Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Factor Graphs and Message Passing Algorithms — Part 1: Introduction

Hans-Andrea Loeliger

The Two Basic Problems

1. Marginalization: Compute
$$\bar{f}_k(x_k) \stackrel{\triangle}{=} \sum_{\substack{x_1, \ldots, x_n \\ \text{except } x_k}} f(x_1, \ldots, x_n)$$

2. Maximization: Compute the "max-marginal"
$$\hat{f}_k(x_k) \stackrel{\triangle}{=} \max_{\substack{x_1, \ldots, x_n \\ \text{except } x_k}} f(x_1, \ldots, x_n)$$

assuming that f is real-valued and nonnegative and has a maximum.
Note that
$$\operatorname{argmax} f(x_1, \ldots, x_n) = \left(\operatorname{argmax} \hat{f}_1(x_1), \ldots, \operatorname{argmax} \hat{f}_n(x_n)\right).$$

For large n, both problems are in general intractable
(even for $x_1, \ldots, x_n \in \{0, 1\}$).

Factorization Helps

For example, if $f(x_1, \ldots, f_n) = f_1(x_1)f_2(x_2)\cdots f_n(x_n)$ then

$$\bar{f}_k(x_k) = \sum_{x_1} f_1(x_1) \cdots \sum_{x_{k-1}} f_{k-1}(x_{k-1}) f_k(x_k) \sum_{x_{k+1}} f_{k+1}(x_{k+1}) \cdots \sum_{x_n} f_n(x_n)$$

and

$$\hat{f}_k(x_k) = \max_{x_1} f_1(x_1) \cdots \max_{x_{k-1}} f_{k-1}(x_{k-1}) f_k(x_k) \max_{x_{k+1}} f_{k+1}(x_{k+1}) \cdots \max_{x_n} f_n(x_n).$$

Factorization helps also beyond this trivial example.
$\longrightarrow$ Factor graphs and message passing algorithms.

Roots

Statistical physics:
- Markov random fields (Ising 1925?)

Signal processing:
- linear state-space models and Kalman filtering: Kalman 1960...
- recursive least-squares adaptive filters
- Hidden Markov models and forward-backward algorithm: Baum et al. 1966...

Error correcting codes:
- Low-density parity check codes: Gallager 1962; Tanner 1981; MacKay 1996; Luby et al. 1998...
- Convolutional codes and Viterbi decoding: Forney 1973...
- Turbo codes: Berrou et al. 1993...

Machine learning, statistics:
- Bayesian networks: Pearl 1988; Shachter 1988; Lauritzen and Spiegelhalter 1988; Shafer and Shenoy 1990...

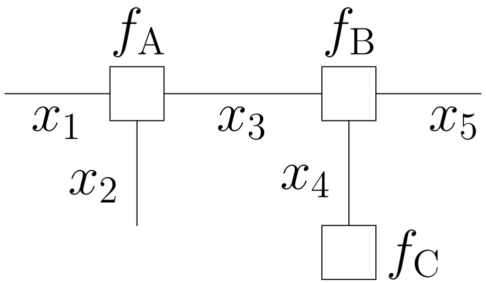
Outline of this talk

Factor Graphs

A factor graph represents the factorization of a function of several variables. We use Forney-style factor graphs (Forney, 2001).

Example:

$$f(x_1, x_2, x_3, x_4, x_5) = f_A(x_1, x_2, x_3) \cdot f_B(x_3, x_4, x_5) \cdot f_C(x_4).$$



Rules:

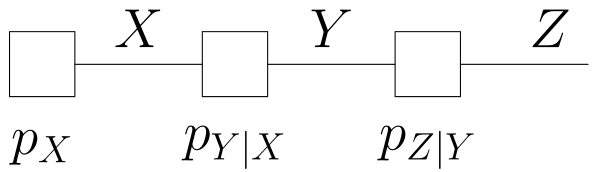
- A node for every factor.
- An edge or half-edge for every variable.
- Node g is connected to edge x iff variable x appears in factor g.

(What if some variable appears in more than 2 factors?)

A main application of factor graphs are stochastic models. Example:

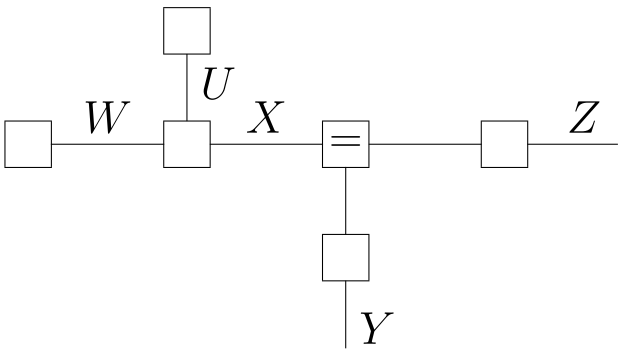
Markov Chain

$$p_{XYZ}(x, y, z) = p_X(x)\, p_{Y|X}(y|x)\, p_{Z|Y}(z|y).$$

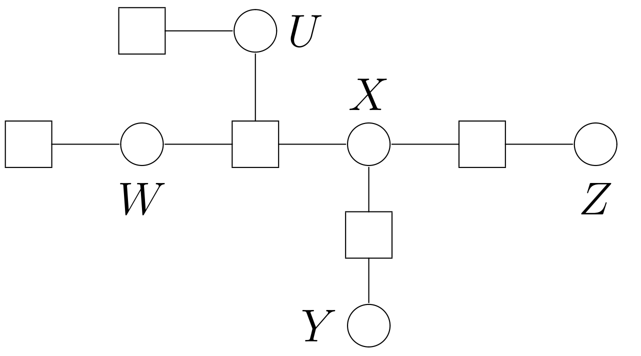


Other Notation Systems for Graphical Models

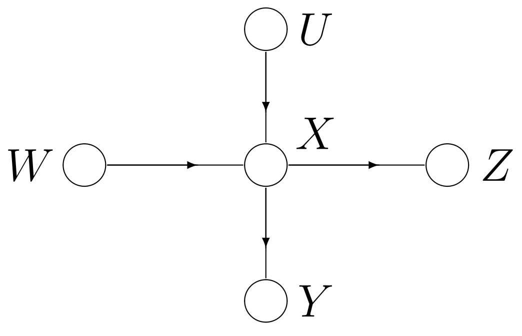
Example: $p(u, w, x, y, z) = p(u)p(w)p(x|u, w)p(y|x)p(z|x)$.



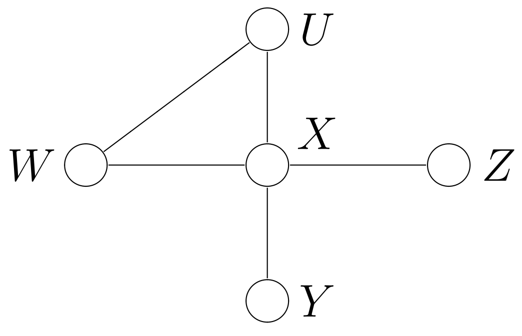
Forney-style factor graph.



Original factor graph [FKLW 1997].

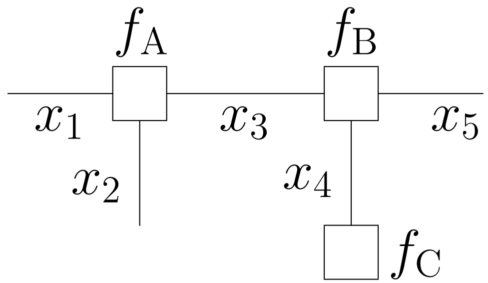


Bayesian network.



Markov random field.

Terminology



Local function $=$ factor (such as f_A, f_B, f_C).

Global function f $=$ product of all local functions; usually (but not always!) real and nonnegative.

A configuration is an assignment of values to all variables.

The configuration space is the set of all configurations, which is the domain of the global function.

A configuration $\omega = (x_1, \ldots, x_5)$ is valid iff $f(\omega) \neq 0$.

Invalid Configurations Do Not Affect Marginals

A configuration $\omega = (x_1, \ldots, x_n)$ is valid iff $f(\omega) \neq 0$.

Recall:

1. Marginalization: Compute

$$\bar{f}_k(x_k) \triangleq \sum_{\substack{x_1, \ldots, x_n \\ \text{except } x_k}} f(x_1, \ldots, x_n)$$

2. Maximization: Compute the "max-marginal"

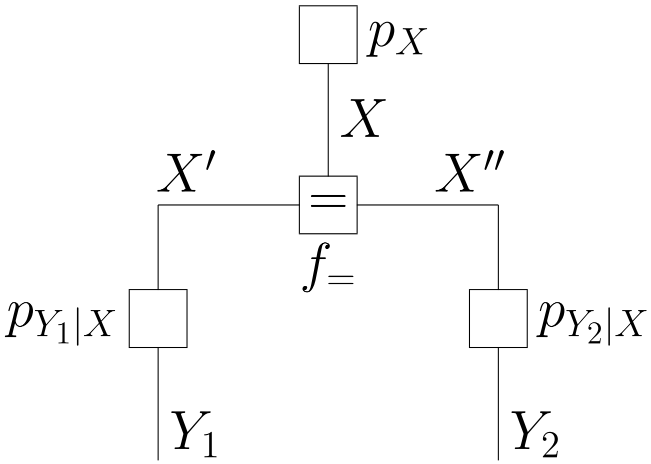
$$\hat{f}_k(x_k) \triangleq \max_{\substack{x_1, \ldots, x_n \\ \text{except } x_k}} f(x_1, \ldots, x_n)$$

assuming that f is real-valued and nonnegative and has a maximum.

Auxiliary Variables

Example: Let Y_1 and Y_2 be two independent observations of X:

$$p(x, y_1, y_2) = p(x)p(y_1|x)p(y_2|x).$$



Literally, the factor graph represents an extended model

$$p(x, x', x'', y_1, y_2) = p(x)p(y_1|x')p(y_2|x'')f_=(x, x', x'')$$

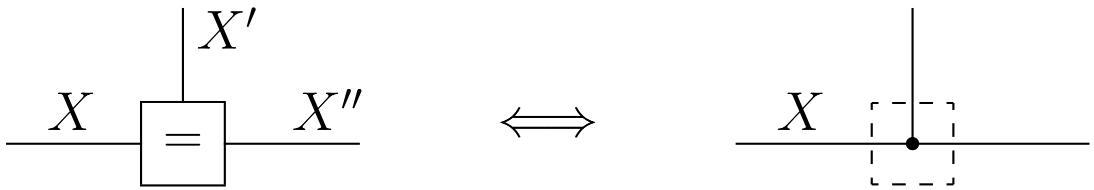
where

$$f_=(x, x', x'') \triangleq \delta(x - x')\delta(x - x'')$$

enforces $X = X' = X''$ for every valid configuration.

Branching Points

Equality constraint nodes may be viewed as branching points:



The factor

$$f_=(x, x', x'') \triangleq \delta(x - x')\delta(x - x'')$$

enforces $X = X' = X''$ for every valid configuration.

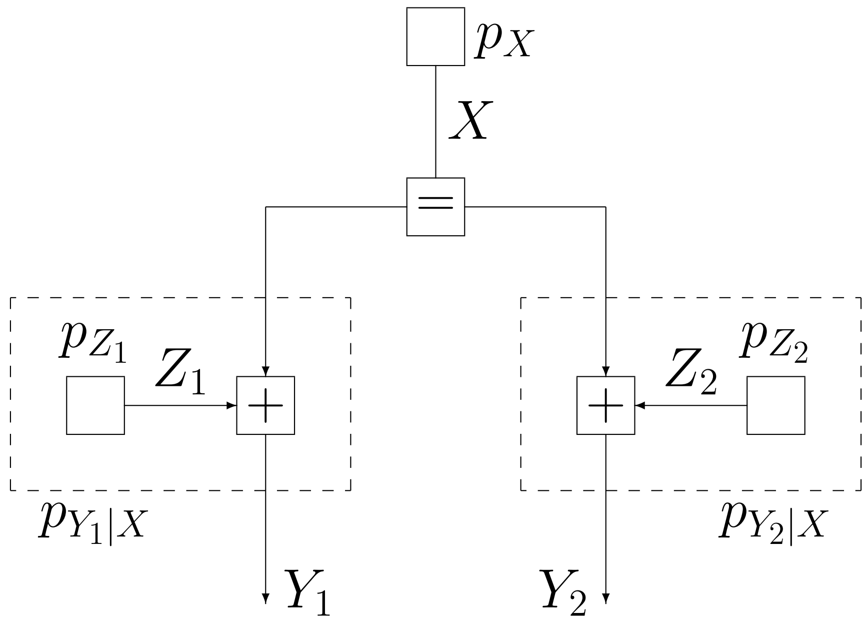
Modularity, Special Symbols, Arrows

As a refinement of the previous example, let

$$Y_1 = X + Z_1 \tag{1}$$

$$Y_2 = X + Z_2 \tag{2}$$

with Z_1 and Z_2 independent of each other and of X:

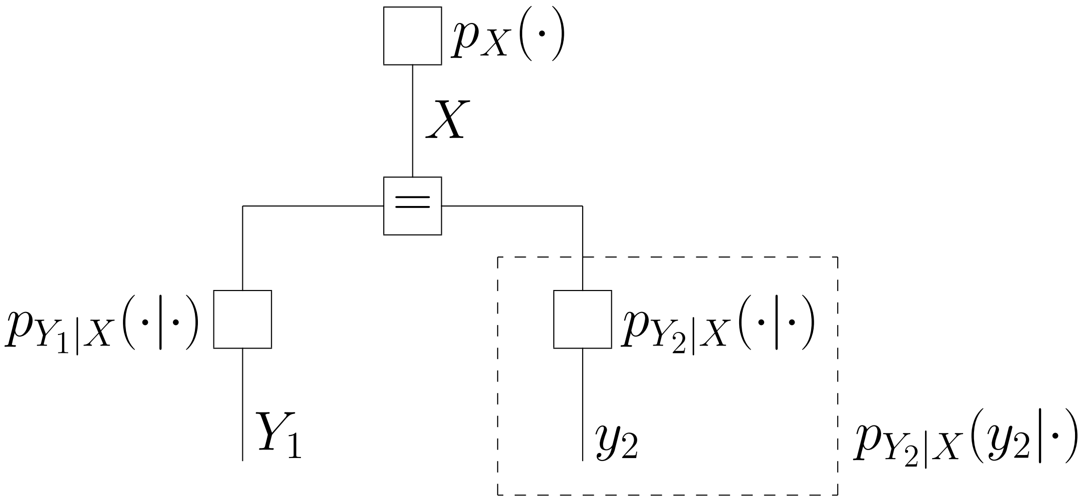


The "+"-nodes represent the factors $\delta(x+z_1-y_1)$ and $\delta(x + z_2 - y_2)$, which enforce (1) and (2) for every valid configuration.

Known Variables vs. Unknown Variables

Known variables (observations, known parameters, ...) may be plugged into the corresponding factors.

Example: $Y_2 = y_2$ observed.

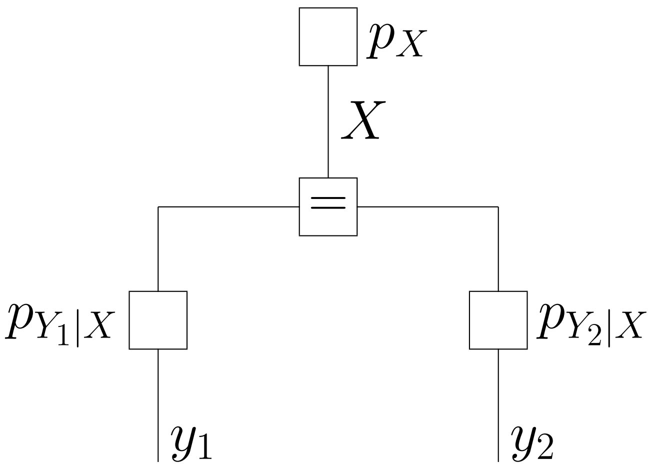


Known variables will be denoted by small letters;
unknown variables will usually be denoted by capital letters.

From A Priori to A Posteriori Probability

Example (cont'd): Let $Y_1 = y_1$ and $Y_2 = y_2$ be two independent observations of X. For fixed y_1 and y_2, we have

$$\begin{aligned} p(x|y_1, y_2) &= \frac{p(x, y_1, y_2)}{p(y_1, y_2)} \\ &\propto p(x, y_1, y_2). \end{aligned}$$

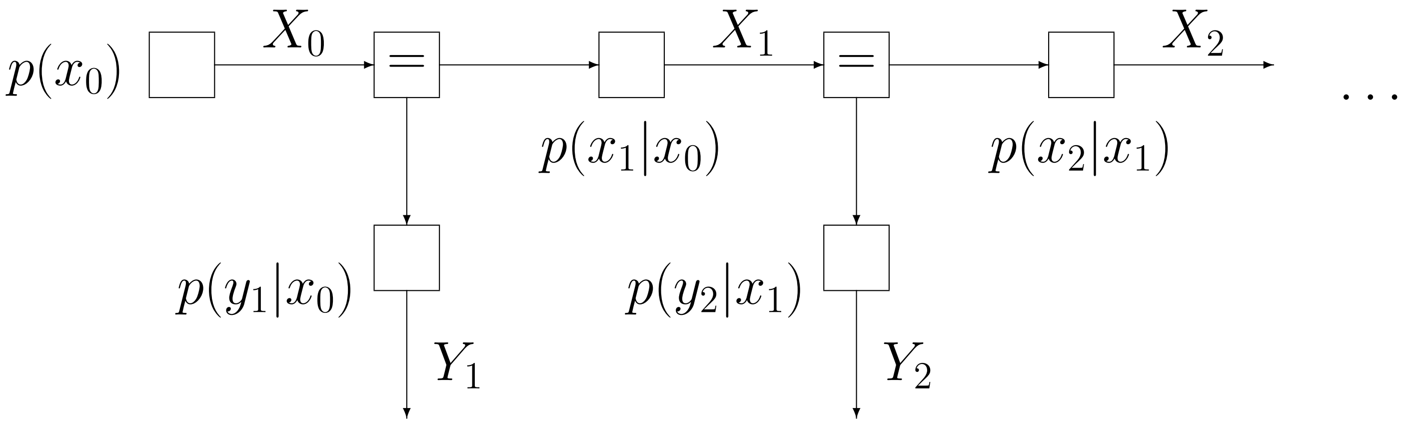


The factorization is unchanged (except for a scale factor).

Example:

Hidden Markov Model

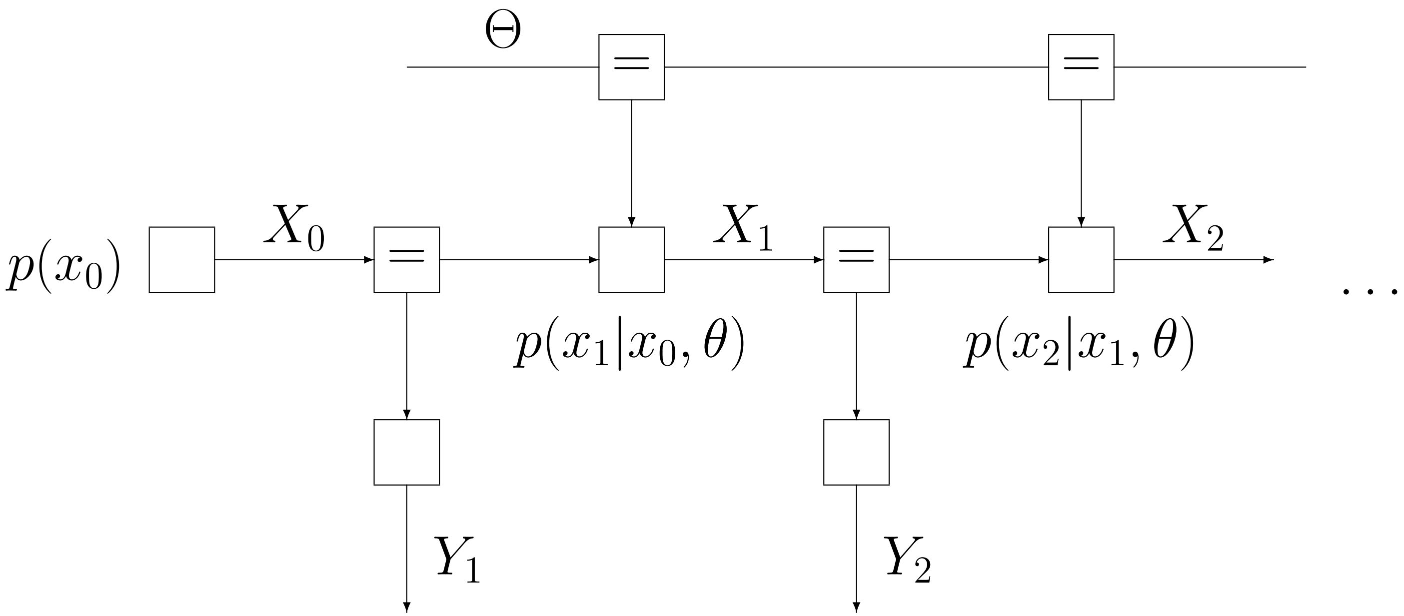
$$p(x_0, x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = p(x_0) \prod_{k=1}^{n} p(x_k|x_{k-1}) p(y_k|x_{k-1})$$



Example:

Hidden Markov Model with Parameter(s)

$$p(x_0, x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n \mid \theta) = p(x_0) \prod_{k=1}^{n} p(x_k|x_{k-1}, \theta) p(y_k|x_{k-1})$$



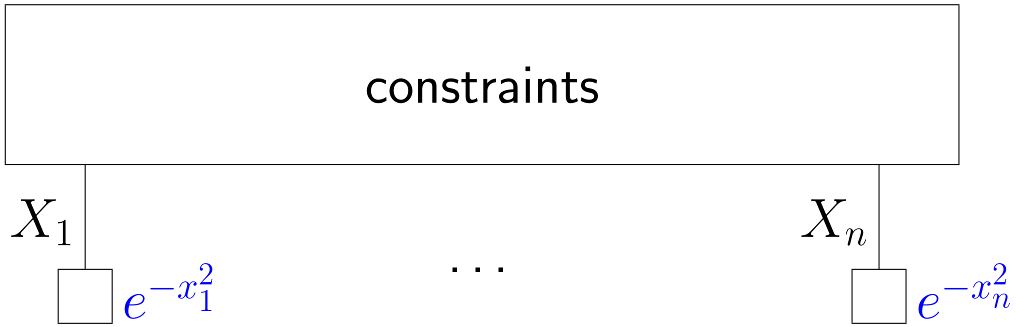
A non-stochastic example:

Least-Squares Problems

Minimizing $\sum_{k=1}^{n} x_k^2$ subject to (linear or nonlinear) constraints is equivalent to maximizing

$$e^{-\sum_{k=1}^{n} x_k^2} = \prod_{k=1}^{n} e^{-x_k^2}$$

subject to the given constraints.



Here, the factor graph represents a nonnegative real-valued function that we wish to maximize.

Outline

Recall:

The Two Basic Problems

1. Marginalization: Compute

$$\bar{f}_k(x_k) \triangleq \sum_{\substack{x_1, \ldots, x_n \\ \text{except } x_k}} f(x_1, \ldots, x_n)$$

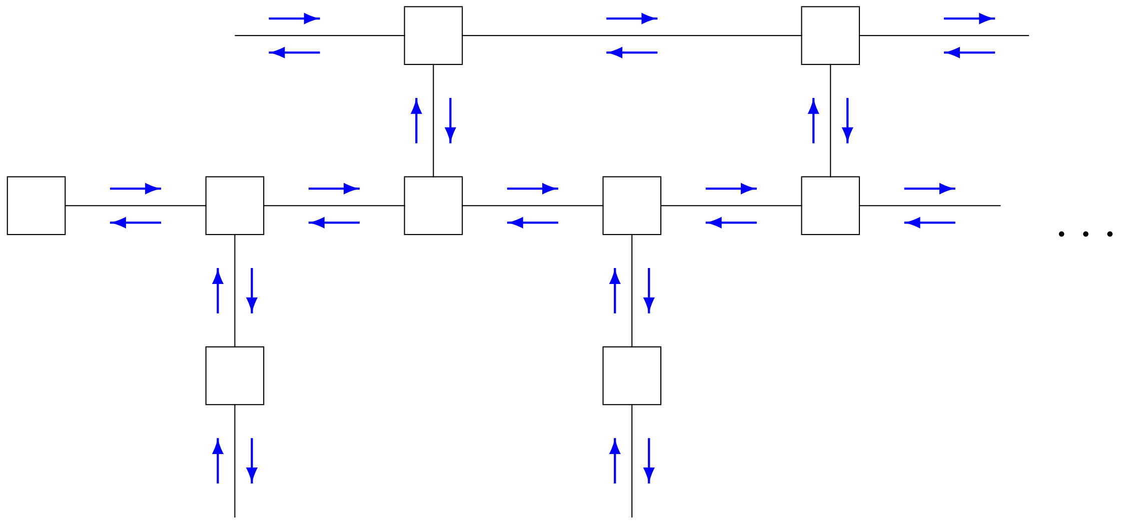
2. Maximization: Compute the "max-marginal"

$$\hat{f}_k(x_k) \triangleq \max_{\substack{x_1, \ldots, x_n \\ \text{except } x_k}} f(x_1, \ldots, x_n)$$

assuming that f is real-valued and nonnegative and has a maximum.

Message Passing Algorithms

operate by passing messages along the edges of a factor graph:



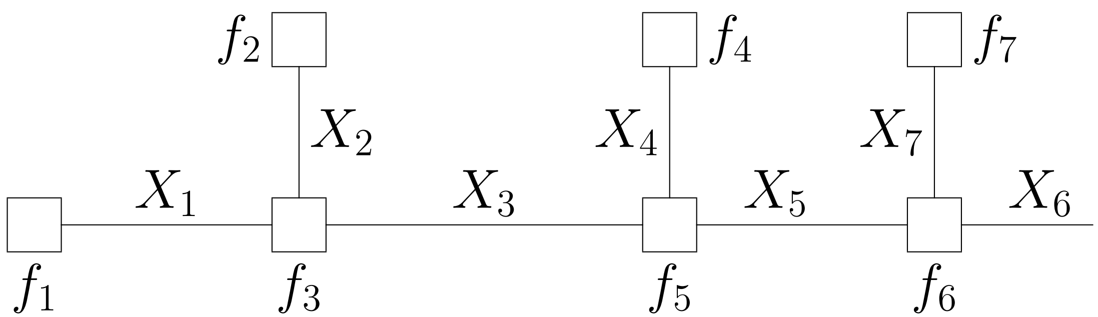
Towards the sum-product algorithm:

Computing Marginals—A Generic Example

Assume we wish to compute

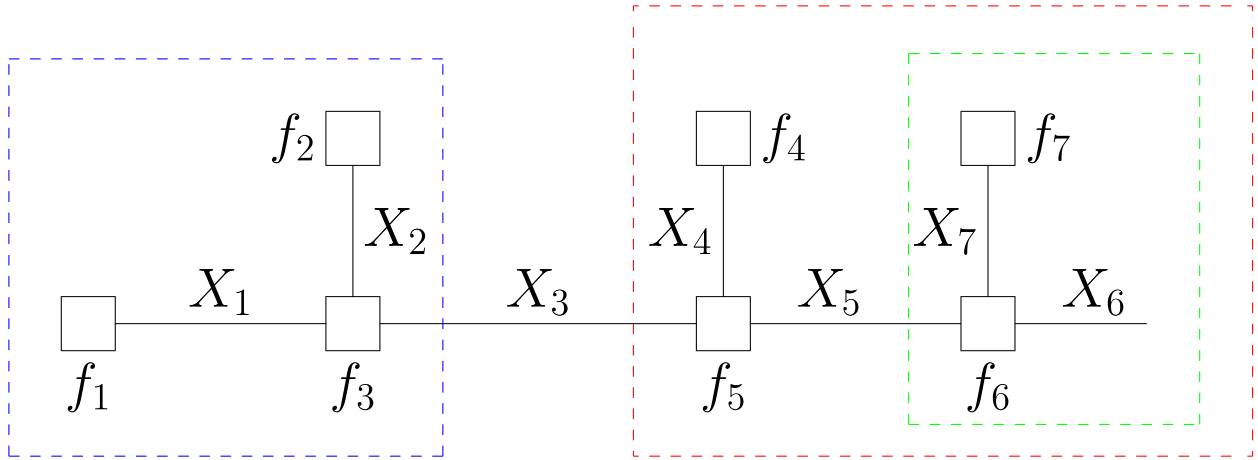
$$\bar{f}_3(x_3) = \sum_{\substack{x_1, \ldots, x_7 \\ \text{except } x_3}} f(x_1, \ldots, x_7)$$

and assume that f can be factored as follows:



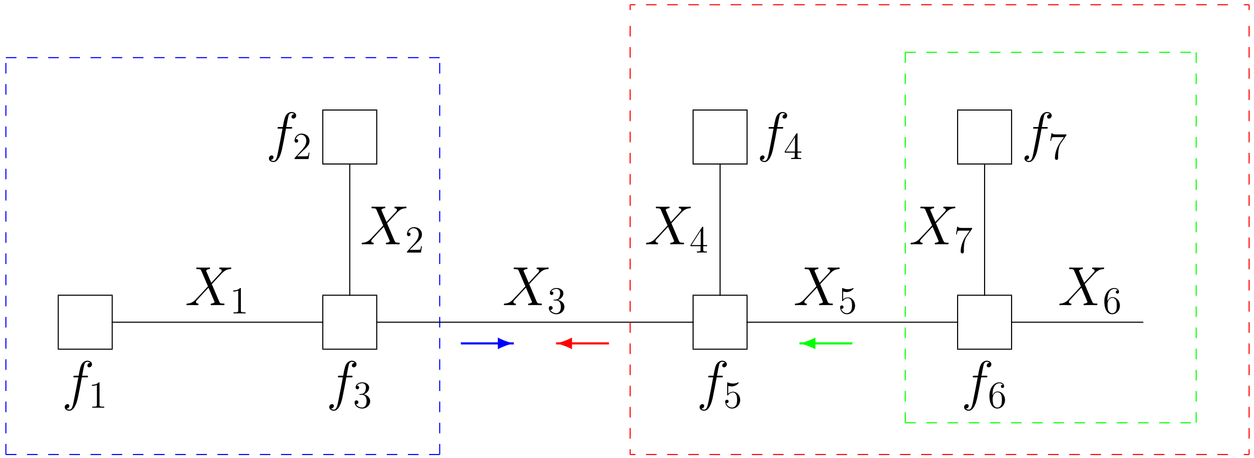
Example cont'd:

Closing Boxes by the Distributive Law



$$\bar{f}_3(x_3) = \left(\sum_{x_1, x_2} f_1(x_1) f_2(x_2) f_3(x_1, x_2, x_3) \right) \cdot \left(\sum_{x_4, x_5} f_4(x_4) f_5(x_3, x_4, x_5) \left(\sum_{x_6, x_7} f_6(x_5, x_6, x_7) f_7(x_7) \right) \right)$$

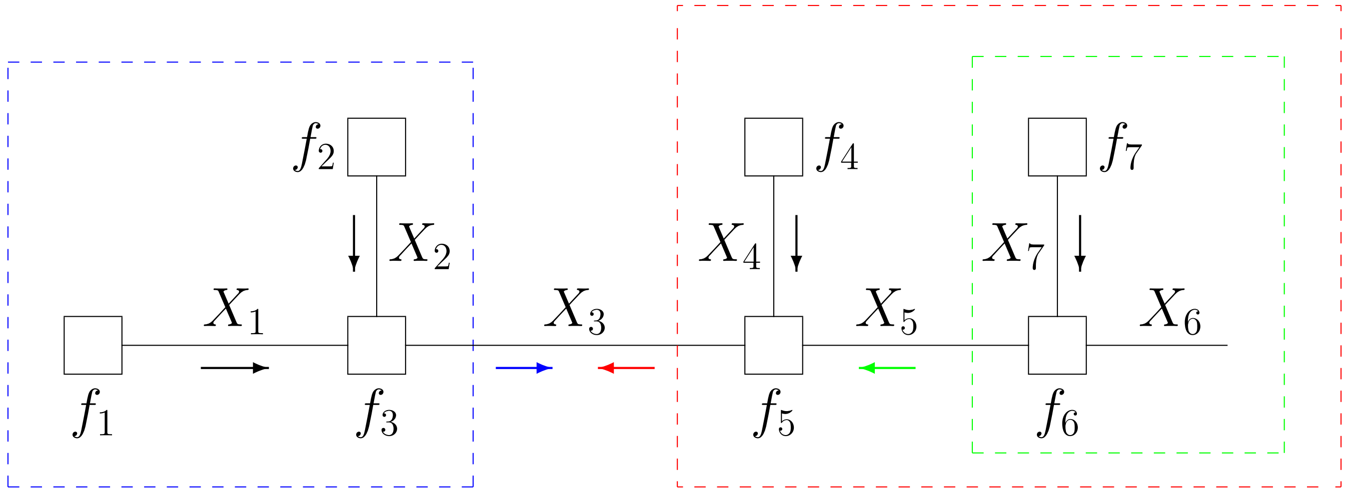
Example cont'd: **Message Passing View**



$$\bar{f}_3(x_3) = \underbrace{\left(\sum_{x_1,x_2} f_1(x_1) f_2(x_2) f_3(x_1,x_2,x_3) \right)}_{\overrightarrow{\mu}_{X_3}(x_3)}$$

$$\cdot \underbrace{\left(\sum_{x_4,x_5} f_4(x_4) f_5(x_3,x_4,x_5) \underbrace{\left(\sum_{x_6,x_7} f_6(x_5,x_6,x_7) f_7(x_7) \right)}_{\overleftarrow{\mu}_{X_5}(x_5)} \right)}_{\overleftarrow{\mu}_{X_3}(x_3)}$$

Example cont'd: **Messages Everywhere**



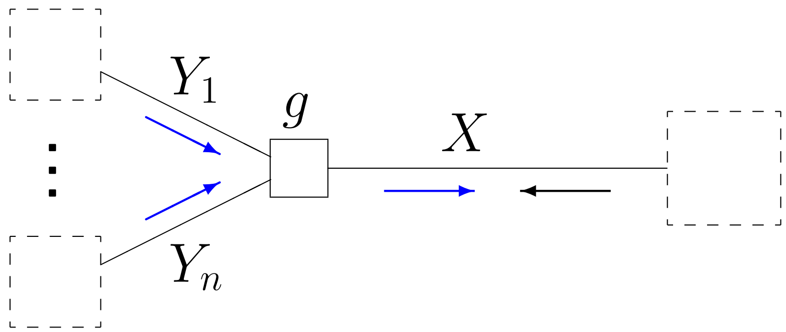
With $\overrightarrow{\mu}_{X_1}(x_1) \stackrel{\triangle}{=} f_1(x_1)$, $\overrightarrow{\mu}_{X_2}(x_2) \stackrel{\triangle}{=} f_2(x_2)$, etc., we have

$$\overrightarrow{\mu}_{X_3}(x_3) = \sum_{x_1,x_2} \overrightarrow{\mu}_{X_1}(x_1)\overrightarrow{\mu}_{X_2}(x_2)f_3(x_1,x_2,x_3)$$

$$\overleftarrow{\mu}_{X_5}(x_5) = \sum_{x_6,x_7} \overrightarrow{\mu}_{X_7}(x_7)f_6(x_5,x_6,x_7)$$

$$\overleftarrow{\mu}_{X_3}(x_3) = \sum_{x_4,x_5} \overrightarrow{\mu}_{X_4}(x_4)\overleftarrow{\mu}_{X_5}(x_5)f_5(x_3,x_4,x_5)$$

The Sum-Product Algorithm (Belief Propagation)



Sum-product message computation rule:

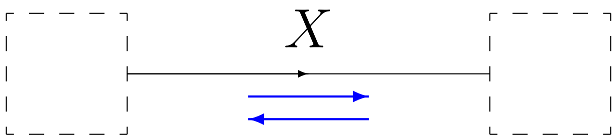
$$\overrightarrow{\mu}_X(x) = \sum_{y_1,\ldots,y_n} g(x, y_1, \ldots, y_n)\overrightarrow{\mu}_{Y_1}(y_1)\cdots\overrightarrow{\mu}_{Y_n}(y_n)$$

Sum-product theorem:

If the factor graph for some global function f has no cycles, then

$$\bar{f}_X(x) = \overrightarrow{\mu}_X(x)\overleftarrow{\mu}_X(x).$$

Arrows and Notation for Messages

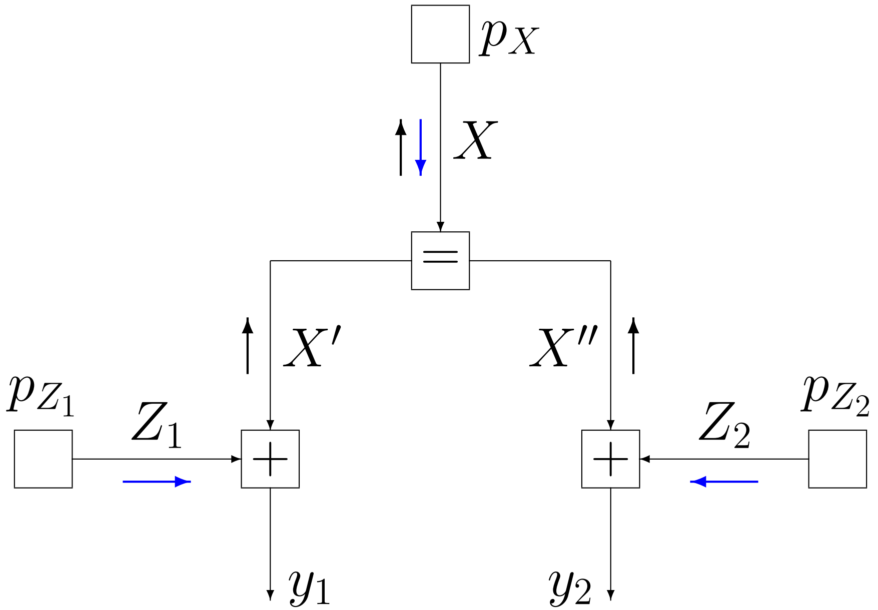


For edges drawn with arrows:

$\overrightarrow{\mu}_X$ denotes the message in the direction of the arrow.
$\overleftarrow{\mu}_X$ denotes the message in the opposite direction.

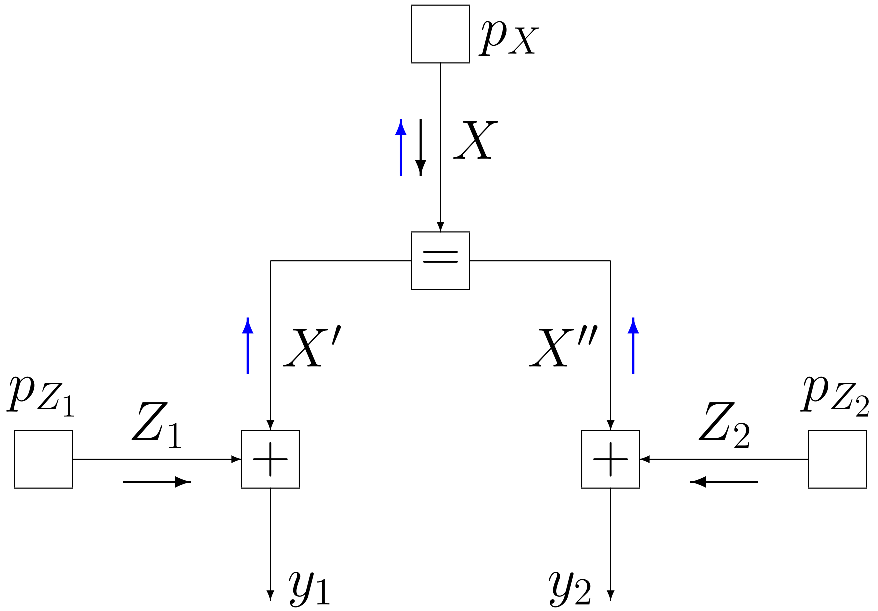
Edges may be drawn with arrows just for the sake of this notation.

Sum-Product Algorithm: a Simple Example



$$\overrightarrow{\mu}_X(x) = p_X(x)$$
$$\overrightarrow{\mu}_{Z_1}(z_1) = p_{Z_1}(z_1)$$
$$\overrightarrow{\mu}_{Z_2}(z_2) = p_{Z_2}(z_2)$$

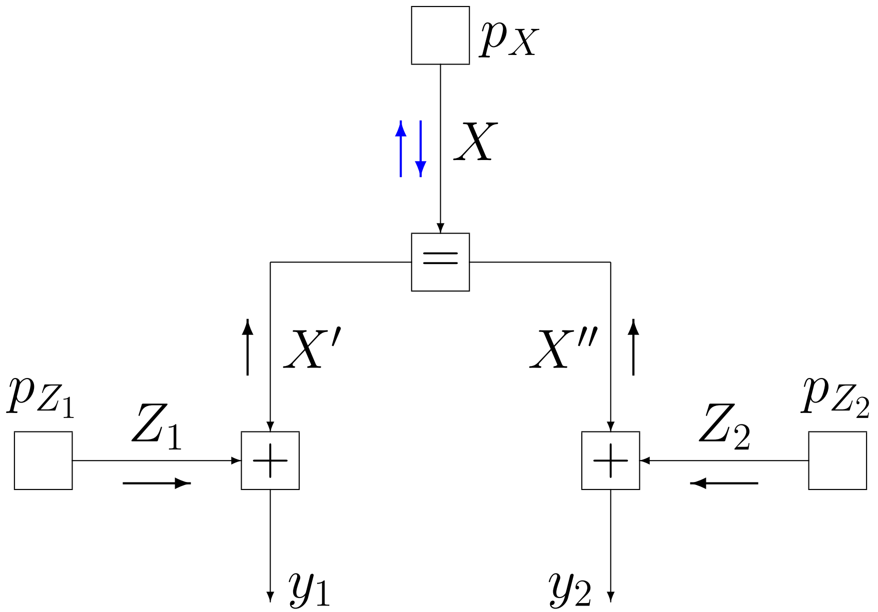
Sum-Product Example cont'd



$$\overleftarrow{\mu}_{X'}(x') = \int_{z_1} \overrightarrow{\mu}_{Z_1}(z_1)\,\delta(x' + z_1 - y_1)\,dz_1 = p_{Z_1}(y_1 - x')$$

$$\overleftarrow{\mu}_X(x) = \int_{x'}\int_{x''} \overleftarrow{\mu}_{X'}(x')\overleftarrow{\mu}_{X''}(x'')\,\delta(x - x')\,\delta(x - x'')\,dx'\,dx'' = p_{Z_1}(y_1 - x)\,p_{Z_2}(y_2 - x)$$

Sum-Product Example cont'd



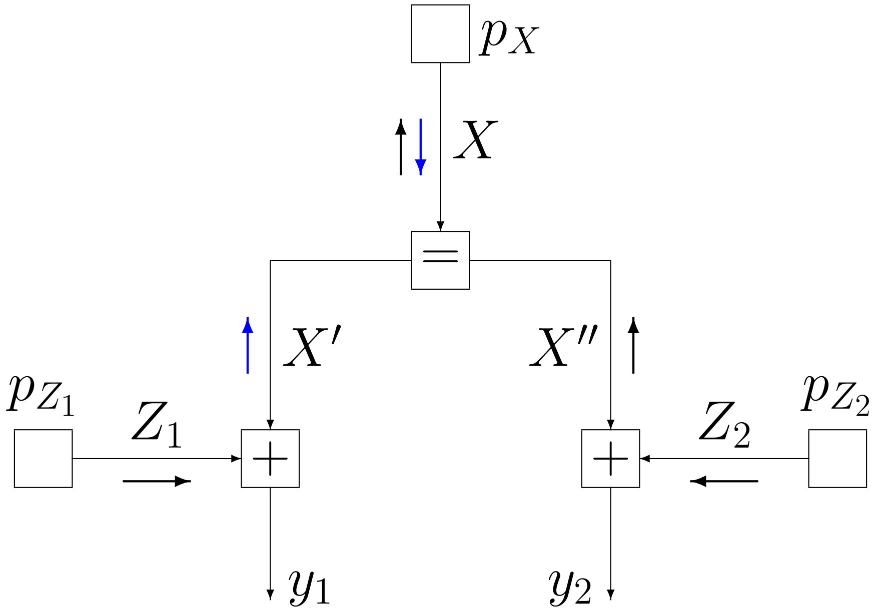
Marginal of the global function at X:

$$\overrightarrow{\mu}_X(x)\overleftarrow{\mu}_X(x) = p_X(x)\,\underbrace{p_{Z_1}(y_1 - x)\,p_{Z_2}(y_2 - x)}_{p(y_1,y_2|x)}$$
$$\propto p(x|y_1, y_2).$$

Messages for Finite-Alphabet Variables

may be represented by a list of function values.

Assume, for example that X takes values in $\{+1, -1\}$:



$$\overrightarrow{\mu}_X = \big(\overrightarrow{\mu}_X(+1), \overrightarrow{\mu}_X(-1)\big) = \big(p_X(+1), p_X(-1)\big)$$

$$\overleftarrow{\mu}_{X'} = \big(\overleftarrow{\mu}_{X'}(+1), \overleftarrow{\mu}_{X'}(-1)\big) = \big(p_{Z_1}(y_1 - 1), p_{Z_1}(y_1 + 1)\big)$$

etc.

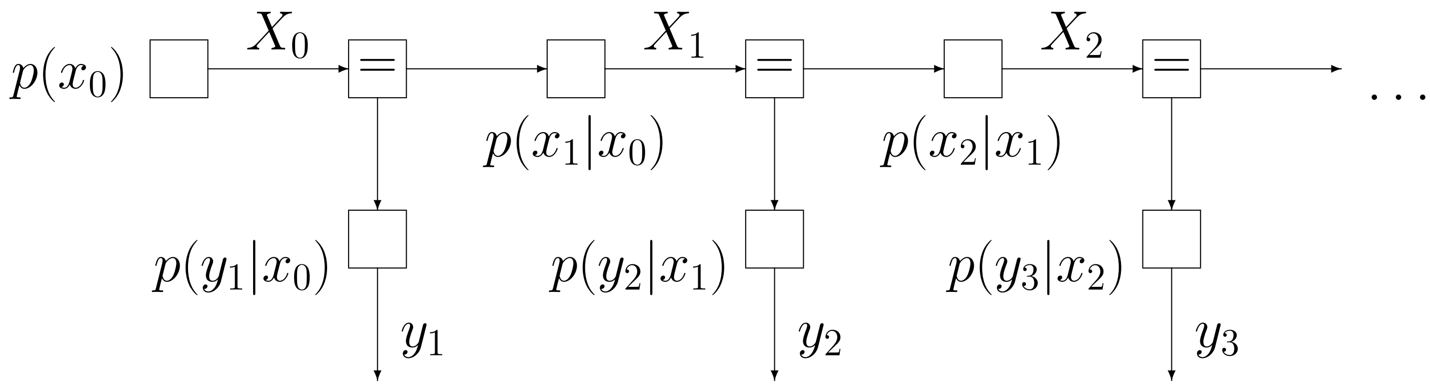
Applying the sum-product algorithm to

Hidden Markov Models

yields recursive algorithms for many things.

Recall the definition of a hidden Markov model (HMM):

$$p(x_0, x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = p(x_0) \prod_{k=1}^{n} p(x_k|x_{k-1}) p(y_k|x_{k-1})$$



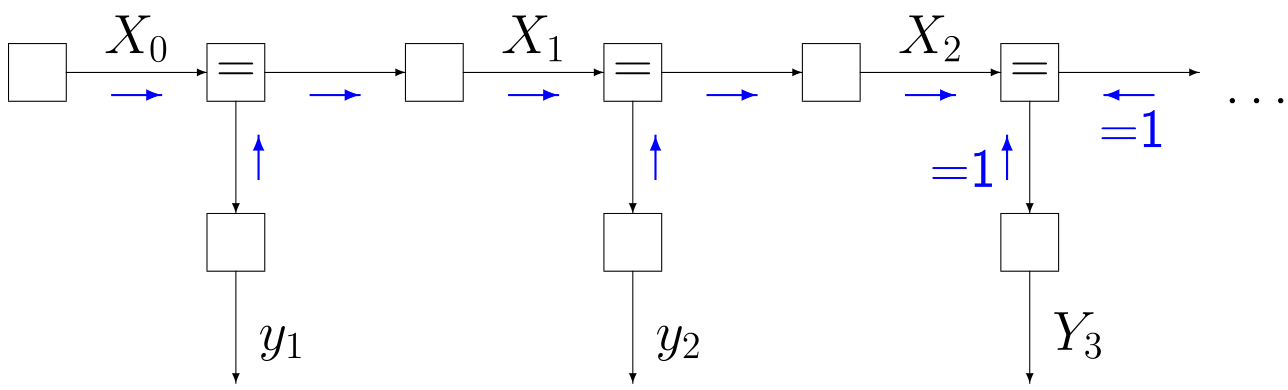
Assume that $Y_1 = y_1, \ldots, Y_n = y_n$ are observed (known).

Sum-product algorithm applied to HMM:

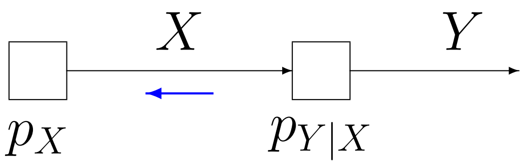
Estimation of Current State

$$p(x_n|y_1,\ldots,y_n) = \frac{p(x_n,y_1,\ldots,y_n)}{p(y_1,\ldots,y_n)}$$
$$\propto p(x_n,y_1,\ldots,y_n)$$
$$= \sum_{x_0}\cdots\sum_{x_{n-1}} p(x_0,x_1,\ldots,x_n,y_1,y_2,\ldots,y_n)$$
$$= \overrightarrow{\mu}_{X_n}(x_n).$$

For $n = 2$:



Backward Message in Chain Rule Model



If $Y = y$ is known (observed):

$$\overleftarrow{\mu}_X(x) = p_{Y|X}(y|x),$$

the likelihood function.

If Y is unknown:

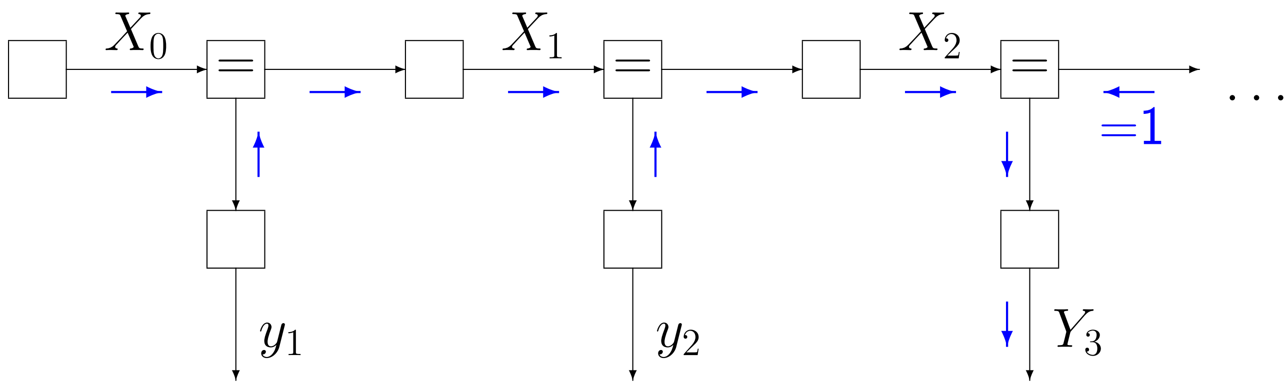
$$\begin{aligned}\overleftarrow{\mu}_X(x) &= \sum_y p_{Y|X}(y|x) \\ &= 1.\end{aligned}$$

Sum-product algorithm applied to HMM:

Prediction of Next Output Symbol

$$
\begin{aligned}
p(y_{n+1}|y_1,\ldots,y_n) &= \frac{p(y_1,\ldots,y_{n+1})}{p(y_1,\ldots,y_n)} \\
&\propto p(y_1,\ldots,y_{n+1}) \\
&= \sum_{x_0,x_1,\ldots,x_n} p(x_0,x_1,\ldots,x_n,y_1,y_2,\ldots,y_n,y_{n+1}) \\
&= \overrightarrow{\mu}_{Y_n}(y_n).
\end{aligned}
$$

For $n = 2$:

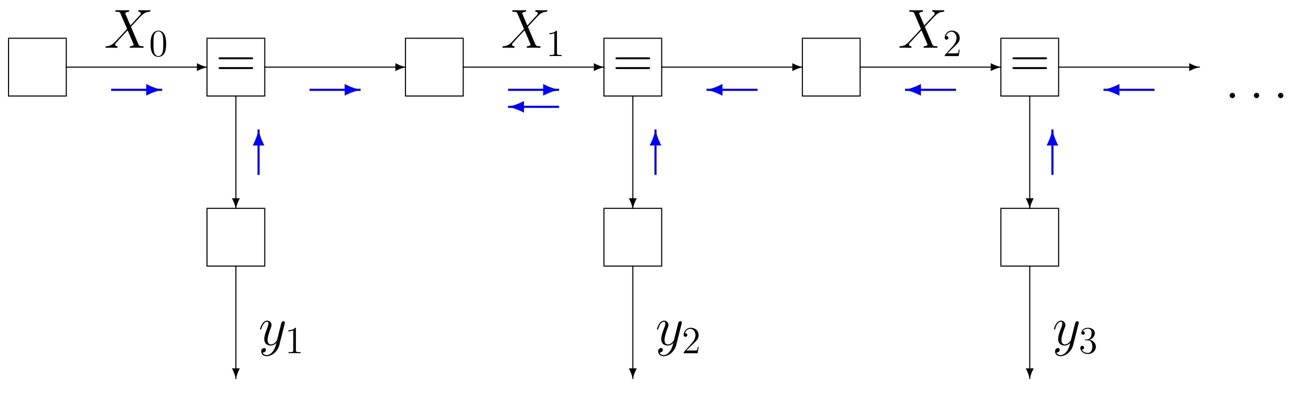


Sum-product algorithm applied to HMM:

Estimation of Time-k State

$$
\begin{aligned}
p(x_k \mid y_1, y_2, \ldots, y_n) &= \frac{p(x_k, y_1, y_2, \ldots, y_n)}{p(y_1, y_2, \ldots, y_n)} \\
&\propto p(x_k, y_1, y_2, \ldots, y_n) \\
&= \sum_{\substack{x_0, \ldots, x_n \\ \text{except } x_k}} p(x_0, x_1, \ldots, x_n, y_1, y_2, \ldots, y_n) \\
&= \overrightarrow{\mu}_{X_k}(x_k)\, \overleftarrow{\mu}_{X_k}(x_k)
\end{aligned}
$$

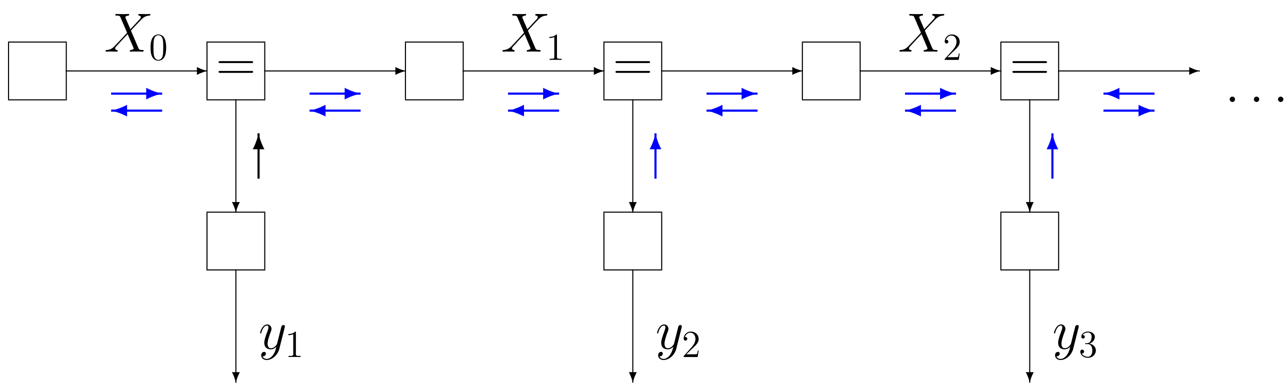
For $k = 1$:



Sum-product algorithm applied to HMM:

All States Simultaneously

$p(x_k|y_1, \ldots, y_n)$ for all k:



In this application, the sum-product algorithm coincides with the Baum-Welch / BCJR forward-backward algorithm.

Scaling of Messages

In all the examples so far:

- The final result (such as $\overrightarrow{\mu}_{X_k}(x_k)\overleftarrow{\mu}_{X_k}(x_k)$) equals the desired quantity (such as $p(x_k|y_1,\ldots,y_n)$) only up to a scale factor.
- The missing scale factor γ may be recovered at the end from the condition

$$\sum_{x_k} \gamma \overrightarrow{\mu}_{X_k}(x_k)\overleftarrow{\mu}_{X_k}(x_k) = 1.$$

- It follows that messages may be scaled freely along the way.
- Such message scaling is often mandatory to avoid numerical problems.

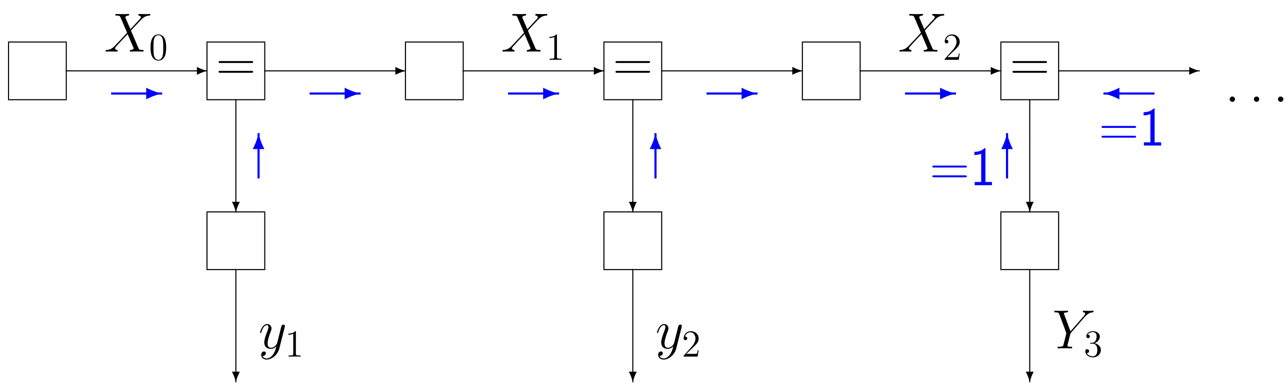
Sum-product algorithm applied to HMM:

Probability of the Observation

$$p(y_1, \ldots, y_n) = \sum_{x_0} \ldots \sum_{x_n} p(x_0, x_1, \ldots, x_n, y_1, y_2, \ldots, y_n)$$
$$= \sum_{x_n} \overrightarrow{\mu}_{X_n}(x_n).$$

This is a number. Scale factors cannot be neglected in this case.

For $n = 2$:



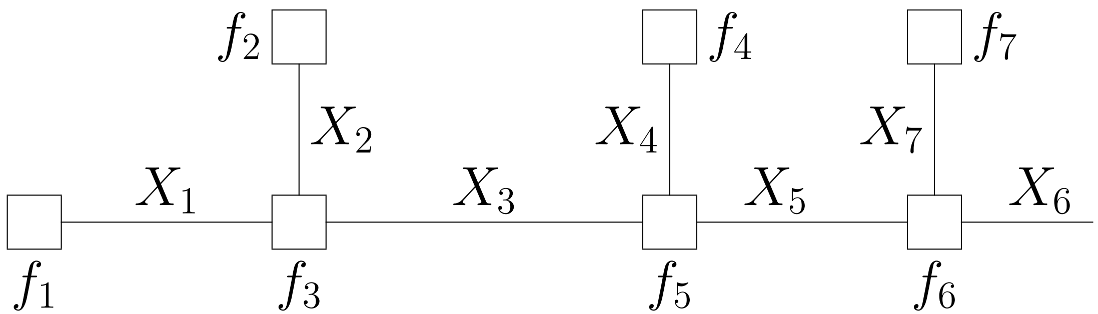
Towards the max-product algorithm:

Computing Max-Marginals—A Generic Example

Assume we wish to compute

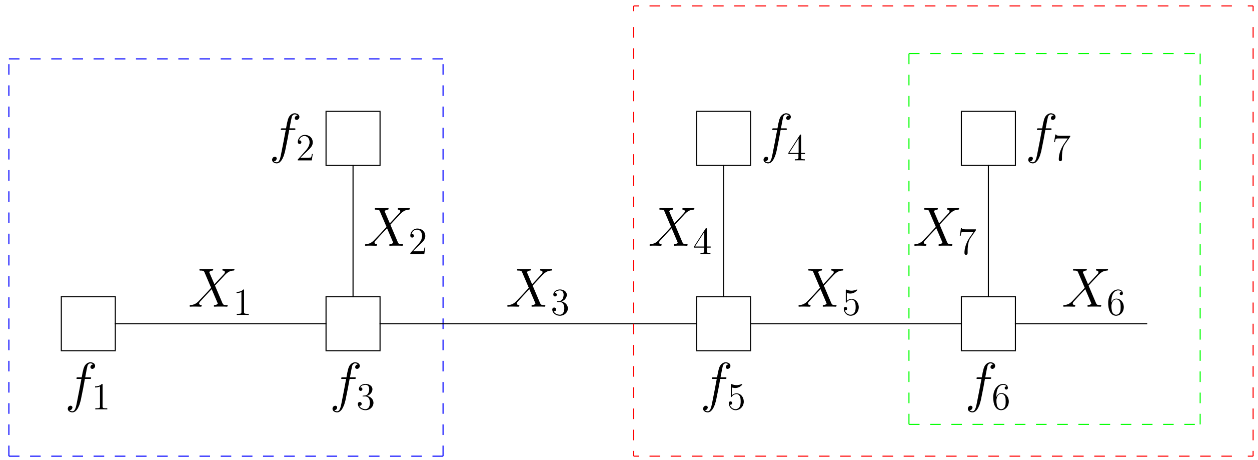
$$\hat{f}_3(x_3) = \max_{\substack{x_1, \ldots, x_7 \\ \text{except } x_3}} f(x_1, \ldots, x_7)$$

and assume that f can be factored as follows:



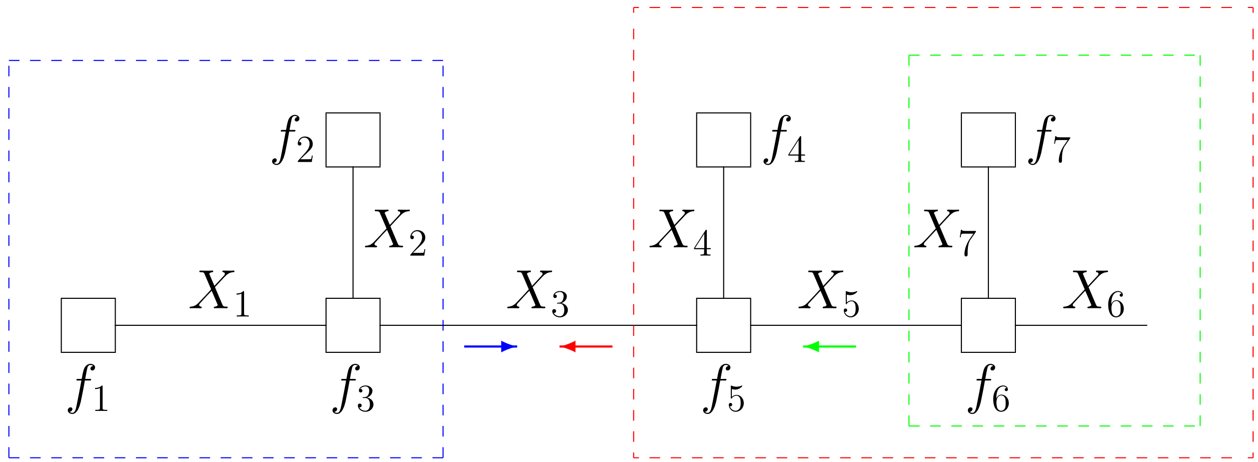
Example:

Closing Boxes by the Distributive Law



$$\hat{f}_3(x_3) = \left(\max_{x_1,x_2} f_1(x_1) f_2(x_2) f_3(x_1, x_2, x_3) \right) \cdot \left(\max_{x_4,x_5} f_4(x_4) f_5(x_3, x_4, x_5) \left(\max_{x_6,x_7} f_6(x_5, x_6, x_7) f_7(x_7) \right) \right)$$

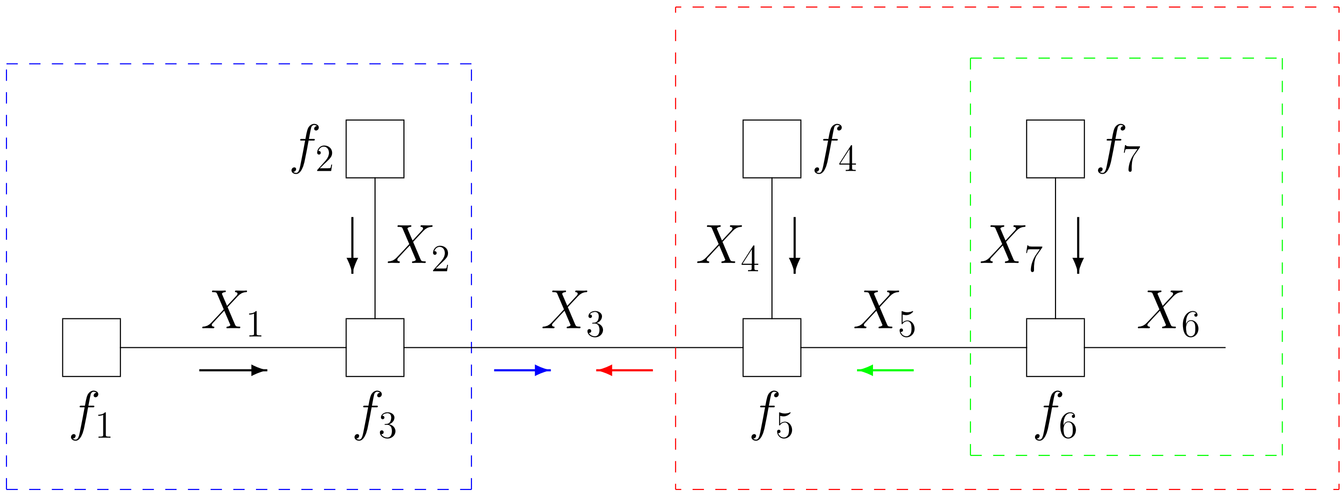
Example cont'd: **Message Passing View**



$$\hat{f}_3(x_3) = \Bigg(\underbrace{\max_{x_1,x_2} f_1(x_1) f_2(x_2) f_3(x_1, x_2, x_3)}_{\overrightarrow{\mu}_{X_3}(x_3)} \Bigg)$$

$$\cdot \underbrace{\Bigg(\max_{x_4,x_5} f_4(x_4) f_5(x_3, x_4, x_5) \Big(\underbrace{\max_{x_6,x_7} f_6(x_5, x_6, x_7) f_7(x_7)}_{\overleftarrow{\mu}_{X_5}(x_5)} \Big) \Bigg)}_{\overleftarrow{\mu}_{X_3}(x_3)}$$

Example cont'd: **Messages Everywhere**



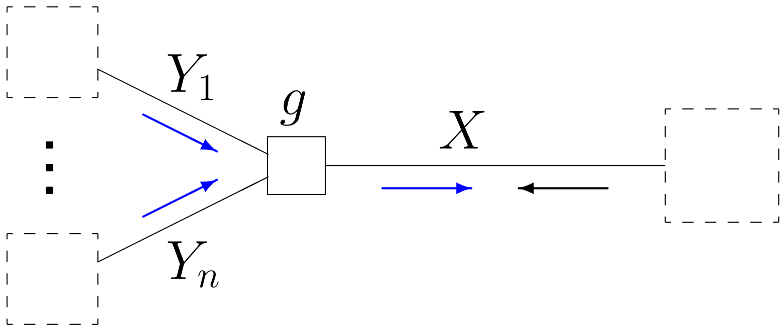
With $\overrightarrow{\mu}_{X_1}(x_1) \triangleq f_1(x_1)$, $\overrightarrow{\mu}_{X_2}(x_2) \triangleq f_2(x_2)$, etc., we have

$$\overrightarrow{\mu}_{X_3}(x_3) = \max_{x_1,x_2} \overrightarrow{\mu}_{X_1}(x_1)\overrightarrow{\mu}_{X_2}(x_2)f_3(x_1,x_2,x_3)$$

$$\overleftarrow{\mu}_{X_5}(x_5) = \max_{x_6,x_7} \overrightarrow{\mu}_{X_7}(x_7)f_6(x_5,x_6,x_7)$$

$$\overleftarrow{\mu}_{X_3}(x_3) = \max_{x_4,x_5} \overrightarrow{\mu}_{X_4}(x_4)\overleftarrow{\mu}_{X_5}(x_5)f_5(x_3,x_4,x_5)$$

The Max-Product Algorithm



Max-product message computation rule:

$$\overrightarrow{\mu}_X(x) = \max_{y_1,\ldots,y_n} g(x, y_1, \ldots, y_n)\overrightarrow{\mu}_{Y_1}(y_1)\cdots\overrightarrow{\mu}_{Y_n}(y_n)$$

Max-product theorem:

If the factor graph for some global function f has no cycles, then

$$\hat{f}_X(x) = \overrightarrow{\mu}_X(x)\overleftarrow{\mu}_X(x).$$

Max-product algorithm applied to HMM:

MAP Estimate of the State Trajectory

The estimate

$$
\begin{aligned}
(\hat{x}_0, \ldots, \hat{x}_n)_{\mathrm{MAP}} &= \operatorname*{argmax}_{x_0,\ldots,x_n} p(x_0, \ldots, x_n | y_1, \ldots, y_n) \\
&= \operatorname*{argmax}_{x_0,\ldots,x_n} p(x_0, \ldots, x_n, y_1, \ldots, y_n)
\end{aligned}
$$

may be obtained by computing

$$
\begin{aligned}
\hat{p}_k(x_k) &\stackrel{\triangle}{=} \max_{\substack{x_1, \ldots, x_n \\ \text{except } x_k}} p(x_0, \ldots, x_n, y_1, \ldots, y_n) \\
&= \overrightarrow{\mu}_{X_k}(x_k) \overleftarrow{\mu}_{X_k}(x_k)
\end{aligned}
$$

for all k by forward-backward max-product sweeps.

In this example, the max-product algorithm is a time-symmetric version of the Viterbi algorithm with soft output.

Max-product algorithm applied to HMM:

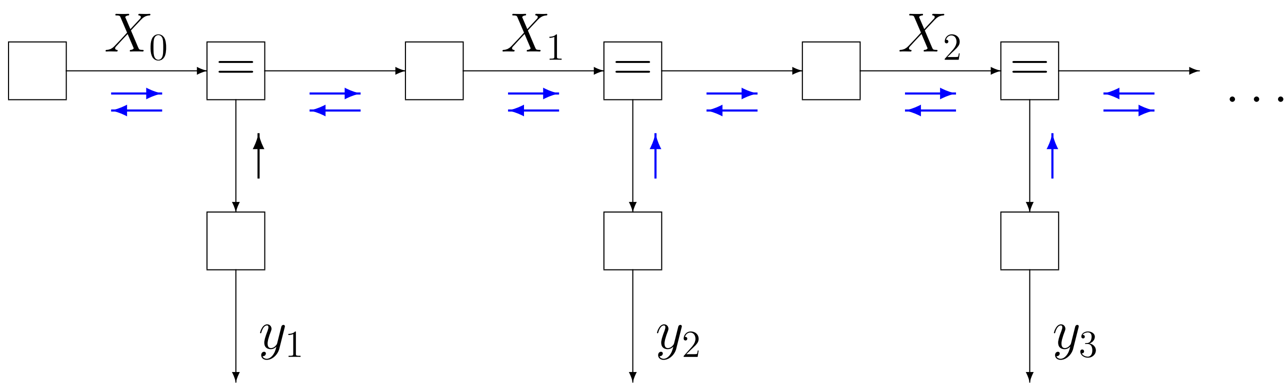
MAP Estimate of the State Trajectory cont'd

Computing

$$\hat{p}_k(x_k) \triangleq \max_{\substack{x_1,\ldots,x_n \\ \text{except } x_k}} p(x_0,\ldots,x_n,y_1,\ldots,y_n)$$

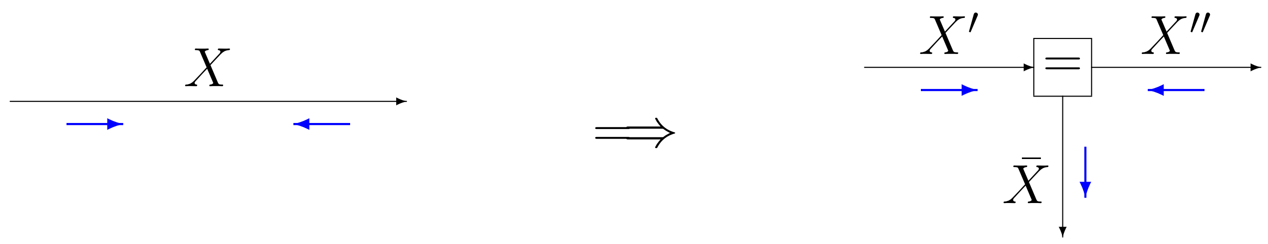
$$= \overrightarrow{\mu}_{X_k}(x_k)\overleftarrow{\mu}_{X_k}(x_k)$$

simultaneously for all k:



Marginals and Output Edges

Marginals such $\overrightarrow{\mu}_X(x)\overleftarrow{\mu}_X(x)$ may be viewed as messages out of a "output half edge" (without incoming message):

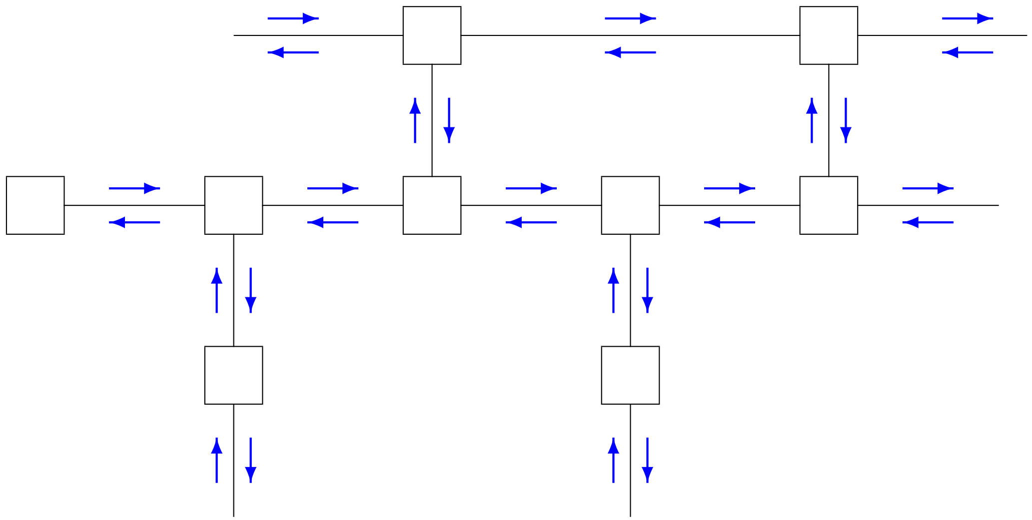


$$\overrightarrow{\mu}_{\bar{X}}(x) = \int_{x'}\int_{x''} \overrightarrow{\mu}_{X'}(x')\, \overleftarrow{\mu}_{X''}(x'')\, \delta(x-x')\,\delta(x-x'')\, dx'\, dx''$$
$$= \overrightarrow{\mu}_{X'}(x)\, \overleftarrow{\mu}_{X''}(x)$$

$\Longrightarrow$ Marginals are computed like messages out of "="-nodes.

Outline

What About Factor Graphs with Cycles?



What About Factor Graphs with Cycles?

- Generally iterative algorithms.
- For example, alternating maximization

$$\hat{x}_{\text{new}} = \operatorname*{argmax}_x f(x, \hat{y}) \quad \text{and} \quad \hat{y}_{\text{new}} = \operatorname*{argmax}_y f(\hat{x}, y)$$

  using the max-product algorithm in each iteration.
- Iterative sum-product message passing gives excellent results for maximization(!) in some applications (e.g., the decoding of error correcting codes).
- Many other useful algorithms can be formulated in message passing form (e.g., gradient ascent, Gibbs sampling, expectation maximization, variational methods,...).
- Rich and vast research area...

Outline

Factor Graph of an Error Correcting Code

$\approx$ Tanner graph of the code (Tanner 1981)

A factor graph of a code $C \subset F^n$ represents (a factorization of) the membership indicator function of the code:

$$I_C : F^n \to \{0, 1\} : x \mapsto \begin{cases} 1, & \text{if } x \in C \\ 0, & \text{else} \end{cases}$$

Factor Graph from Parity Check Matrix

Example: $(7,4,3)$ binary Hamming code. ($F \triangleq$ GF(2).)

$$C = \{x \in F^n : Hx^T = 0\}$$

with

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

The membership indicator function

$$I_C : F^n \to \{0,1\} : x \mapsto \begin{cases} 1, & \text{if } x \in C \\ 0, & \text{else} \end{cases}$$

of this code may be written as

$$I_C(x_1, \ldots, x_n) = \delta(x_1 \oplus x_2 \oplus x_3 \oplus x_5) \cdot \delta(x_2 \oplus x_3 \oplus x_4 \oplus x_6) \cdot \delta(x_3 \oplus x_4 \oplus x_5 \oplus x_7)$$

where $\oplus$ denotes addition modulo 2. Each factor corresponds to one row of the parity check matrix.

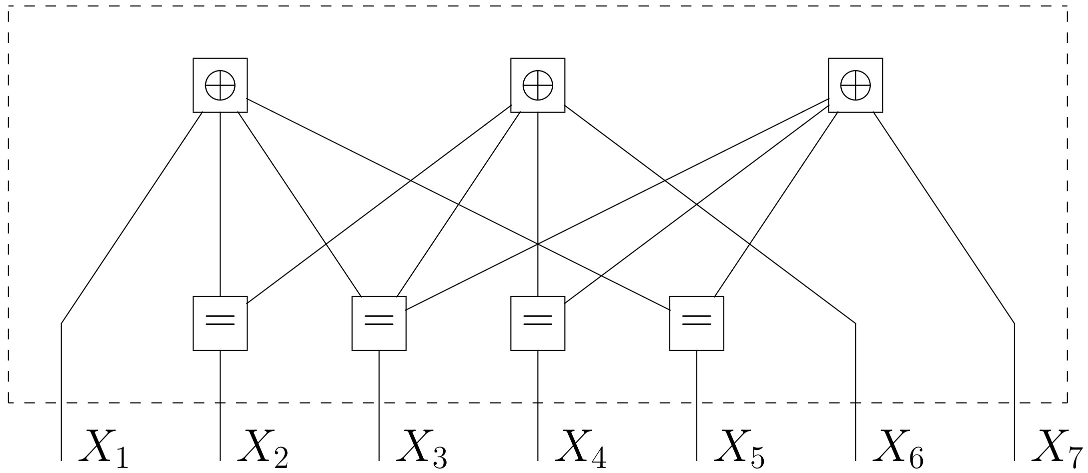
Factor Graph from Parity Check Matrix (cont'd)

Example: $(7, 4, 3)$ binary Hamming code.

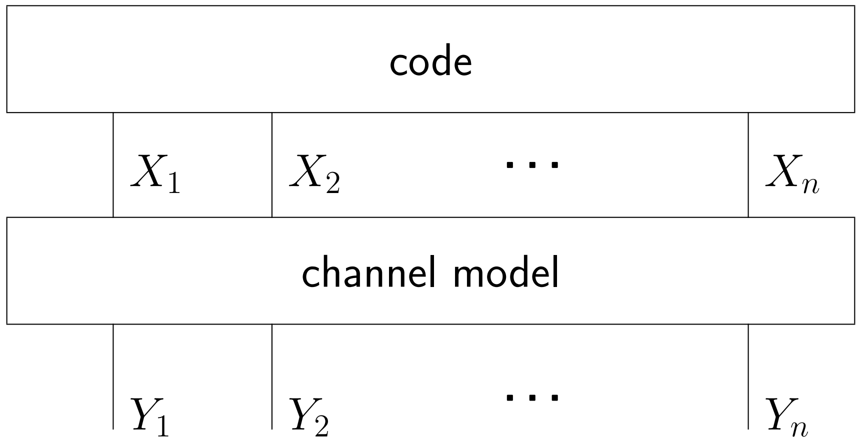
$$C = \{x \in F^n : Hx^T = 0\}$$

with

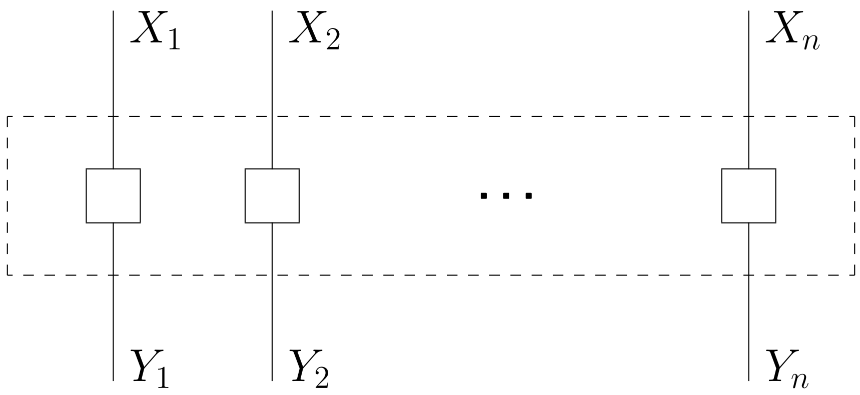
$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$



Factor Graph for Joint Code / Channel Model



Example: memoryless channel:



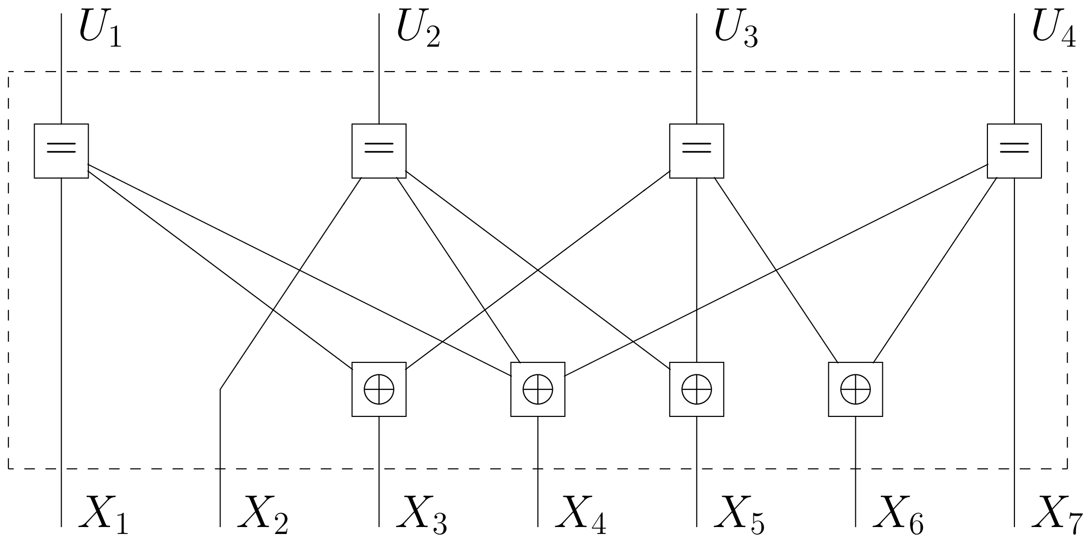
Factor Graph from Generator Matrix

Example: $(7, 4, 3)$ binary Hamming code is the image of

$$F^k \longrightarrow F^n : u \mapsto uG$$

with

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

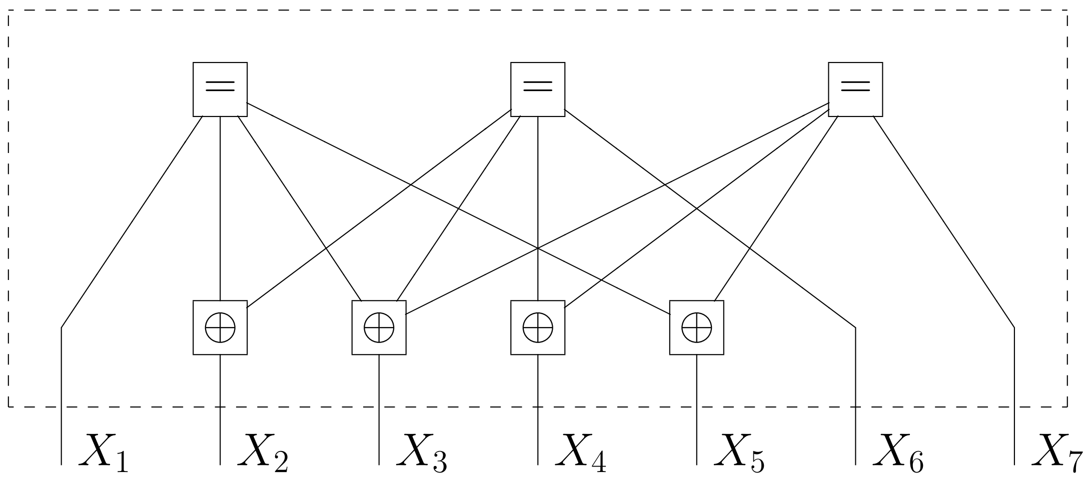


Factor Graph of Dual Code

is obtained by interchanging partity check nodes and equality check nodes (Kschischang, Forney).

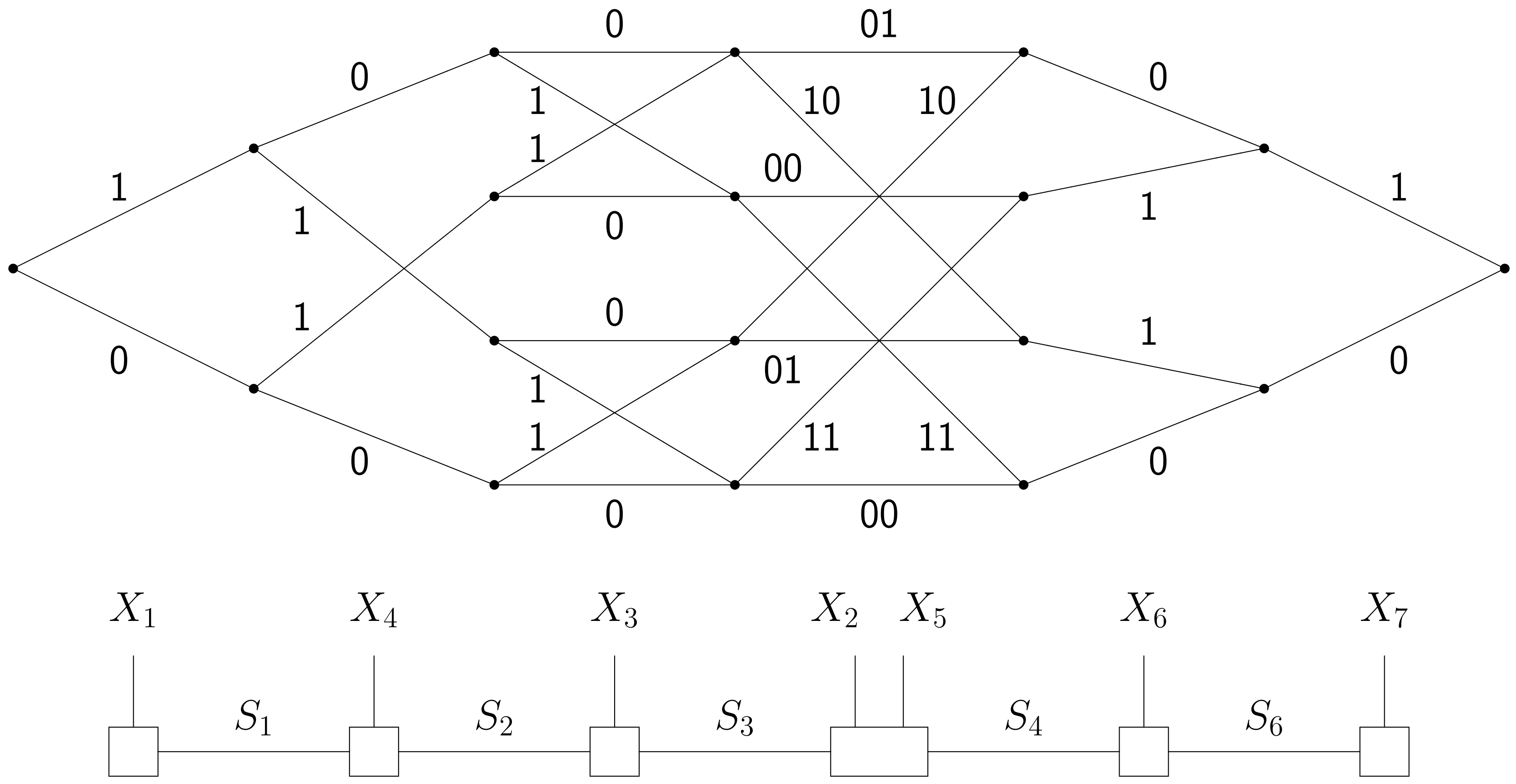
Works only for Forney-style factor graphs where all code symbols are external (half-edge) variables.

Example: dual of $(7, 4, 3)$ binary Hamming code

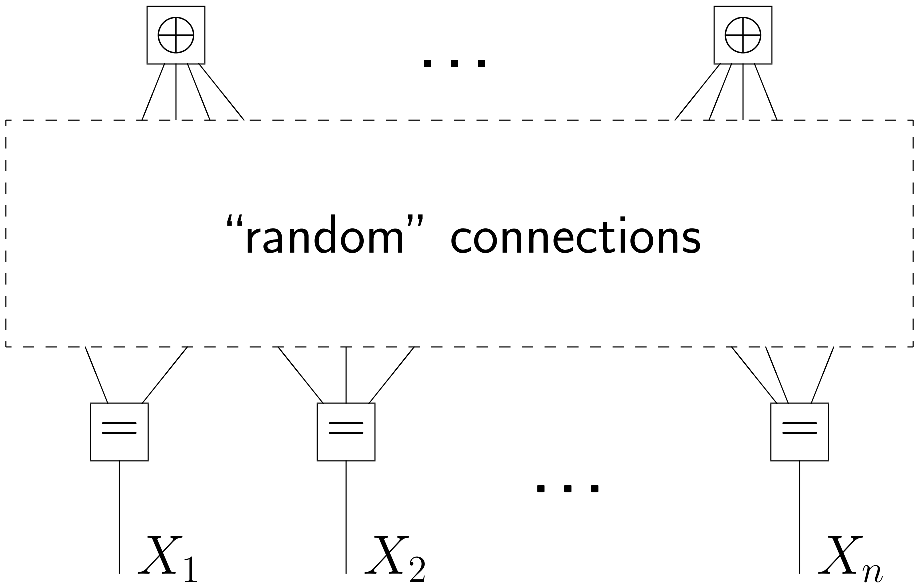


Factor Graph Corresponding to Trellis

Example: $(7, 4, 3)$ binary Hamming code



Factor Graph of Low-Density Parity-Check Codes



Standard decoder: iterative sum-product message passing.

Convergence is not guaranteed!

Much recent / ongoing research on improved decoding:
Yedidia, Freeman, Weiss; Feldman; Wainwright; Chertkov...

Single-Number Parameterizations of Soft-Bit Messages

Difference: $\Delta \stackrel{\triangle}{=} \dfrac{\mu(0) - \mu(1)}{\mu(0) + \mu(1)}$ = mean of $\{+1, -1\}$-representation

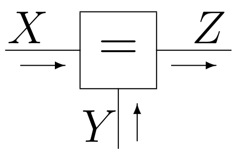
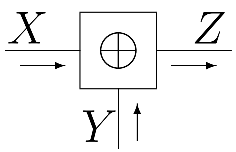
Ratio: $\Lambda \stackrel{\triangle}{=} \mu(0)/\mu(1)$

Logarithm of ratio: $L \stackrel{\triangle}{=} \log\big(\mu(0)/\mu(1)\big)$

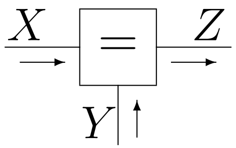
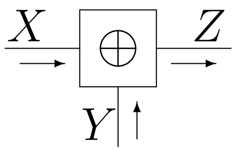
Conversions among Parameterizations

	$\begin{pmatrix}\mu(0)\\ \mu(1)\end{pmatrix}$	$\Delta = m$	Λ	L
$\begin{pmatrix}\mu(0)\\ \mu(1)\end{pmatrix}$		$\begin{pmatrix}\frac{1+\Delta}{2}\\ \frac{1-\Delta}{2}\end{pmatrix}$	$\begin{pmatrix}\frac{\Lambda}{\Lambda+1}\\ \frac{1}{\Lambda+1}\end{pmatrix}$	$\begin{pmatrix}\frac{e^L}{e^L+1}\\ \frac{1}{e^L+1}\end{pmatrix}$
$\Delta = m$	$\frac{\mu(0)-\mu(1)}{\mu(0)+\mu(1)}$		$\frac{\Lambda-1}{\Lambda+1}$	$\tanh(L/2)$
Λ	$\frac{\mu(0)}{\mu(1)}$	$\frac{1+\Delta}{1-\Delta}$		e^L
L	$\ln\frac{\mu(0)}{\mu(1)}$	$2\tanh^{-1}(\Delta)$	$\ln\Lambda$	
σ^2	$\frac{4\mu(0)\mu(1)}{(\mu(0)+\mu(1))^2}$	$1-m^2$	$\frac{4\Lambda}{(\Lambda+1)^2}$	$\frac{4}{e^L+e^{-L}+2}$

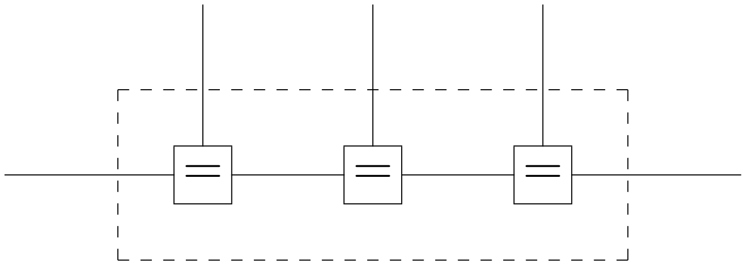
Sum-Product Rules for Binary Parity Check Codes

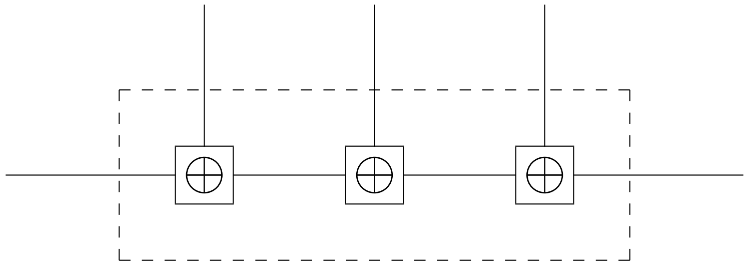
 $\delta[x-y]\,\delta[x-z]$	$\begin{pmatrix}\mu_Z(0)\\ \mu_Z(1)\end{pmatrix} = \begin{pmatrix}\mu_X(0)\,\mu_Y(0)\\ \mu_X(1)\,\mu_Y(1)\end{pmatrix}$ $\Delta_Z = \frac{\Delta_X + \Delta_Y}{1+\Delta_X\Delta_Y}$ $\Lambda_Z = \Lambda_X \cdot \Lambda_Y$ $L_Z = L_X + L_Y$
 $\delta[x \oplus y \oplus z]$	$\begin{pmatrix}\mu_Z(0)\\ \mu_Z(1)\end{pmatrix} = \begin{pmatrix}\mu_X(0)\,\mu_Y(0) + \mu_X(1)\,\mu_Y(1)\\ \mu_X(0)\,\mu_Y(1) + \mu_X(1)\,\mu_Y(0)\end{pmatrix}$ $\Delta_Z = \Delta_X \cdot \Delta_Y$ $\Lambda_Z = \frac{1+\Lambda_X\Lambda_Y}{\Lambda_X + \Lambda_Y}$ $\tanh(L_Z/2) = \tanh(L_X/2)\cdot\tanh(L_Y/2)$

Max-Product Rules for Binary Parity Check Codes

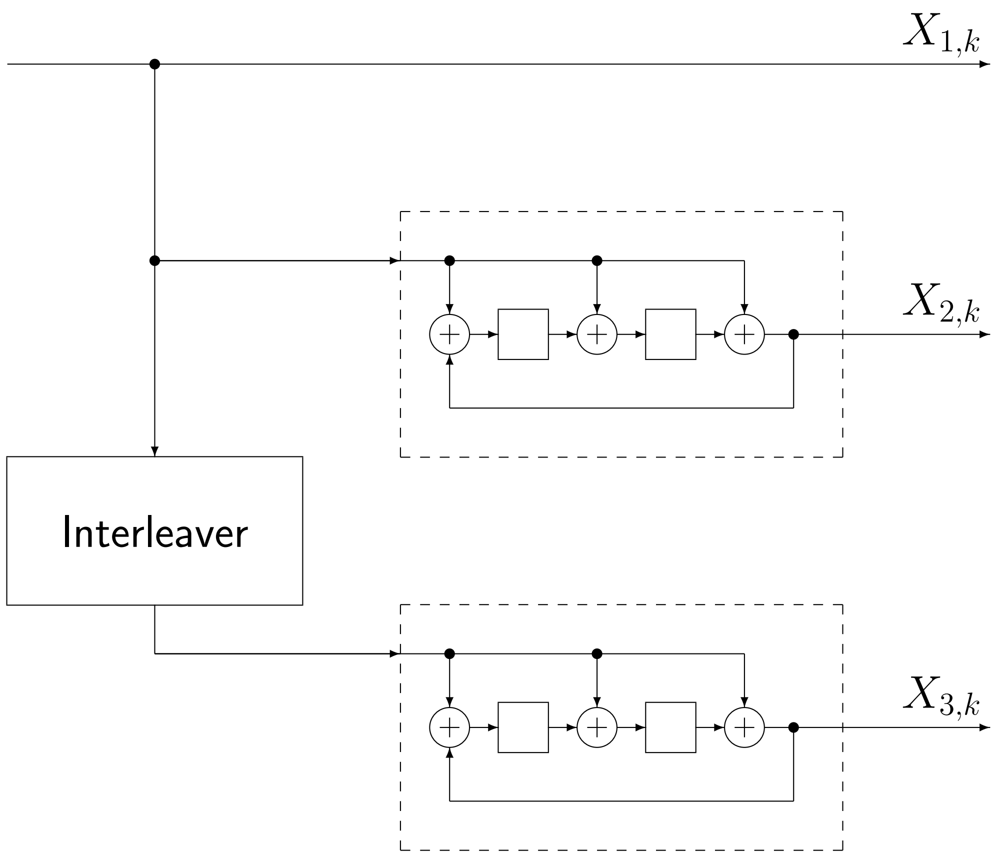
 $\delta[x-y]\,\delta[x-z]$	$\begin{pmatrix} \mu_Z(0) \\ \mu_Z(1) \end{pmatrix} = \begin{pmatrix} \mu_X(0)\,\mu_Y(0) \\ \mu_X(1)\,\mu_Y(1) \end{pmatrix}$ $L_Z = L_X + L_Y$
 $\delta[x \oplus y \oplus z]$	$\begin{pmatrix} \mu_Z(0) \\ \mu_Z(1) \end{pmatrix} = \begin{pmatrix} \max\left\{\mu_X(0)\,\mu_Y(0),\ \mu_X(1)\,\mu_Y(1)\right\} \\ \max\left\{\mu_X(0)\,\mu_Y(1),\ \mu_X(1)\,\mu_Y(0)\right\} \end{pmatrix}$ $\lvert L_Z \rvert = \min\left\{\lvert L_X \rvert, \lvert L_Y \rvert\right\}$ $\mathrm{sgn}(L_Z) = \mathrm{sgn}(L_X)\cdot\mathrm{sgn}(L_Y)$

Decomposition of Multi-Bit Checks

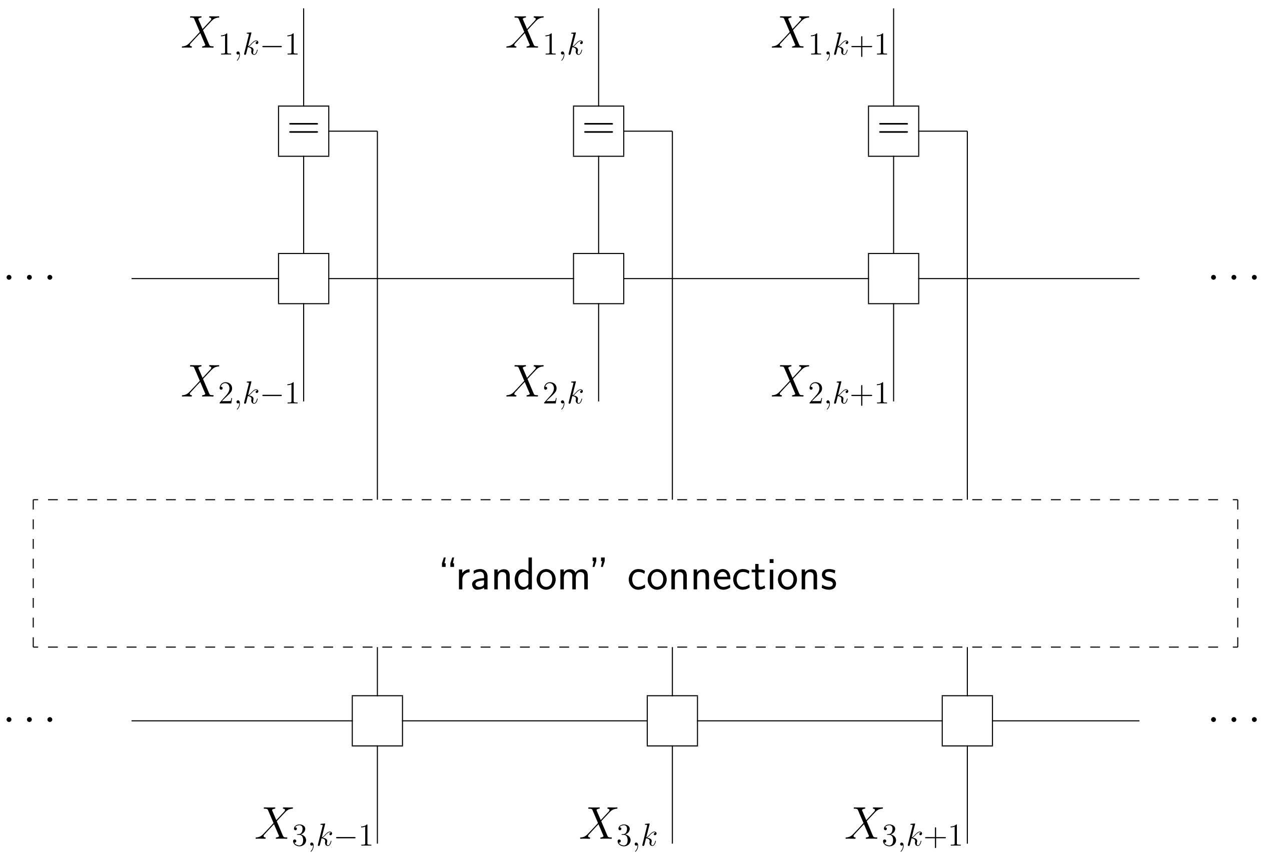




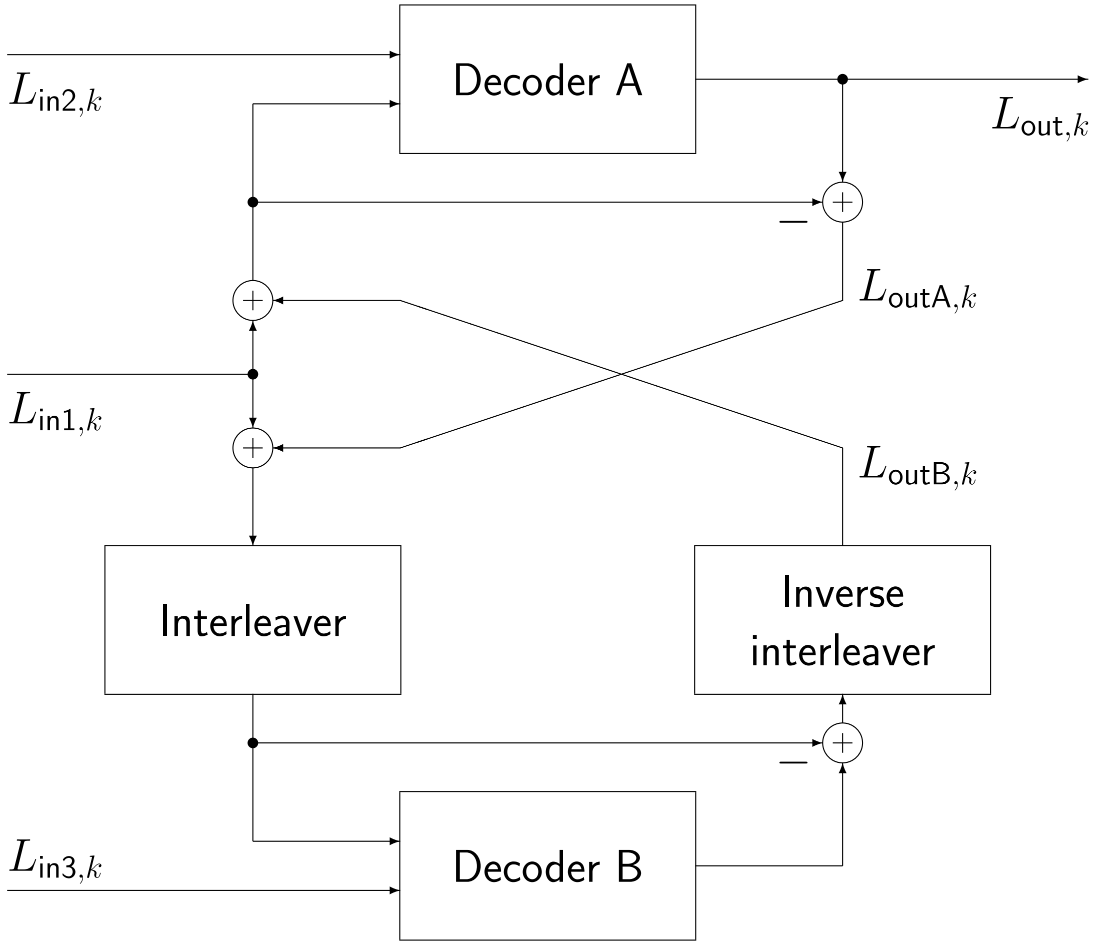
Encoder of a Turbo Code



Factor Graph of a Turbo Code



Turbo Decoder: Conventional View



Messages in a Turbo Decoder