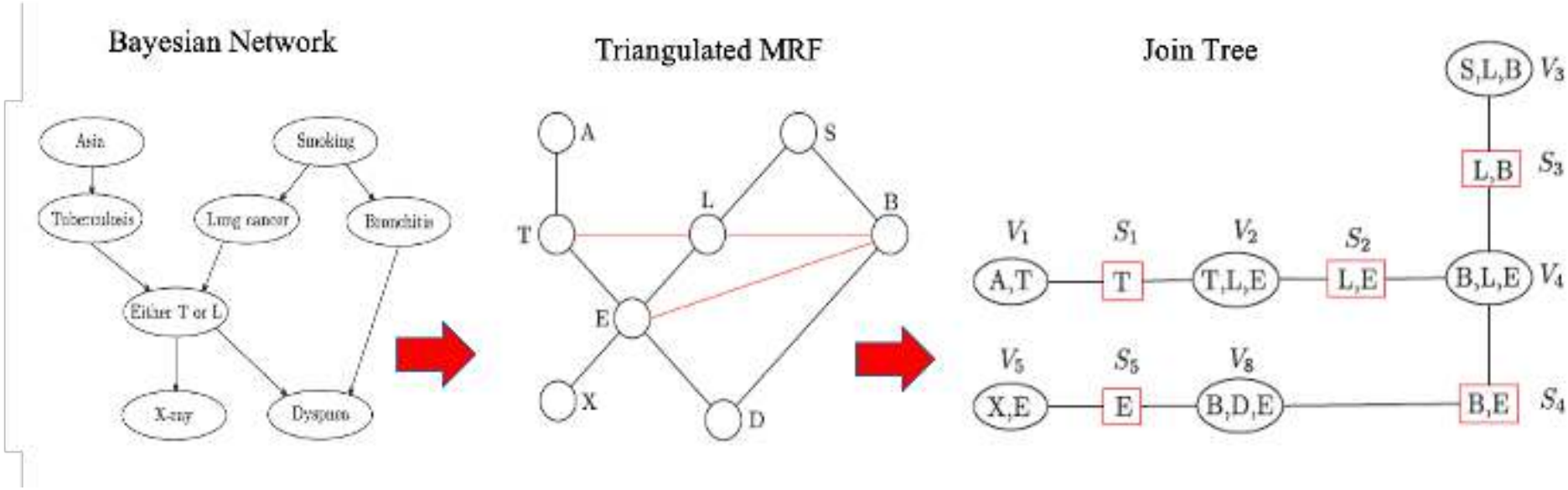
- There are two main approaches: (a) Constructing a Junction Tree; (b) Applying what is called Loopy Belief Propagation
- (b) Loopy BP is exactly like the case described for trees, but does not end in two passes. We cannot guarantee it will converge
- The junction tree approach (a) is more reliable, but requires preprocessing and works if and only if the original graph is triangulated

Triangulated graph

An undirected graph is said to be triangulated if and only if for every cycle of length greater than three the graph has a chord. A chord is an edge joining two nonconsecutive nodes in the cycle.

Example

,


Bayesian Network
Asia
Smoking
Tuberculosis
Lung cancer
Bronchitis
Either T or L
X-ray
Dyspnea
Triangulated MRF
A
S
L
B
T
E
X
D
Join Tree
S,L,B
V_3
L,B
S_3
V_1
S_1
V_2
S_2
A,T
T
T,L,E
L,E
B,L,E
V_4
V_5
S_5
V_8
X,E
E
B,D,E
B,E
S_4

Construction of Junction Trees from MRF

- Starting from a triangulated graph (all cycles with length $n > 3$ have a chord) we said one can build a join tree (or junction tree) we showed on the left of the previous figure
- Nodes in the junction tree are cliques and they have the running intersection property: the path between two join tree nodes corresponding to clique U and V contains the intersection of them $U \cap V$
- The join trees, instead of factors and variables like factor graphs, are bipartite graphs that connect the clique V_i and the clique $V_j, j > i$ through a node/set called separator $S_i \subset V_j$,

Construction of Junction Trees from MRF

- Starting from a triangulated graph (all cycles with length $n > 3$ have a chord) we said one can build a join tree (or junction tree) we showed on the left of the previous figure
- Nodes in the junction tree are cliques and they have the running intersection property: the path between two join tree nodes corresponding to clique U and V contains the intersection of them $U \cap V$
- The join trees, instead of factors and variables like factor graphs, are bipartite graphs that connect the clique V_i and the clique $V_j, j > i$ through a node/set called separator $S_i \subset V_j$,

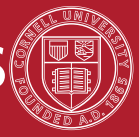
Construction of Junction Trees from MRF

- Starting from a triangulated graph (all cycles with length $n > 3$ have a chord) we said one can build a join tree (or junction tree) we showed on the left of the previous figure
- Nodes in the junction tree are cliques and they have the running intersection property: the path between two join tree nodes corresponding to clique U and V contains the intersection of them $U \cap V$
- The join trees, instead of factors and variables like factor graphs, are bipartite graphs that connect the clique V_i and the clique $V_j, j > i$ through a node/set called separator $S_i \subset V_j$,

Construction of Junction tree from MRFs

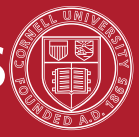
1. Select a node in a clique of the triangulated graph, not shared by other cliques.
2. Place nodes in the set V_i from the clique, as long as they are not shared by other cliques.
3. Denote the set of the remaining nodes of this clique as S_i, where i is the number of the nodes eliminated so far. This set is called a separator.
4. Select another maximal clique and repeat the process to form V_{i+1}.
5. Continue the process until all cliques have been placed in a set V_j and have a separator S_j.
6. Once all nodes are in a set V_i, join together the parts so that each separator, S_i, is joined to V_i on one of its sides and to a clique node (set) V_j, $(j > i)$, such that $S_i \subset V_j$.

Construction of Junction tree from MRFs



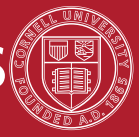
1. Select a node in a clique of the triangulated graph, not shared by other cliques.
2. Place nodes in the set V_i from the clique, as long as they are not shared by other cliques.
3. Denote the set of the remaining nodes of this clique as S_i, where i is the number of the nodes eliminated so far. This set is called a separator.
4. Select another maximal clique and repeat the process to form V_{i+1}.
5. Continue the process until all cliques have been placed in a set V_j and have a separator S_j.
6. Once all nodes are in a set V_i, join together the parts so that each separator, S_i, is joined to V_i on one of its sides and to a clique node (set) V_j, $(j > i)$, such that $S_i \subset V_j$.

Construction of Junction tree from MRFs



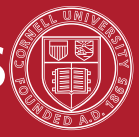
1. Select a node in a clique of the triangulated graph, not shared by other cliques.
2. Place nodes in the set V_i from the clique, as long as they are not shared by other cliques.
3. Denote the set of the remaining nodes of this clique as S_i, where i is the number of the nodes eliminated so far. This set is called a separator.
4. Select another maximal clique and repeat the process to form V_{i+1}.
5. Continue the process until all cliques have been placed in a set V_j and have a separator S_j.
6. Once all nodes are in a set V_i, join together the parts so that each separator, S_i, is joined to V_i on one of its sides and to a clique node (set) V_j, $(j > i)$, such that $S_i \subset V_j$.

Construction of Junction tree from MRFs



1. Select a node in a clique of the triangulated graph, not shared by other cliques.
2. Place nodes in the set V_i from the clique, as long as they are not shared by other cliques.
3. Denote the set of the remaining nodes of this clique as S_i, where i is the number of the nodes eliminated so far. This set is called a separator.
4. Select another maximal clique and repeat the process to form V_{i+1}.
5. Continue the process until all cliques have been placed in a set V_j and have a separator S_j.
6. Once all nodes are in a set V_i, join together the parts so that each separator, S_i, is joined to V_i on one of its sides and to a clique node (set) V_j, $(j > i)$, such that $S_i \subset V_j$.

Construction of Junction tree from MRFs

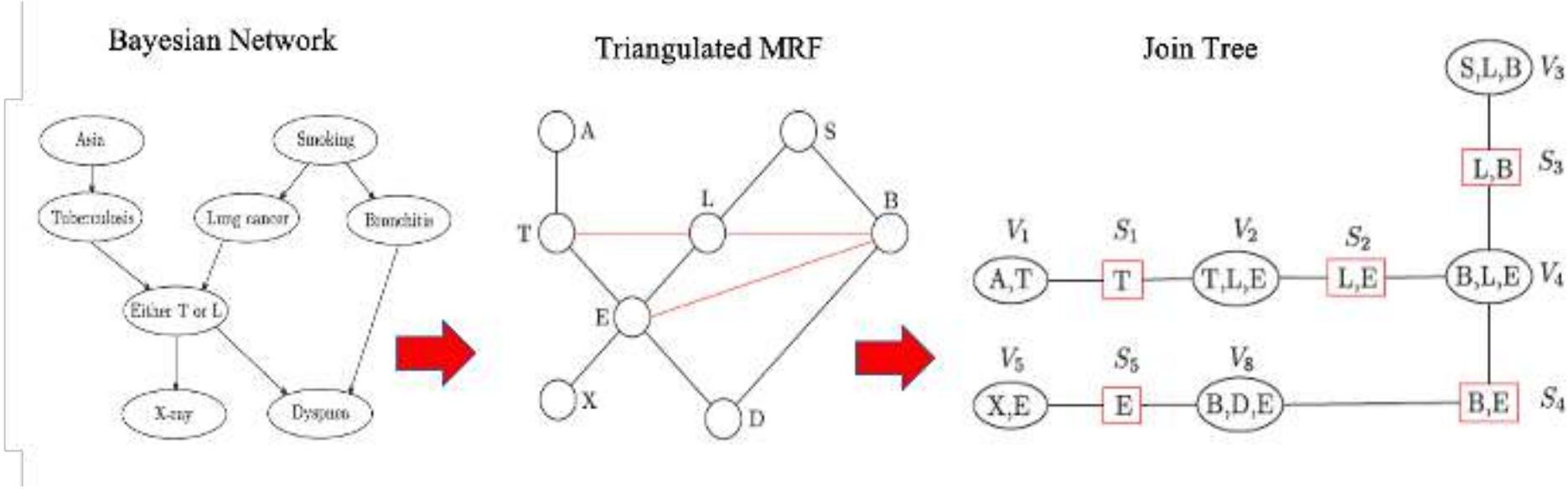


1. Select a node in a clique of the triangulated graph, not shared by other cliques.
2. Place nodes in the set V_i from the clique, as long as they are not shared by other cliques.
3. Denote the set of the remaining nodes of this clique as S_i, where i is the number of the nodes eliminated so far. This set is called a separator.
4. Select another maximal clique and repeat the process to form V_{i+1}.
5. Continue the process until all cliques have been placed in a set V_j and have a separator S_j.
6. Once all nodes are in a set V_i, join together the parts so that each separator, S_i, is joined to V_i on one of its sides and to a clique node (set) V_j, $(j > i)$, such that $S_i \subset V_j$.

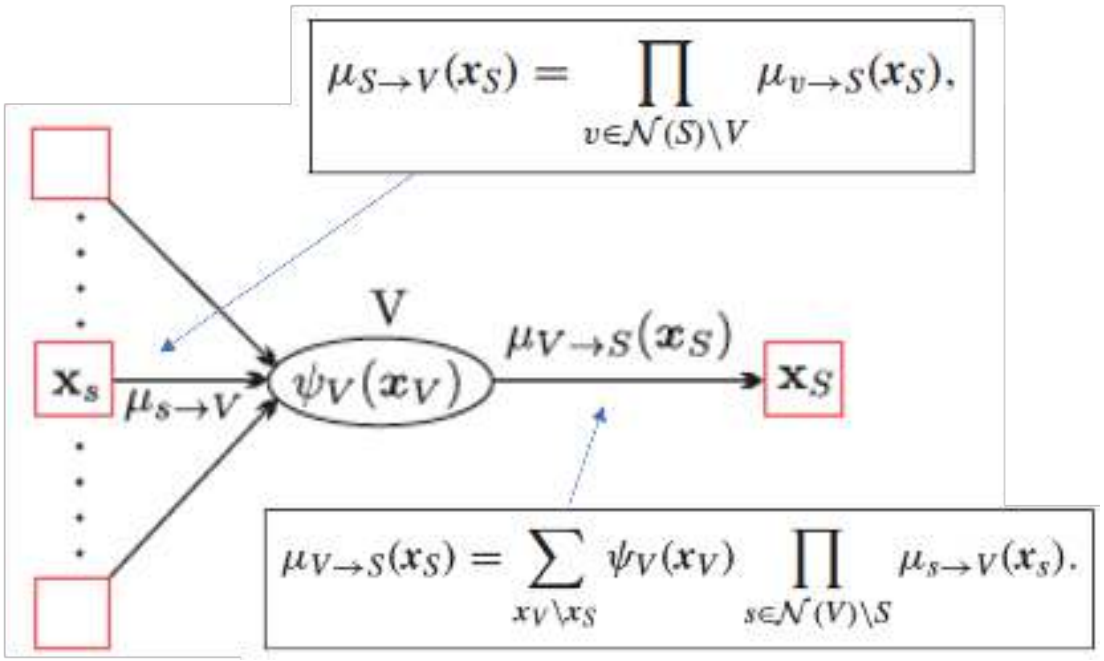
Construction of Junction Trees from MRF

Cornell University

,

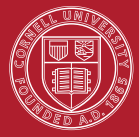


Sum-product and max-product on junction trees



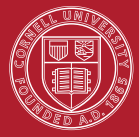
- Now summations include more than one variable at a time
- Separators S pass messages to one of its connected cliques V
- Each clique marginalizes w.r.t. all clique variables except those in the separator $\boldsymbol{x}_V \setminus \boldsymbol{x}_S$ and passes the message to the connected separators.

Outline



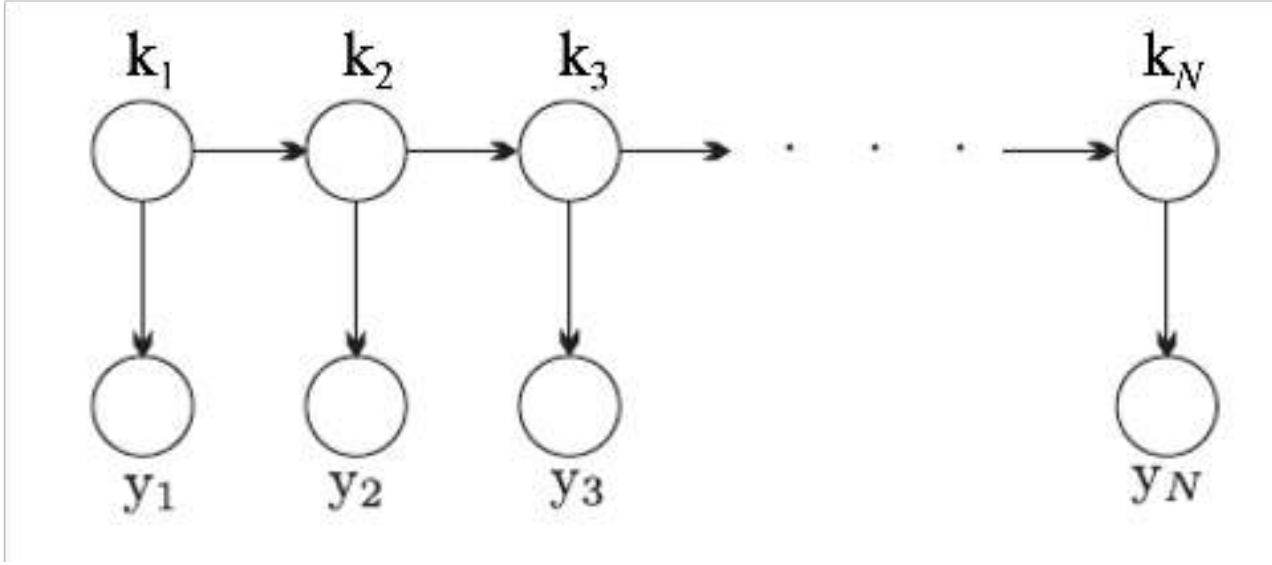
Cornell University

Outline



Cornell University

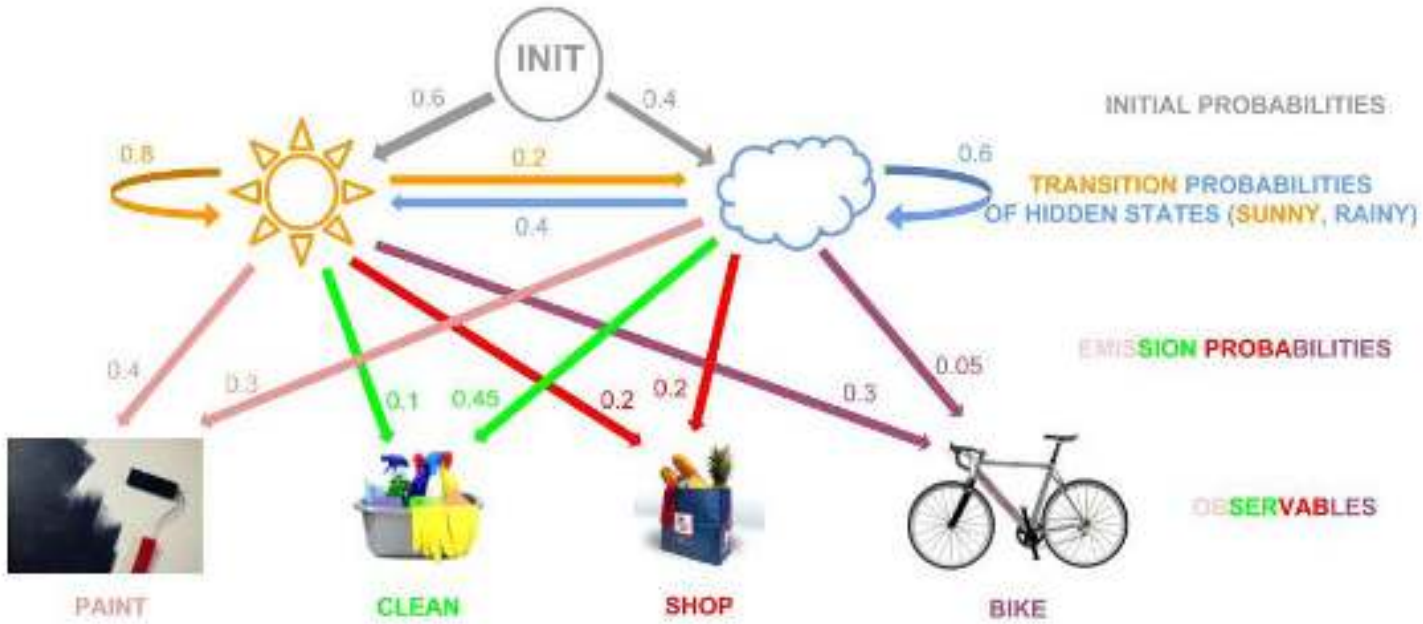
Hidden Markov Models



- The HMM is used to model a quasi-stationary process that undergoes sudden changes among K subprocesses, where K is the number of possible latent states of a Markov Chain process k_n.
- Conventionally the transition probability matrix $P_n = P$ is constant (i.e. it is an homogeneous process):

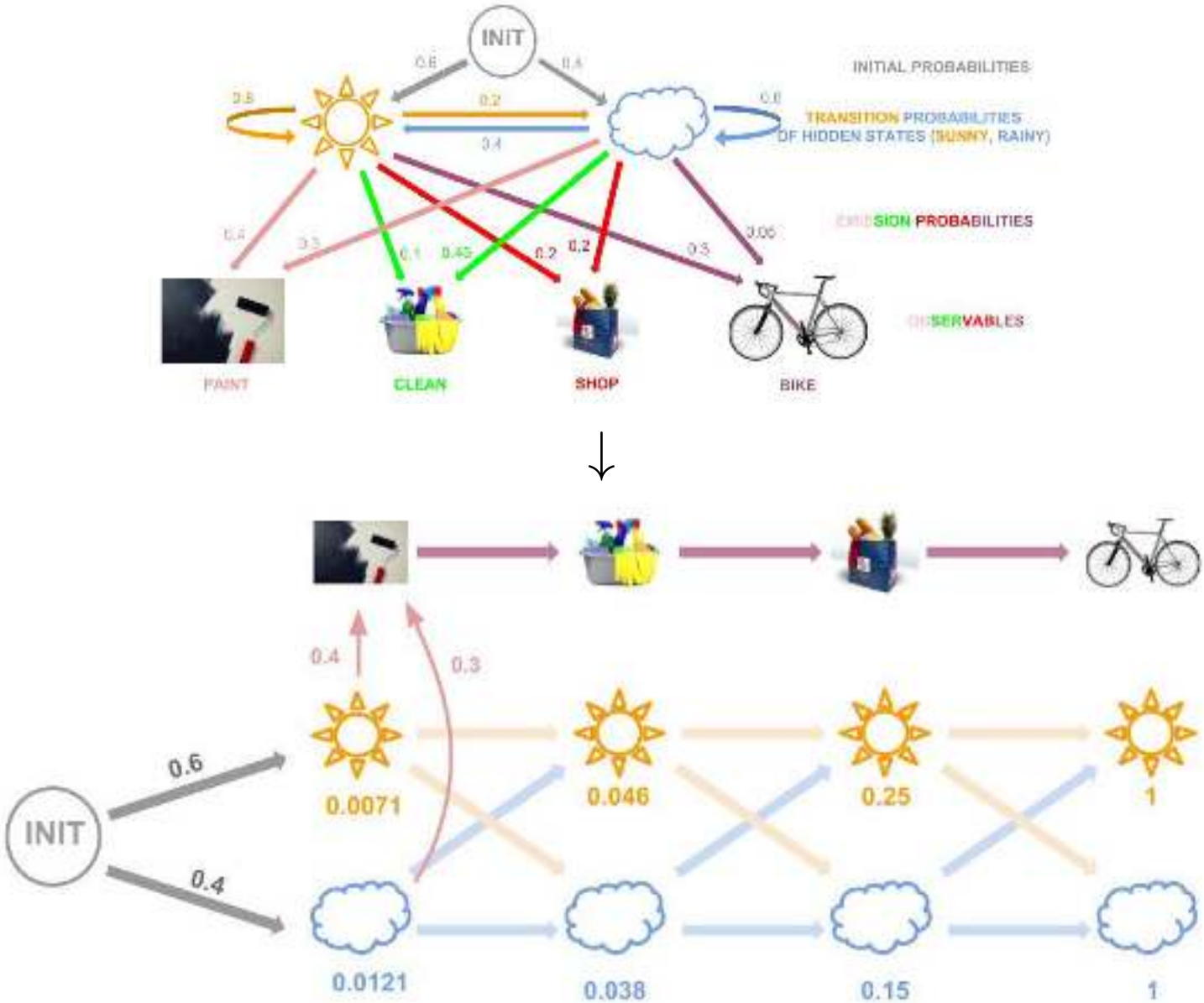
$$\forall\ n \quad P(\mathrm{k}_n = i | \mathrm{k}_{n-1} = j) = P_{ij}, \quad i, j = 1, 2, \ldots, K$$

Example

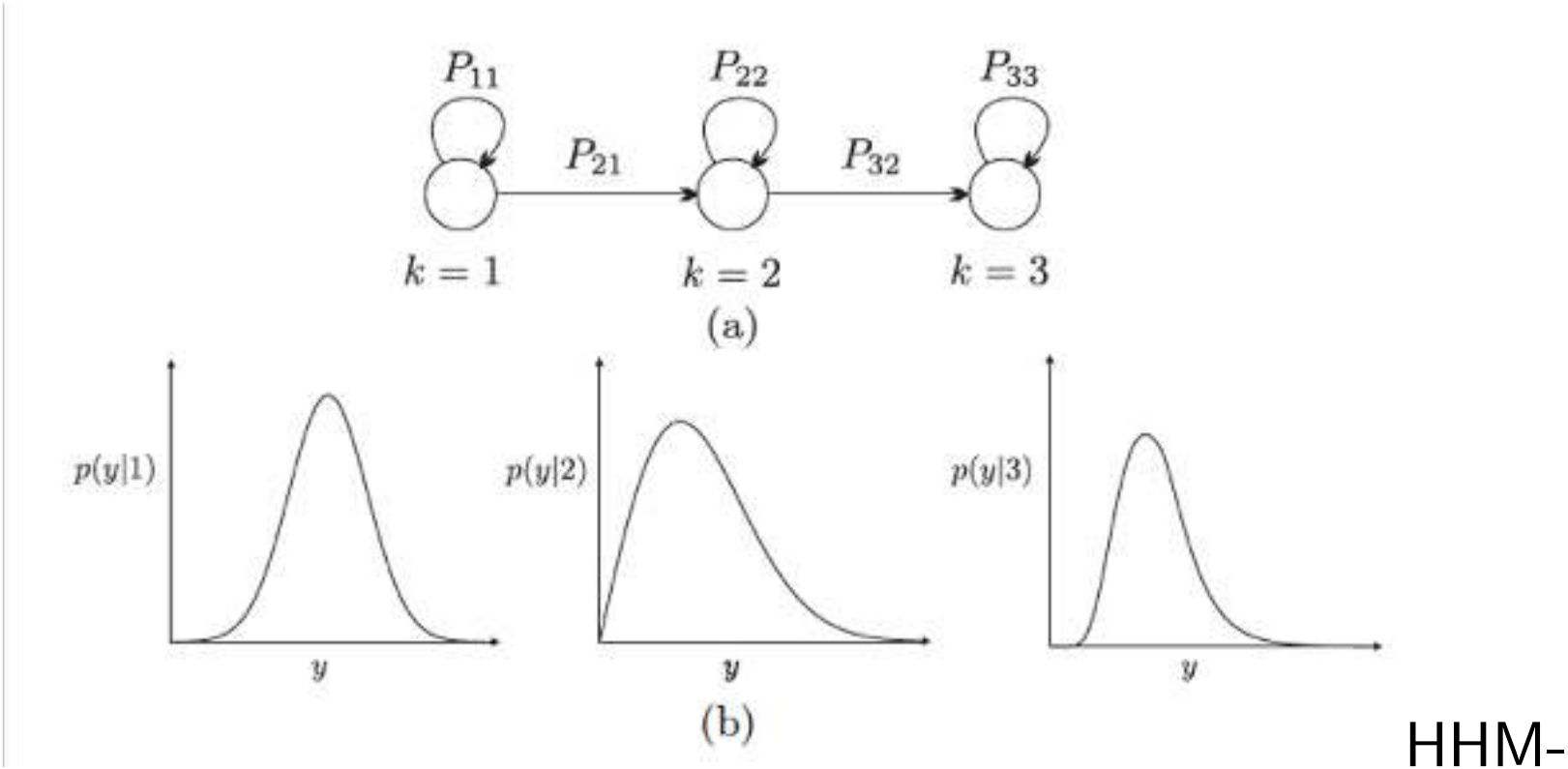


- The weather is the state $k \in \{1, 2\}$ $1 \equiv$ rainy and $2 \equiv$ sunny, the observation is the activity.

Example


INIT
INITIAL PROBABILITIES
TRANSITION PROBABILITIES
OF HIDDEN STATES (SUNNY, RAINY)
SION PROBABILITIES
SERVABLES
PAINT
CLEAN
SHOP
BIKE
0.4
0.3
0.6
0.4
0.0071
0.046
0.25
1
0.0121
0.038
0.15
1

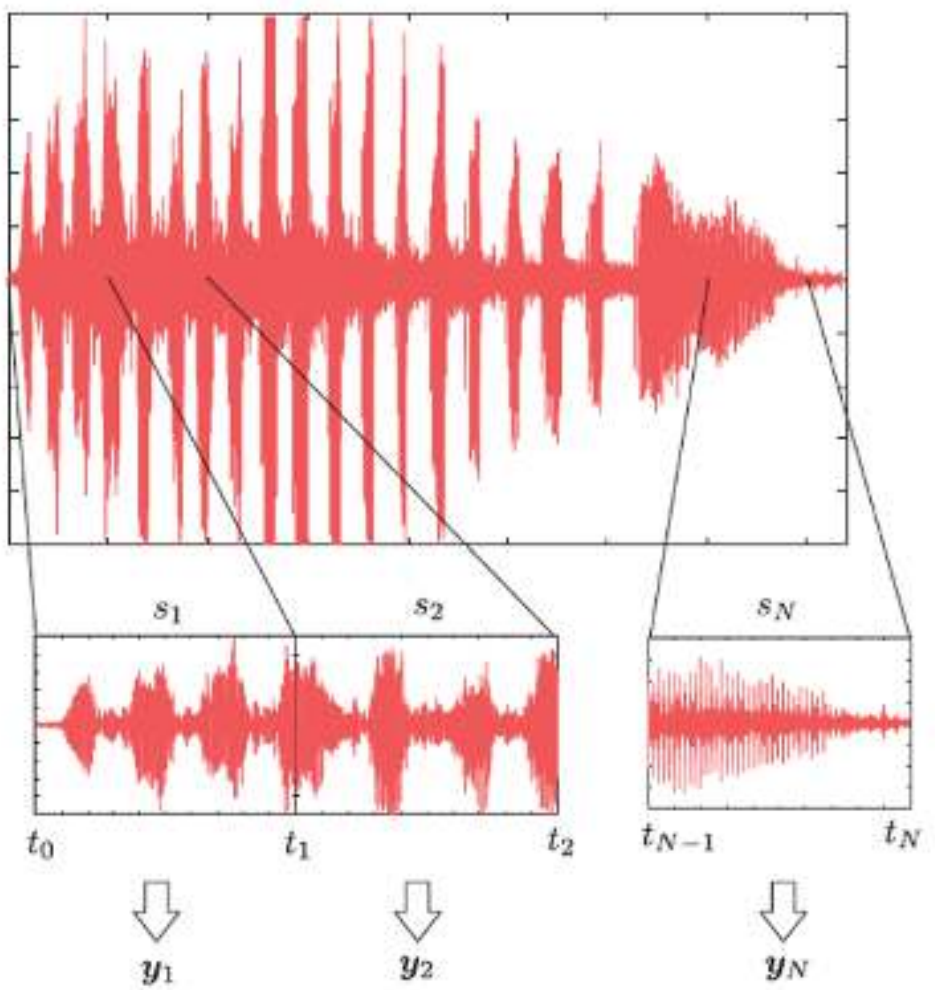
Emission probabilities



HHM-example.png

- (a) A three-state left-to-right HMM model. (b) Each state emission is a sample characterized by a different conditional density function $p(y|k)$.

Example: Speech



A speech segment and N time windows, each one of length equal to 500 ms. They correspond to time intervals [0, 500], [500, 1000], and [3500, 4000], respectively. From each one of them, a feature vector, $\boldsymbol{y}_n$, is generated (overlap between successive windows is actually allowed).

Inference for HMM

- The following parameters are available:
  1. Number of states K and set of transition probabilities $P_{ij}, i, j = 1, 2, \ldots, K$ of going from state j to state i.
  2. The probabilities for the initial state at $n = 1$ to be at state k, that is, $P_k, k = 1, 2, \ldots, K$.
  3. The state emission distributions $p(\boldsymbol{y}|k), k = 1, 2, \ldots, K$ which can either be discrete or continuous (for discrete observations this is called the Emission Probability Matrix) . These probability distributions may be parameterized, $p(\boldsymbol{y}|k; \boldsymbol{\theta}_k), k = 1, 2, \ldots, K$.

The HMM distribution

- It is useful to define a random vector $\boldsymbol{x}_n \in \{0,1\}^K$ such that:

$$[\boldsymbol{x}_n]_i = x_{n,i} = \begin{cases} 1 & \text{if } k_n = i \\ 0 & \text{else} \end{cases}$$

so that we can write the state probability as follows:

$$P(\boldsymbol{x}_1) = \prod_{k=1}^{K} P_k^{x_{1,k}}, \qquad P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}) = \prod_{i=1}^{K}\prod_{j=1}^{K} P_{ij}^{x_{n-1,j}x_{n,i}}$$

$$p(\boldsymbol{y}_n|\boldsymbol{x}_n) = \prod_{i=1}^{K} (p(\boldsymbol{y}_n|k;\boldsymbol{\theta}_k))^{x_{n,k}}$$

- Also, let $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]$ and $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$:

$$p(Y,X) = P(\boldsymbol{x}_1)p(\boldsymbol{y}_1|\boldsymbol{x}_1)\prod_{n=2}^{N} P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n)$$

The HMM distribution

- It is useful to define a random vector $\boldsymbol{x}_n \in \{0,1\}^K$ such that:

$$[\boldsymbol{x}_n]_i = x_{n,i} = \begin{cases} 1 & \text{if } k_n = i \\ 0 & \text{else} \end{cases}$$

so that we can write the state probability as follows:

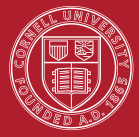
$$P(\boldsymbol{x}_1) = \prod_{k=1}^{K} P_k^{x_{1,k}}, \qquad P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}) = \prod_{i=1}^{K}\prod_{j=1}^{K} P_{ij}^{x_{n-1,j}x_{n,i}}$$

$$p(\boldsymbol{y}_n|\boldsymbol{x}_n) = \prod_{i=1}^{K} (p(\boldsymbol{y}_n|k;\boldsymbol{\theta}_k))^{x_{n,k}}$$

- Also, let $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]$ and $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$:

$$p(Y,X) = P(\boldsymbol{x}_1)p(\boldsymbol{y}_1|\boldsymbol{x}_1)\prod_{n=2}^{N} P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n)$$

Factor graph for HMM

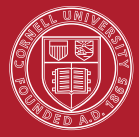


- The HMM is a BN and it yields a tree, so is safely mapped onto a factor graph to perform inferences, such as: what is the most likely state at time n?
- The MAP estimate of $\boldsymbol{x}_n$ having seen $(\boldsymbol{y}_1, \ldots \boldsymbol{y}_n)$ is the max. of:

$$\max_{\boldsymbol{x}_n} P(\boldsymbol{x}_n | \boldsymbol{y}_1, \ldots \boldsymbol{y}_n) = \frac{1}{Z} \alpha(\boldsymbol{x}_n), \quad \text{where} \quad \alpha(\boldsymbol{x}_n) := p(\boldsymbol{y}_1, \ldots \boldsymbol{y}_n, \boldsymbol{x}_n)$$

and $Z = p(\boldsymbol{y}_1, \ldots \boldsymbol{y}_n) = \sum_{\boldsymbol{x}_n} \alpha(\boldsymbol{x}_n)$

Factor graph for HMM

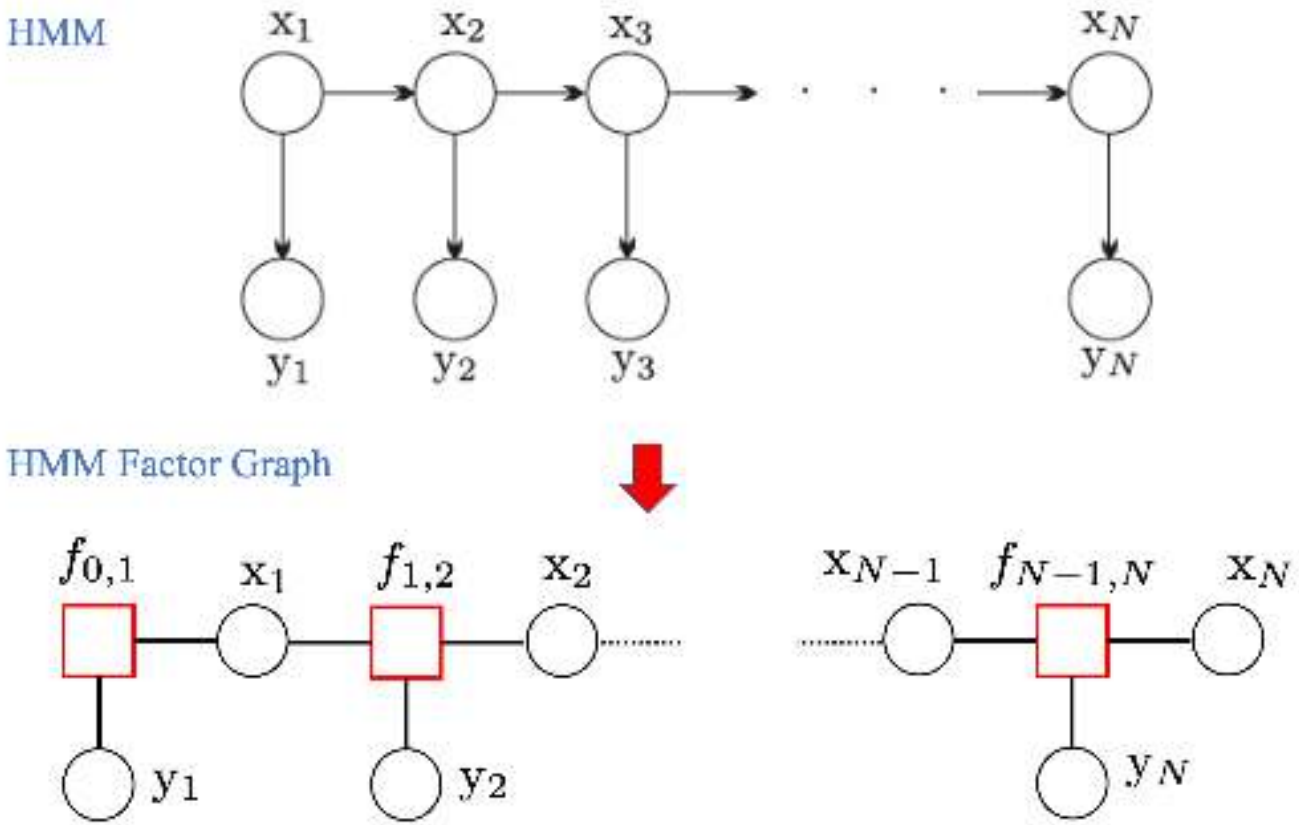


- The HMM is a BN and it yields a tree, so is safely mapped onto a factor graph to perform inferences, such as: what is the most likely state at time n?
- The MAP estimate of $\boldsymbol{x}_n$ having seen $(\boldsymbol{y}_1, \dots \boldsymbol{y}_n)$ is the max. of:

$$\max_{\boldsymbol{x}_n} P(\boldsymbol{x}_n | \boldsymbol{y}_1, \dots \boldsymbol{y}_n) = \frac{1}{Z} \alpha(\boldsymbol{x}_n), \quad \text{where} \quad \alpha(\boldsymbol{x}_n) := p(\boldsymbol{y}_1, \dots \boldsymbol{y}_n, \boldsymbol{x}_n)$$

and $Z = p(\boldsymbol{y}_1, \dots \boldsymbol{y}_n) = \sum_{\boldsymbol{x}_n} \alpha(\boldsymbol{x}_n)$

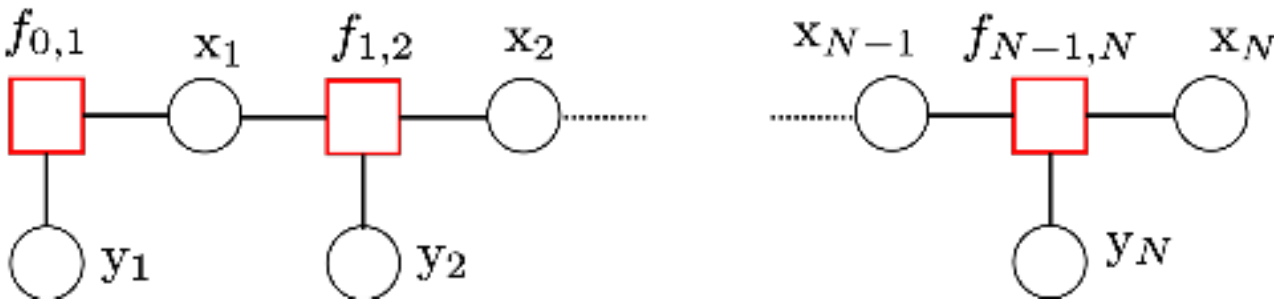
The HMM factor graph



- If we adopt the convention: $f_{0,1}(y_1, x_1) = P(x_1)p(y_1|x_1)$, and

$$f_{n-1,n}(x_{n-1}, y_n, x_n) = p(x_n|x_{n-1})p(y_n|x_n) \quad n = 2, \ldots, N$$

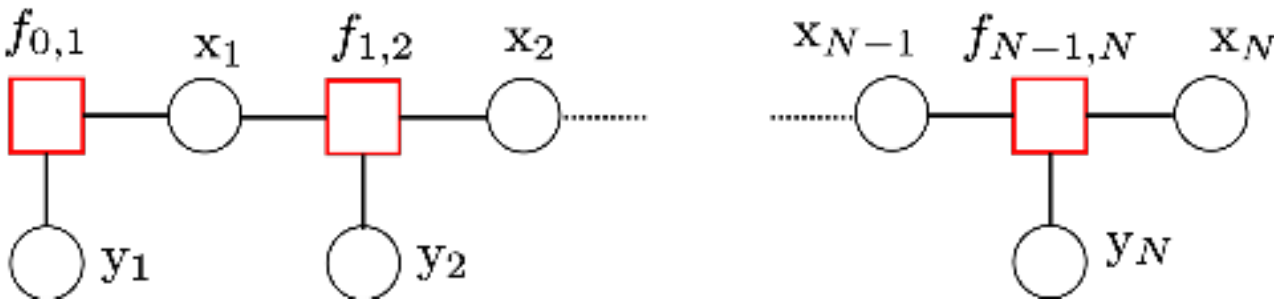
The Sum-Product Algorithm: The HMM Case



- The $\mathbf{y}_n$ are all instantiated and we are conditioning on their values $\boldsymbol{y}_n$ for $n = 1, \ldots, N$
- For each leaf $\mathbf{y}_n$ the message is $\mu_{\boldsymbol{y}_n \to f_{n-1,n}}(\boldsymbol{y}_n) = 1$
- The messages from the hidden states $\boldsymbol{x}_n$ to the factors are

$$\mu_{\boldsymbol{x}_1 \to f_{1,2}}(\boldsymbol{x}_1) = \overbrace{f_{0,1}(\boldsymbol{x}_1, \boldsymbol{y}_1)}^{=P(\boldsymbol{x}_1)p(\boldsymbol{y}_1|\boldsymbol{x}_1)}, \quad \overbrace{\mu_{\boldsymbol{x}_n \to f_{n,n+1}}(\boldsymbol{x}_n)}^{=\alpha(\boldsymbol{x}_n)} = \mu_{f_{n-1,n} \to \boldsymbol{x}_n}(\boldsymbol{x}_n), \quad n > 1$$

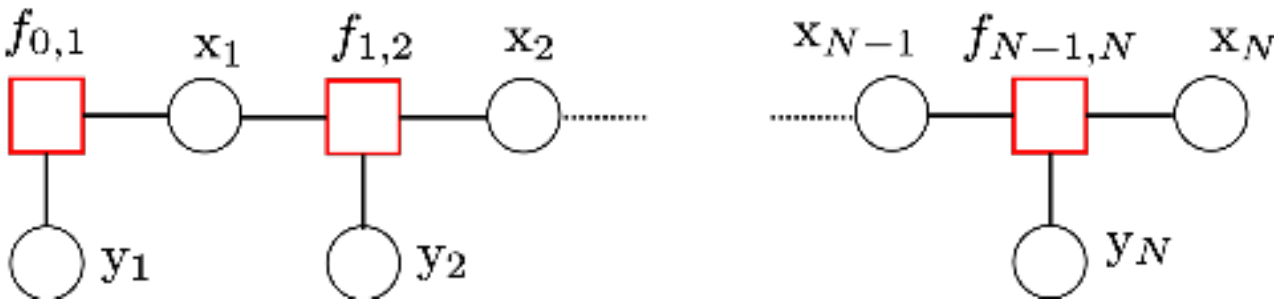
The Sum-Product Algorithm: The HMM Case



- The $\mathbf{y}_n$ are all instantiated and we are conditioning on their values $\boldsymbol{y}_n$ for $n = 1, \ldots, N$
- For each leaf $\mathbf{y}_n$ the message is $\mu_{\boldsymbol{y}_n \to f_{n-1,n}}(\boldsymbol{y}_n) = 1$
- The messages from the hidden states $\boldsymbol{x}_n$ to the factors are

$$\mu_{\boldsymbol{x}_1 \to f_{1,2}}(\boldsymbol{x}_1) = \overbrace{f_{0,1}(\boldsymbol{x}_1, \boldsymbol{y}_1)}^{=P(\boldsymbol{x}_1)p(\boldsymbol{y}_1|\boldsymbol{x}_1)}, \quad \overbrace{\mu_{\boldsymbol{x}_n \to f_{n,n+1}}(\boldsymbol{x}_n)}^{=\alpha(\boldsymbol{x}_n)} = \mu_{f_{n-1,n} \to \boldsymbol{x}_n}(\boldsymbol{x}_n), \quad n > 1$$

The Sum-Product Algorithm: The HMM Case



- The $\mathbf{y}_n$ are all instantiated and we are conditioning on their values $\boldsymbol{y}_n$ for $n = 1, \ldots, N$
- For each leaf $\mathbf{y}_n$ the message is $\mu_{\boldsymbol{y}_n \to f_{n-1,n}}(\boldsymbol{y}_n) = 1$
- The messages from the hidden states $\boldsymbol{x}_n$ to the factors are

$$\mu_{\boldsymbol{x}_1 \to f_{1,2}}(\boldsymbol{x}_1) = \overbrace{f_{0,1}(\boldsymbol{x}_1, \boldsymbol{y}_1)}^{=P(\boldsymbol{x}_1)p(\boldsymbol{y}_1|\boldsymbol{x}_1)}, \quad \overbrace{\mu_{\boldsymbol{x}_n \to f_{n,n+1}}(\boldsymbol{x}_n)}^{=\alpha(\boldsymbol{x}_n)} = \mu_{f_{n-1,n} \to \boldsymbol{x}_n}(\boldsymbol{x}_n), \quad n > 1$$

The Sum-Product Algorithm: The HMM Case

The recursion for the computation of $P(\boldsymbol{x}_n|\boldsymbol{y}_1,\dots\boldsymbol{y}_n) = \frac{1}{Z}\alpha(\boldsymbol{x}_n)$ to get $\alpha(\boldsymbol{x}_n)$ is:

$$\mu_{f_{n-1,n}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) = \sum_{\boldsymbol{x}_{n-1}} f_{n-1,n}(\boldsymbol{x}_{n-1},\boldsymbol{y}_n,\boldsymbol{x}_n)\mu_{\boldsymbol{x}_{n-1}\to f_{n-1,n}}(\boldsymbol{x}_{n-1})\mu_{\boldsymbol{y}_n\to f_{n-1,n}}(\boldsymbol{y}_n)$$

- The likelihood $\alpha(\boldsymbol{x}_n) := p(\boldsymbol{y}_1,\dots\boldsymbol{y}_n,\boldsymbol{x}_n)$ of the state and the observations up to time n is:

$$\overbrace{\mu_{f_{n-1,n}\to\boldsymbol{x}_n}(\boldsymbol{x}_n)}^{=\alpha(\boldsymbol{x}_n)} = \sum_{\boldsymbol{x}_{n-1}} \overbrace{f_{n-1,n}(\boldsymbol{x}_{n-1},\boldsymbol{y}_n,\boldsymbol{x}_n)}^{=P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n)} \mu_{\boldsymbol{x}_{n-11}\to f_{n-1,n}}(\boldsymbol{x}_{n-1}) \overbrace{\mu_{\boldsymbol{y}_n\to f_{n-1,n}}(\boldsymbol{y}_n)}^{=1}$$

$$= \sum_{\boldsymbol{x}_{n-1}} \overbrace{f_{n-1,n}(\boldsymbol{x}_{n-1},\boldsymbol{y}_n,\boldsymbol{x}_n)}^{=P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n)} \overbrace{\mu_{f_{n-2,n-1}\to\boldsymbol{x}_{n-1}}(\boldsymbol{x}_{n-1})}^{=\alpha(\boldsymbol{x}_{n-1})}$$

$$= p(\boldsymbol{y}_n|\boldsymbol{x}_n) \sum_{\boldsymbol{x}_{n-1}} P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})\alpha(\boldsymbol{x}_{n-1})$$

The Sum-Product Algorithm: The HMM Case

The recursion for the computation of $P(\boldsymbol{x}_n|\boldsymbol{y}_1,\dots\boldsymbol{y}_n) = \frac{1}{Z}\alpha(\boldsymbol{x}_n)$ to get $\alpha(\boldsymbol{x}_n)$ is:

$$\mu_{f_{n-1,n}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) = \sum_{\boldsymbol{x}_{n-1}} f_{n-1,n}(\boldsymbol{x}_{n-1},\boldsymbol{y}_n,\boldsymbol{x}_n)\mu_{\boldsymbol{x}_{n-1}\to f_{n-1,n}}(\boldsymbol{x}_{n-1})\mu_{\boldsymbol{y}_n\to f_{n-1,n}}(\boldsymbol{y}_n)$$

- The likelihood $\alpha(\boldsymbol{x}_n) := p(\boldsymbol{y}_1,\dots\boldsymbol{y}_n,\boldsymbol{x}_n)$ of the state and the observations up to time n is:

$$\overbrace{\mu_{f_{n-1,n}\to\boldsymbol{x}_n}(\boldsymbol{x}_n)}^{=\alpha(\boldsymbol{x}_n)} = \sum_{\boldsymbol{x}_{n-1}} \overbrace{f_{n-1,n}(\boldsymbol{x}_{n-1},\boldsymbol{y}_n,\boldsymbol{x}_n)}^{=P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n)} \mu_{\boldsymbol{x}_{n-11}\to f_{n-1,n}}(\boldsymbol{x}_{n-1}) \overbrace{\mu_{\boldsymbol{y}_n\to f_{n-1,n}}(\boldsymbol{y}_n)}^{=1}$$

$$= \sum_{\boldsymbol{x}_{n-1}} \overbrace{f_{n-1,n}(\boldsymbol{x}_{n-1},\boldsymbol{y}_n,\boldsymbol{x}_n)}^{=P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n)} \overbrace{\mu_{f_{n-2,n-1}\to\boldsymbol{x}_{n-1}}(\boldsymbol{x}_{n-1})}^{=\alpha(\boldsymbol{x}_{n-1})}$$

$$= p(\boldsymbol{y}_n|\boldsymbol{x}_n) \sum_{\boldsymbol{x}_{n-1}} P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})\alpha(\boldsymbol{x}_{n-1})$$

The Sum-Product Algorithm: The HMM Case

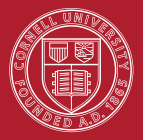
- To get $p(\boldsymbol{x}_n|Y) = \frac{\alpha(\boldsymbol{x}_n)\beta(\boldsymbol{x}_n)}{p(Y)}$ we need to get $\beta(\boldsymbol{x}_n) := p(\boldsymbol{y}_{n+1}, \dots \boldsymbol{y}_N|\boldsymbol{x}_n)$ by traversing the chain from node $\boldsymbol{y}_N$ backwards:

$$\overbrace{\mu_{f_{n,n+1}\to\boldsymbol{x}_n}(\boldsymbol{x}_n)}^{=\beta(\boldsymbol{x}_n)} = \sum_{\boldsymbol{x}_{n+1}} \overbrace{f_{n,n+1}(\boldsymbol{x}_n, \boldsymbol{y}_{n+1}, \boldsymbol{x}_{n+1})}^{=P(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n)p(\boldsymbol{y}_{n+1}|\boldsymbol{x}_{n+1})} \mu_{\boldsymbol{x}_{n+1}\to f_{n,n+1}}(\boldsymbol{x}_{n+1}) \overbrace{\mu_{\boldsymbol{y}_{n+1}\to f_{n,n+1}}(\boldsymbol{y}_{n+1})}^{=1}$$

$$= \sum_{\boldsymbol{x}_{n+1}} \overbrace{f_{n,n+1}(\boldsymbol{x}_n, \boldsymbol{y}_{n+1}, \boldsymbol{x}_{n+1})}^{=P(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n)p(\boldsymbol{y}_{n+1}|\boldsymbol{x}_{n+1})} \overbrace{\mu_{f_{n+1,n+2}\to\boldsymbol{x}_{n+1}}(\boldsymbol{x}_{n+1})}^{=\beta(\boldsymbol{x}_{n+1})}$$

$$= \sum_{\boldsymbol{x}_{n+1}} p(\boldsymbol{y}_{n+1}|\boldsymbol{x}_{n+1}) P(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n) \beta(\boldsymbol{x}_{n+1})$$

The Max-Product or Max-Sum



- If we substitute the summation with a maximization we have the max product algorithm that for the estimate of the entire X from the entire observation record $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)$
- To perform a MAP we have to pass messages forward and backwards until we reach each of the variable nodes $\boldsymbol{x}_n$

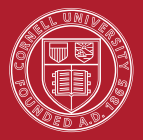
$$\mu_{f_{n-1,n}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) = \max_{\boldsymbol{x}_{n-1}} \left(f_{n-1,n}(\boldsymbol{x}_{n-1}, \boldsymbol{y}_n, \boldsymbol{x}_n) \mu_{f_{n-2,n-1}\to\boldsymbol{x}_{n-1}}(\boldsymbol{x}_{n-1}) \right)$$

$$\mu_{f_{n,n+1}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) = \max_{\boldsymbol{x}_{n+1}} \left(f_{n,n+1}(\boldsymbol{x}_n, \boldsymbol{y}_{n+1}, \boldsymbol{x}_{n+1}) \mu_{f_{n+1,n+2}\to\boldsymbol{x}_{n+1}}(\boldsymbol{x}_{n+1}) \right)$$

$$\max P(X|Y) = \max_{\boldsymbol{x}_n} \left(\mu_{f_{n-1,n}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) \mu_{f_{n,n+1}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) \right)$$

- The sequence can be found by backtracking. Note that one can clearly start from $\boldsymbol{x}_N$, i.e. the last node, to backtrack the Max-product problem.

The Max-Product or Max-Sum



- If we substitute the summation with a maximization we have the max product algorithm that for the estimate of the entire X from the entire observation record $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)$
- To perform a MAP we have to pass messages forward and backwards until we reach each of the variable nodes $\boldsymbol{x}_n$

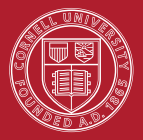
$$\mu_{f_{n-1,n}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) = \max_{\boldsymbol{x}_{n-1}} \left(f_{n-1,n}(\boldsymbol{x}_{n-1}, \boldsymbol{y}_n, \boldsymbol{x}_n)\mu_{f_{n-2,n-1}\to\boldsymbol{x}_{n-1}}(\boldsymbol{x}_{n-1}) \right)$$

$$\mu_{f_{n,n+1}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) = \max_{\boldsymbol{x}_{n+1}} \left(f_{n,n+1}(\boldsymbol{x}_n, \boldsymbol{y}_{n+1}, \boldsymbol{x}_{n+1})\mu_{f_{n+1,n+2}\to\boldsymbol{x}_{n+1}}(\boldsymbol{x}_{n+1}) \right)$$

$$\max P(X|Y) = \max_{\boldsymbol{x}_n} \left(\mu_{f_{n-1,n}\to\boldsymbol{x}_n}(\boldsymbol{x}_n)\mu_{f_{n,n+1}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) \right)$$

- The sequence can be found by backtracking. Note that one can clearly start from $\boldsymbol{x}_N$, i.e. the last node, to backtrack the Max-product problem.

The Max-Product or Max-Sum



- If we substitute the summation with a maximization we have the max product algorithm that for the estimate of the entire X from the entire observation record $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)$
- To perform a MAP we have to pass messages forward and backwards until we reach each of the variable nodes $\boldsymbol{x}_n$

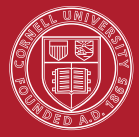
$$\mu_{f_{n-1,n}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) = \max_{\boldsymbol{x}_{n-1}}\left(f_{n-1,n}(\boldsymbol{x}_{n-1},\boldsymbol{y}_n,\boldsymbol{x}_n)\mu_{f_{n-2,n-1}\to\boldsymbol{x}_{n-1}}(\boldsymbol{x}_{n-1})\right)$$

$$\mu_{f_{n,n+1}\to\boldsymbol{x}_n}(\boldsymbol{x}_n) = \max_{\boldsymbol{x}_{n+1}}\left(f_{n,n+1}(\boldsymbol{x}_n,\boldsymbol{y}_{n+1},\boldsymbol{x}_{n+1})\mu_{f_{n+1,n+2}\to\boldsymbol{x}_{n+1}}(\boldsymbol{x}_{n+1})\right)$$

$$\max P(X|Y) = \max_{\boldsymbol{x}_n}\left(\mu_{f_{n-1,n}\to\boldsymbol{x}_n}(\boldsymbol{x}_n)\mu_{f_{n,n+1}\to\boldsymbol{x}_n}(\boldsymbol{x}_n)\right)$$

- The sequence can be found by backtracking. Note that one can clearly start from $\boldsymbol{x}_N$, i.e. the last node, to backtrack the Max-product problem.

Outline



Cornell University

Knowledge Graphs MRF

- Knowledge graphs encode the existence of facts, dependency graphs encode a statistical template for the dependencies between random A_{hrt} through the factorization in potential functions

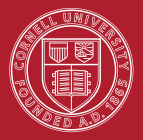
$$P(\mathbf{A}|\boldsymbol{\theta}) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi(\boldsymbol{a}_c|\boldsymbol{\theta}), \quad Z = \sum_{\forall \boldsymbol{a}_c}\prod_{c\in\mathcal{C}}\psi(\boldsymbol{a}_c|\boldsymbol{\theta})$$

- A common way of modeling what are the right cliques is to resort to what are called Markov Logic Networks and then adopt a log-linear model, where:

$$\ln\psi(\boldsymbol{a}_c|\theta_c) = \theta_c n_c(\boldsymbol{a}_c) \quad \Rightarrow P(\mathbf{A}|\boldsymbol{\theta}) = \frac{1}{Z}\prod_{c\in\mathcal{C}} e^{\theta_c n_c(\boldsymbol{a}_c)}$$

where $n_c(\boldsymbol{a}_c)$ is the number of the A_{hrt} that appear in a Markov Logic formula

Knowledge Graphs MRF



- Knowledge graphs encode the existence of facts, dependency graphs encode a statistical template for the dependencies between random A_{hrt} through the factorization in potential functions

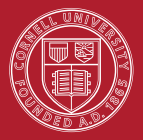
$$P(\mathbf{A}|\boldsymbol{\theta}) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi(\boldsymbol{a}_c|\boldsymbol{\theta}), \quad Z = \sum_{\forall \boldsymbol{a}_c}\prod_{c\in\mathcal{C}}\psi(\boldsymbol{a}_c|\boldsymbol{\theta})$$

- A common way of modeling what are the right cliques is to resort to what are called Markov Logic Networks and then adopt a log-linear model, where:

$$\ln\psi(\boldsymbol{a}_c|\theta_c) = \theta_c n_c(\boldsymbol{a}_c) \quad \Rightarrow P(\mathbf{A}|\boldsymbol{\theta}) = \frac{1}{Z}\prod_{c\in\mathcal{C}} e^{\theta_c n_c(\boldsymbol{a}_c)}$$

where $n_c(\boldsymbol{a}_c)$ is the number of the A_{hrt} that appear in a Markov Logic formula

Markov Logic formula



- For example these formulas:
  - "Alice is a friend of Bob"
  - "Alice is a smoker"
  - "Bob is a smoker"

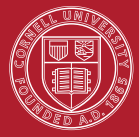
  if we believe that friends smoking habits are similar, create a clique between the corresponding facts A_{hrt}, which can be true or false
- To encode the fact that the statements are likely to be simultaneously true we choose $\theta_c > 0$ while $\theta_c < 0$ means that they are more likely to be false together
- This is just an example and other statistical templates can be adopted,

Markov Logic formula

- For example these formulas:
  - "Alice is a friend of Bob"
  - "Alice is a smoker"
  - "Bob is a smoker"

  if we believe that friends smoking habits are similar, create a clique between the corresponding facts A_{hrt}, which can be true or false
- To encode the fact that the statements are likely to be simultaneously true we choose $\theta_c > 0$ while $\theta_c < 0$ means that they are more likely to be false together
- This is just an example and other statistical templates can be adopted,

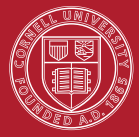
Markov Logic formula



- For example these formulas:
  - "Alice is a friend of Bob"
  - "Alice is a smoker"
  - "Bob is a smoker"

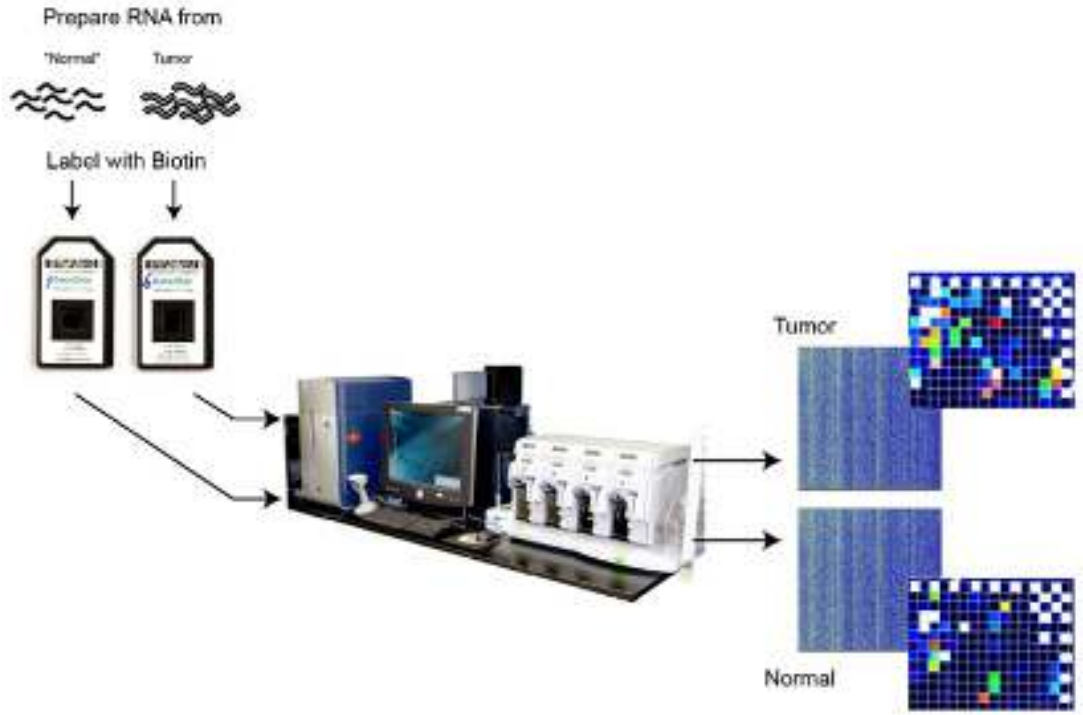
  if we believe that friends smoking habits are similar, create a clique between the corresponding facts A_{hrt}, which can be true or false
- To encode the fact that the statements are likely to be simultaneously true we choose $\theta_c > 0$ while $\theta_c < 0$ means that they are more likely to be false together
- This is just an example and other statistical templates can be adopted,

Outline



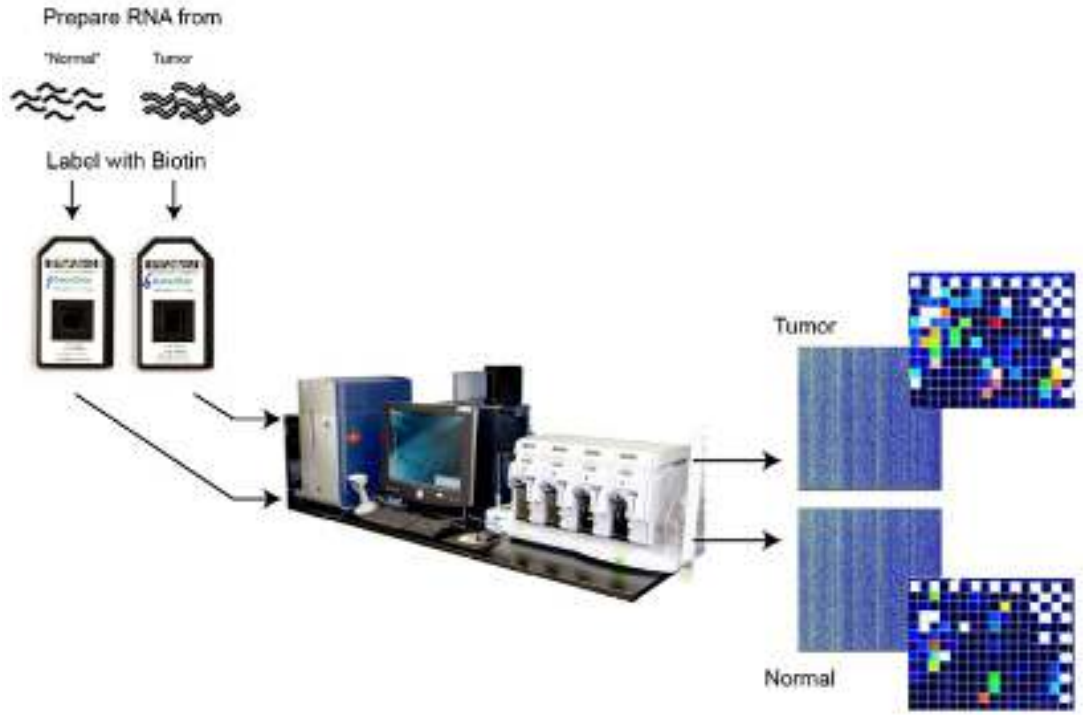
Cornell University

Genomics



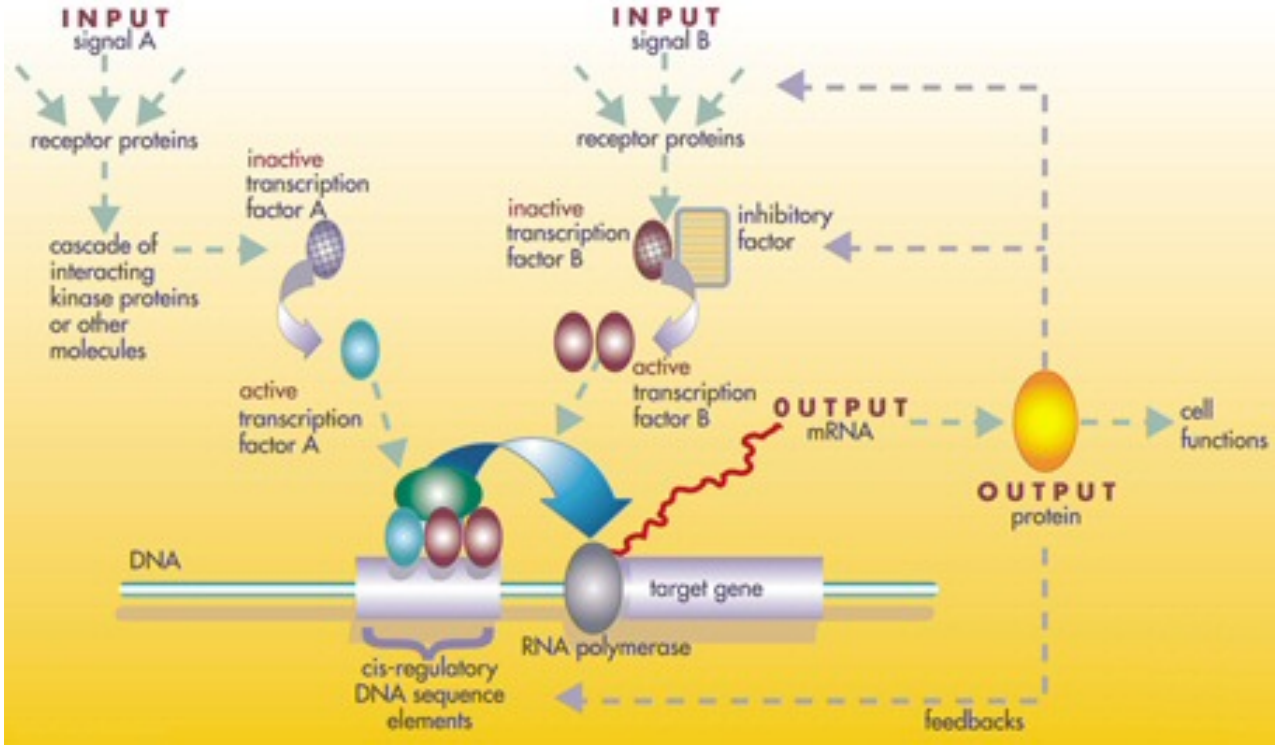
- Genomic expression arrays allow detection of DNA or RNA molecules by hybridizing or sticking to target DNA molecules or RNA molecules on a glass slide
- DNA or RNA molecule are marked by various labels through dyes that can be seen under a microscope

Genomics



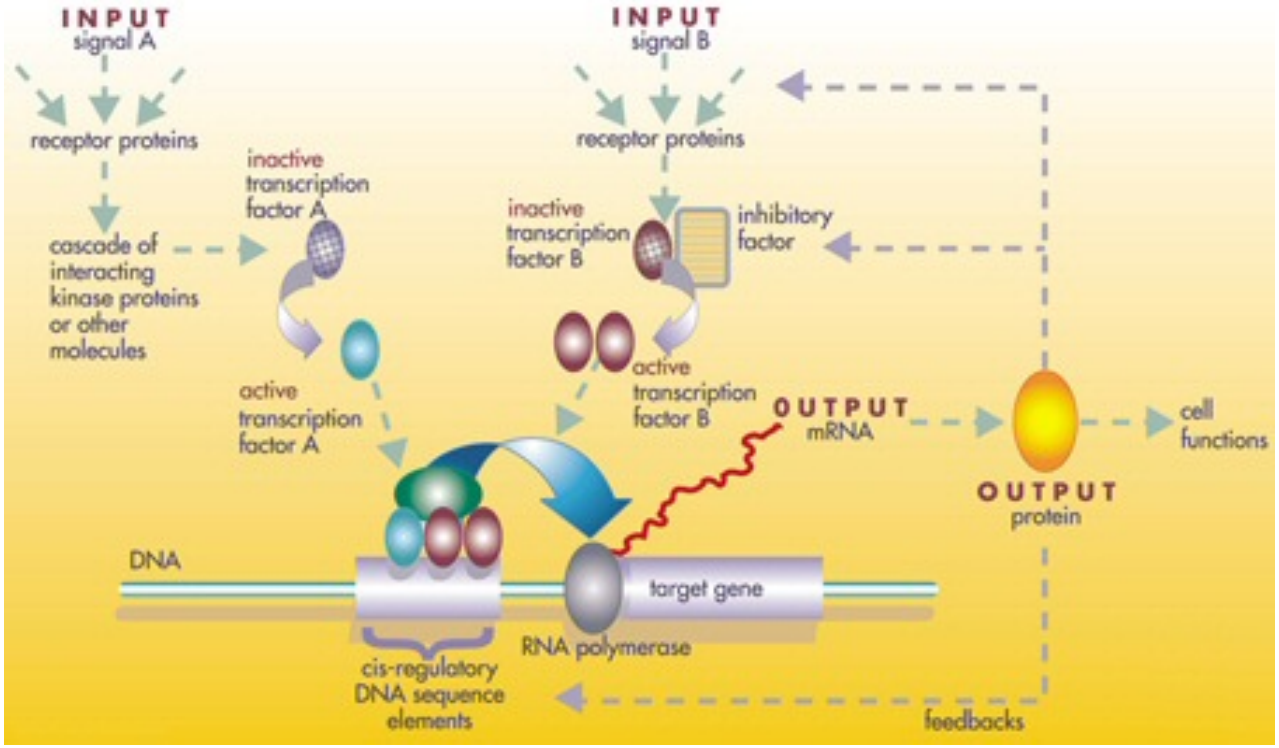
- Genomic expression arrays allow detection of DNA or RNA molecules by hybridizing or sticking to target DNA molecules or RNA molecules on a glass slide
- DNA or RNA molecule are marked by various labels through dyes that can be seen under a microscope

Gene regulatory networks



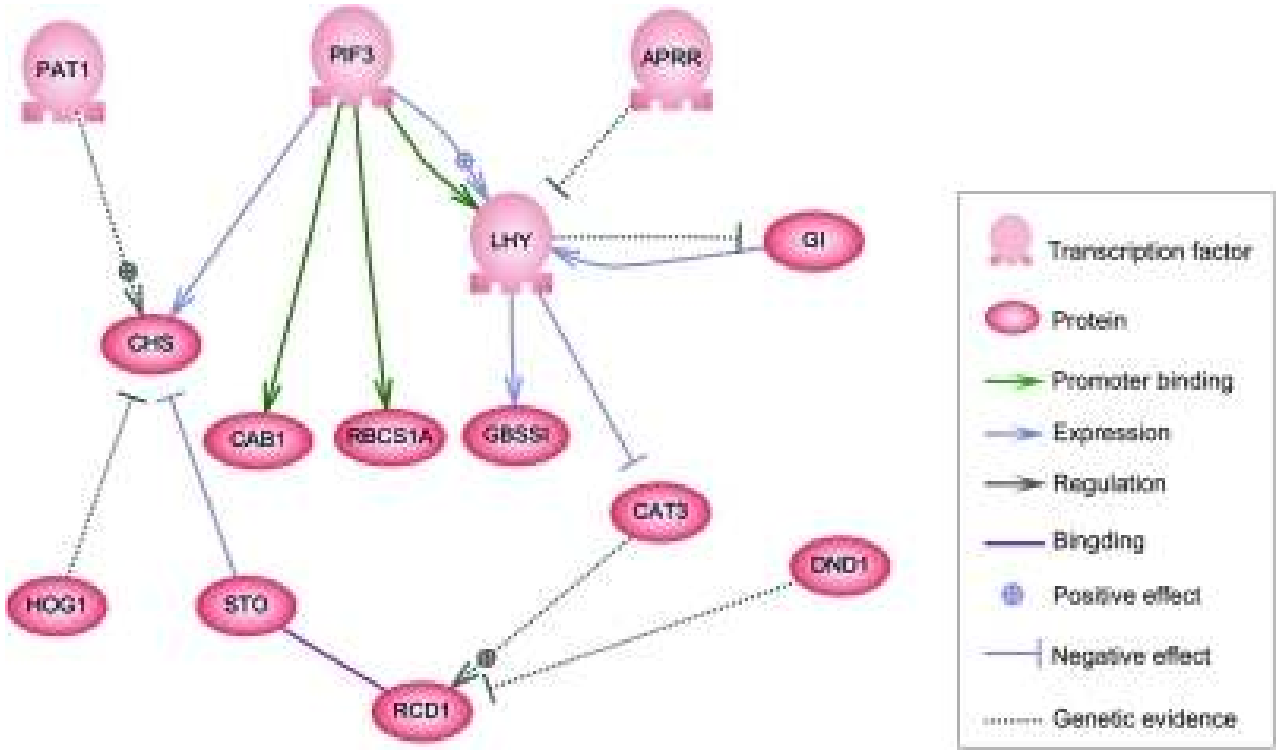
- A gene (or genetic) regulatory network (GRN) is a set of molecular regulators interacting with each other and substances in the cell.
- These interactions control the gene expression levels of mRNA and proteins which determine the function of the cell.

Gene regulatory networks



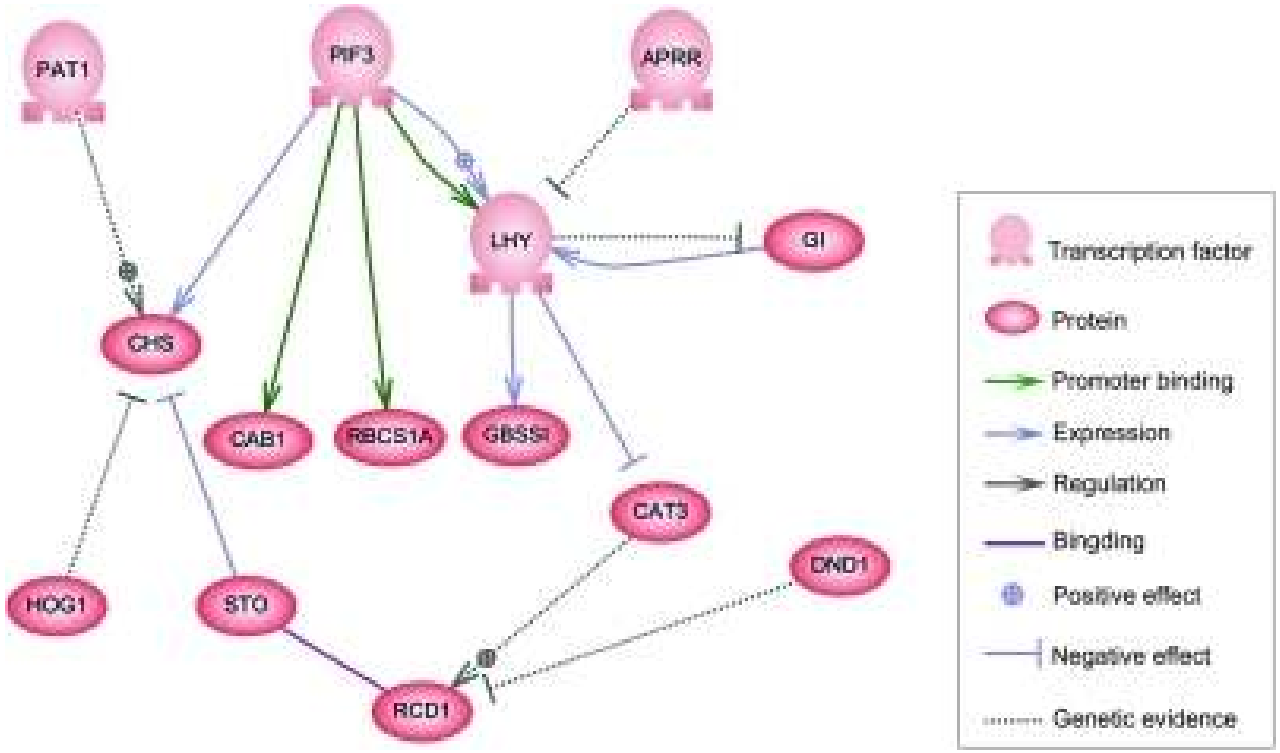
- A gene (or genetic) regulatory network (GRN) is a set of molecular regulators interacting with each other and substances in the cell.
- These interactions control the gene expression levels of mRNA and proteins which determine the function of the cell.

Gene regulatory networks



- The nodes of this network can represent genes, proteins, mRNAs, protein/protein complexes or cellular processes.
- Generally, the interactions are mostly not known, and there is an interest in inferring them.

Gene regulatory networks



- The nodes of this network can represent genes, proteins, mRNAs, protein/protein complexes or cellular processes.
- Generally, the interactions are mostly not known, and there is an interest in inferring them.

Undirected GRN inference with Gaussian Model

- Since we know that the interactions are *sparse*, this is a good candidate for a BN representation.
- A popular option to obtain an undirected graph is to use a parametric Gaussian BN that represents the interactions
- Gaussian models: $p(x_i|\text{Pa}_i) = \mathcal{N}\left(\sum_{k:x_k \in \text{Pa}_i} \theta_{ik} x_k + \theta_{i0}, \sigma_i^2\right)$ where $\theta_{ik} = [\Theta]_{ik}$ is the so called precision matrix

$$\Theta = \Sigma^{-1} \quad \Sigma = \text{covariance matrix} = \mathbb{E}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^\top]$$

- Basically we want to estimate a sparse Θ and we $\boldsymbol{\theta}_0 = \Sigma^{-1/2}\boldsymbol{\mu}$

Undirected GRN inference with Gaussian Model

- Since we know that the interactions are *sparse*, this is a good candidate for a BN representation.
- A popular option to obtain an undirected graph is to use a parametric Gaussian BN that represents the interactions
- Gaussian models: $p(x_i|\text{Pa}_i) = \mathcal{N}\left(\sum_{k:x_k \in \text{Pa}_i} \theta_{ik} x_k + \theta_{i0}, \sigma_i^2\right)$ where $\theta_{ik} = [\Theta]_{ik}$ is the so called precision matrix

$$\Theta = \Sigma^{-1} \quad \Sigma = \text{covariance matrix} = \mathbb{E}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^\top]$$

- Basically we want to estimate a sparse Θ and we $\boldsymbol{\theta}_0 = \Sigma^{-1/2}\boldsymbol{\mu}$

Undirected GRN inference with Gaussian Model

- Since we know that the interactions are *sparse*, this is a good candidate for a BN representation.
- A popular option to obtain an undirected graph is to use a parametric Gaussian BN that represents the interactions
- Gaussian models: $p(x_i|\text{Pa}_i) = \mathcal{N}\left(\sum_{k:x_k \in \text{Pa}_i} \theta_{ik} x_k + \theta_{i0}, \sigma_i^2\right)$ where $\theta_{ik} = [\boldsymbol{\Theta}]_{ik}$ is the so called precision matrix

$$\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1} \quad \boldsymbol{\Sigma} = \text{covariance matrix} = \mathbb{E}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^\top]$$

- Basically we want to estimate a sparse $\boldsymbol{\Theta}$ and we $\theta_0 = \boldsymbol{\Sigma}^{-1/2}\mu$

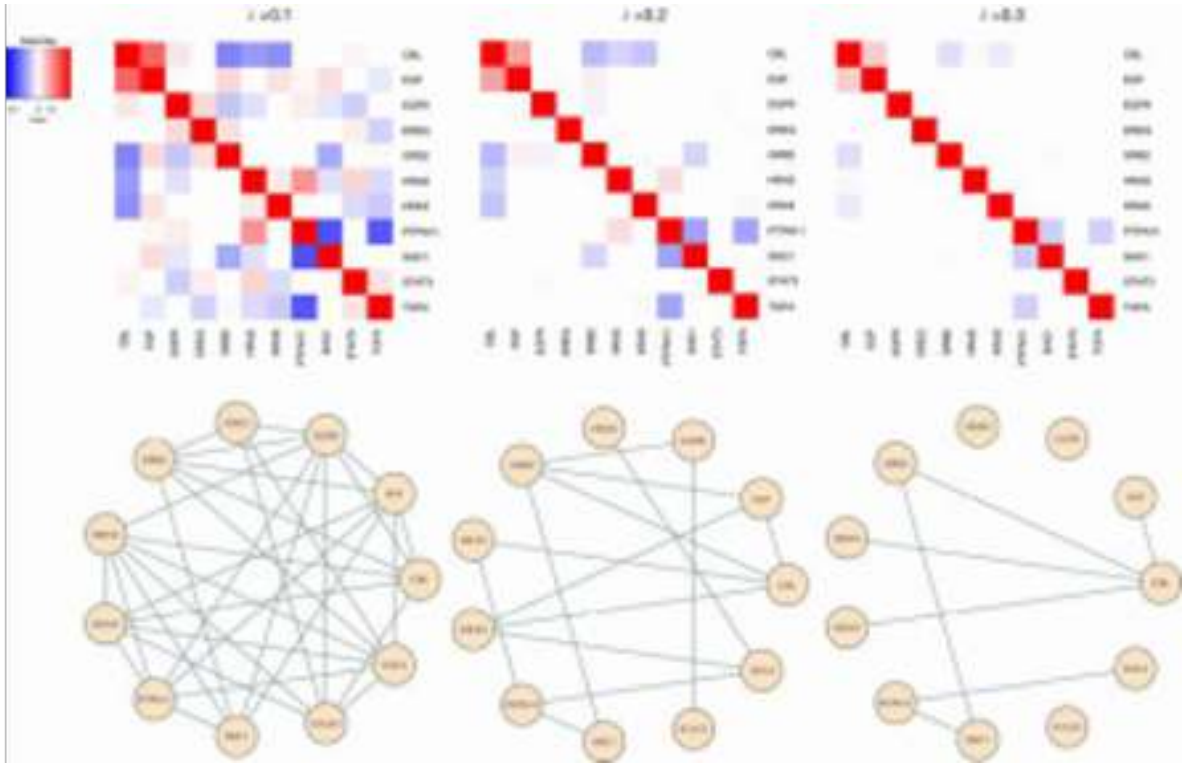
Undirected GRN inference with Gaussian Model

- Since we know that the interactions are *sparse*, this is a good candidate for a BN representation.
- A popular option to obtain an undirected graph is to use a parametric Gaussian BN that represents the interactions
- Gaussian models: $p(x_i|\text{Pa}_i) = \mathcal{N}\left(\sum_{k:x_k \in \text{Pa}_i} \theta_{ik} x_k + \theta_{i0}, \sigma_i^2\right)$ where $\theta_{ik} = [\boldsymbol{\Theta}]_{ik}$ is the so called precision matrix

$$\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1} \quad \boldsymbol{\Sigma} = \text{covariance matrix} = \mathbb{E}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^\top]$$

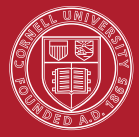
- Basically we want to estimate a sparse $\boldsymbol{\Theta}$ and we $\boldsymbol{\theta}_0 = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{\mu}$

Undirected GRN inference using Gaussian BNs



- In this case the method to used is called Graphical LASSO, where LASSO stands for Least Absolute Shrinkage and Selection Operator

Graphical LASSO

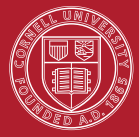


- The Graphical LASSO extimate is obtained by maximizing the penalized log likelihood with respect to the precision matrix with a sparse penalty on :

$$\hat{\boldsymbol{\Theta}} = \max_{\boldsymbol{\Theta}} \left\{ \ln \det(\boldsymbol{\Theta}) - \mathrm{Tr}(\hat{\Sigma}^{-1}\boldsymbol{\Theta}) - \lambda \|\boldsymbol{\Theta}\|_1 \right\}$$

where $\hat{\Sigma} = \sum_n (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}})$

MRF exponential model

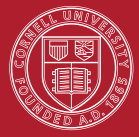


- Graphical LASSO depends on the Gaussian modeling
- Other possible models use parametric MRF models for the regulatory link variables x_ℓ.
- For example

$$P(\boldsymbol{x}) = \frac{1}{Z} e^{-E(\boldsymbol{x})} \qquad E(\boldsymbol{x}) = \sum_{\ell} f(x_\ell) + \lambda \sum_{p \in \mathcal{N}_\ell} f(x_\ell, x_p)$$

where often $f(x_\ell) \propto |x_\ell - 1|$ and $f(x_\ell, x_p) \propto |x_\ell - x_p|$

MRF exponential model

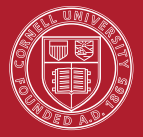


- Graphical LASSO depends on the Gaussian modeling
- Other possible models use parametric MRF models for the regulatory link variables x_ℓ.
- For example

$$P(\boldsymbol{x}) = \frac{1}{Z} e^{-E(\boldsymbol{x})} \qquad E(\boldsymbol{x}) = \sum_{\ell} f(x_\ell) + \lambda \sum_{p \in \mathcal{N}_\ell} f(x_\ell, x_p)$$

where often $f(x_\ell) \propto |x_\ell - 1|$ and $f(x_\ell, x_p) \propto |x_\ell - x_p|$

MRF exponential model

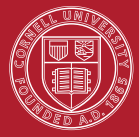


- Graphical LASSO depends on the Gaussian modeling
- Other possible models use parametric MRF models for the regulatory link variables x_ℓ.
- For example

$$P(\boldsymbol{x}) = \frac{1}{Z} e^{-E(\boldsymbol{x})} \qquad E(\boldsymbol{x}) = \sum_{\ell} f(x_\ell) + \lambda \sum_{p \in \mathcal{N}_\ell} f(x_\ell, x_p)$$

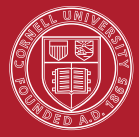
where often $f(x_\ell) \propto |x_\ell - 1|$ and $f(x_\ell, x_p) \propto |x_\ell - x_p|$

Outline



Cornell University

Conclusions



- in this lecture we explained how to perform computations with graphical models with
  1. Factor graphs
  2. Junction trees
- We also introduced a few examples of applications