ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Factor Graphs and Message Passing Algorithms — Part 2: Model-Based Signal Processing

Based largely on

H.-A. Loeliger, J. Dauwels, Junli Hu, S. Korl, Li Ping, and F. R. Kschischang,
"The factor graph approach to model-based signal processing,"
Proceedings of the IEEE, June 2007.

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. Note that

$$\operatorname{argmax} f(x_1,\ldots,x_n) = \left(\operatorname{argmax} \hat{f}_1(x_1),\ldots,\operatorname{argmax} \hat{f}_n(x_n)\right).$$

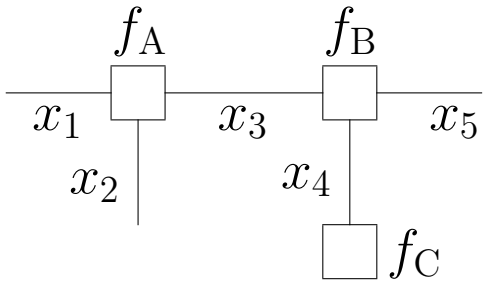
Recall:

Factor Graphs

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

Example:

$$f(x_1, x_2, x_3, x_4, x_5) = f_{\mathrm{A}}(x_1, x_2, x_3) \cdot f_{\mathrm{B}}(x_3, x_4, x_5) \cdot f_{\mathrm{C}}(x_4).$$

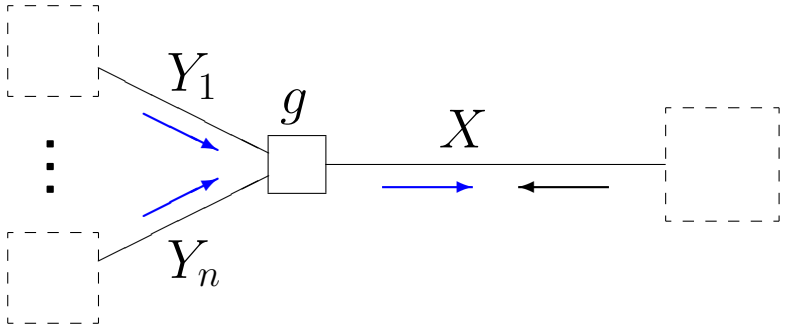


Rules:

- 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.

Recall:

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

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

Outline

Scalar Gaussian Messages

Messages of the form

$$\mu(x) \propto e^{-(x-m)^2/2\sigma^2}.$$

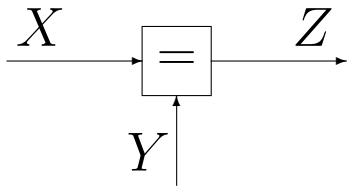
Arrow notation:

$\overrightarrow{\mu}_X$ parameterized by mean $\overrightarrow{m}_X$ and variance $\overrightarrow{\sigma}^2_X$;

$\overleftarrow{\mu}_X$ parameterized by mean $\overleftarrow{m}_X$ and variance $\overleftarrow{\sigma}^2_X$.

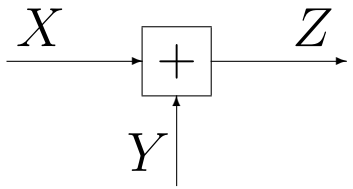
Scalar Gaussian Message Computation Rules

Sum-product rule and max-product rule coincide.



$$1/\overrightarrow{\sigma}_Z^2 = 1/\overrightarrow{\sigma}_X^2 + 1/\overrightarrow{\sigma}_Y^2$$

$$\overrightarrow{m}_Z/\overrightarrow{\sigma}_Z^2 = \overrightarrow{m}_X/\overrightarrow{\sigma}_X^2 + \overrightarrow{m}_Y/\overrightarrow{\sigma}_Y^2$$

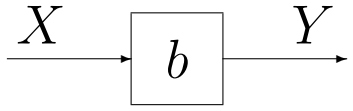


$$\overrightarrow{m}_Z = \overrightarrow{m}_X + \overrightarrow{m}_Y$$

$$\overleftarrow{m}_X = \overleftarrow{m}_Z - \overrightarrow{m}_Y$$

$$\overrightarrow{\sigma}_Z^2 = \overrightarrow{\sigma}_X^2 + \overrightarrow{\sigma}_Y^2$$

$$\overleftarrow{\sigma}_X^2 = \overleftarrow{\sigma}_Z^2 + \overrightarrow{\sigma}_Y^2$$



$$\overrightarrow{m}_Y = b\,\overrightarrow{m}_X$$

$$\overrightarrow{\sigma}_Y^2 = b^2\,\overrightarrow{\sigma}_X^2$$

Example:

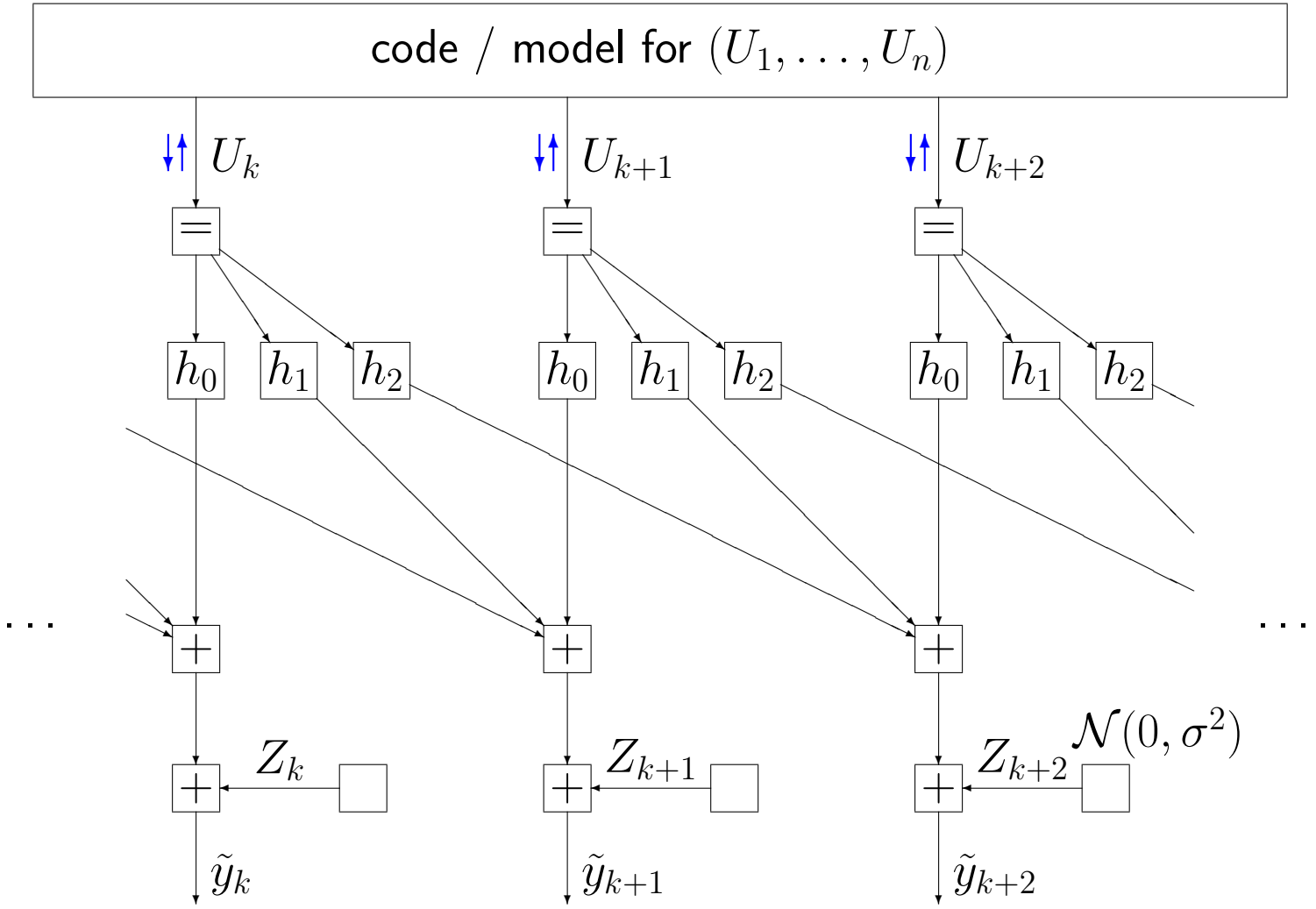
Equalization

$$\tilde{Y}_k = \sum_{\ell=0}^{M} h_\ell U_{k-\ell} + Z_k,$$

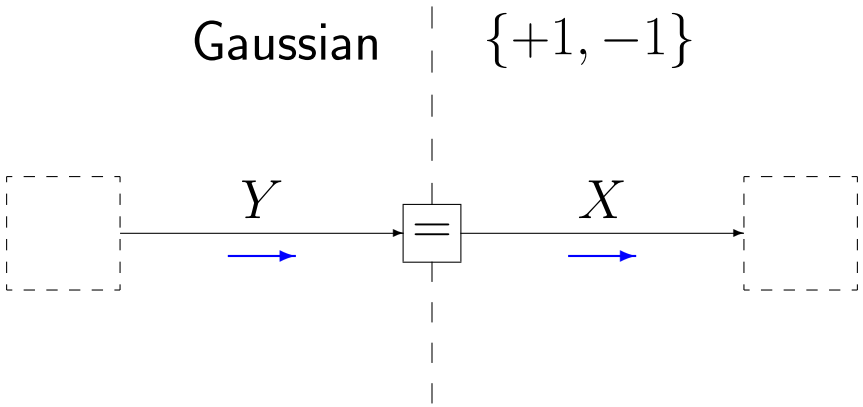
with

U_k	transmitted symbols (real)
$h_0, \ldots, h_M$	known channel coefficients (real)
Z_k	white Gaussian noise
$\tilde{Y}_k = \tilde{y}_k$	observed noisy channel output (real)

Scalar Factor Graph for Equalization for $M = 2$:



Gaussian $\longrightarrow$ Binary

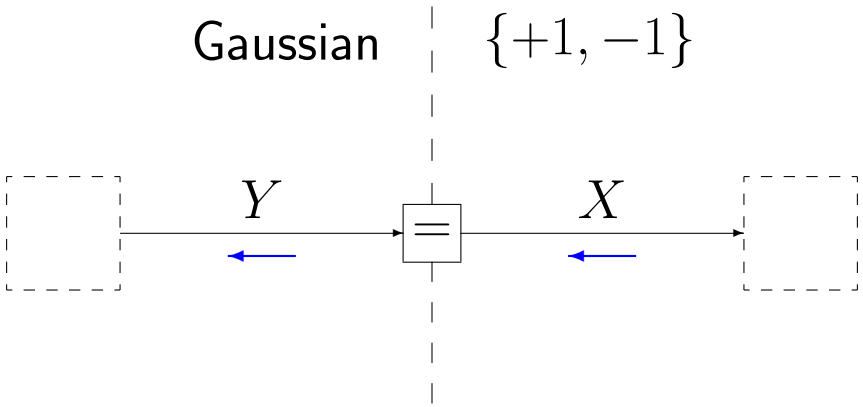


$$\overrightarrow{\mu}_X(x) = \int_y \overrightarrow{\mu}_Y(y)\,\delta(x-y)\,dy = \overrightarrow{\mu}_Y(x)$$

and thus

$$\begin{aligned}\overrightarrow{L}_X &\stackrel{\triangle}{=} \ln \frac{\overrightarrow{\mu}_X(+1)}{\overrightarrow{\mu}_X(-1)} \\ &= 2\overrightarrow{m}_Y/\overrightarrow{\sigma}_Y^2.\end{aligned}$$

Gaussian ← Binary



Obvious approach: matching mean and variance:

$$\overleftarrow{m}_Y = \overleftarrow{m}_X = \frac{\overleftarrow{\mu}_X(+1) - \overleftarrow{\mu}_X(-1)}{\overleftarrow{\mu}_X(+1) + \overleftarrow{\mu}_X(-1)}$$

$$\overleftarrow{\sigma}^2_Y = \overleftarrow{\sigma}^2_X = 1 - \overleftarrow{m}^2_X.$$

Vector Gaussian Messages

Messages of the form

$$\mu(x) \propto \exp\left(-\beta(x-m)^H W(x-m)\right)$$

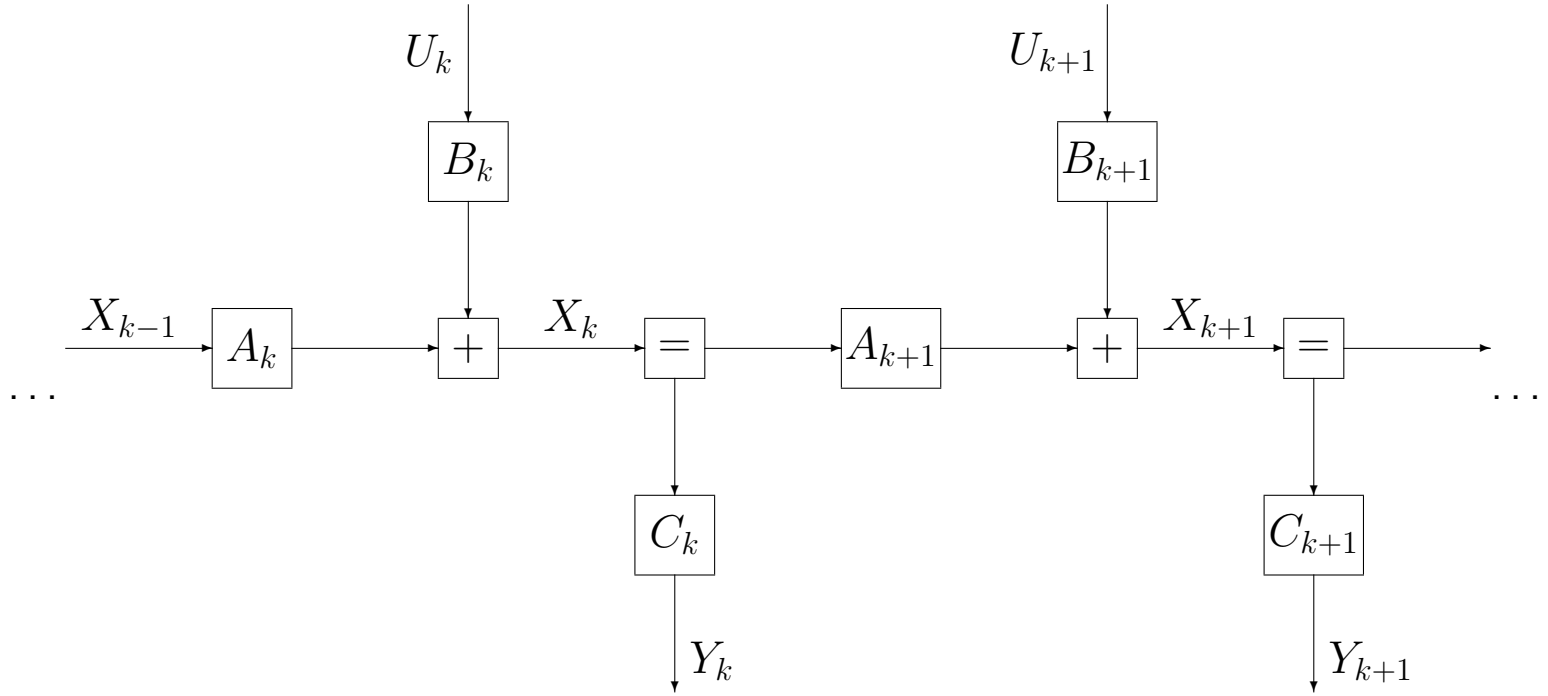
with mean vector m and positive (semi-) definite weight matrix W;
$\beta = 1/2$ in the real case and $\beta = 1$ in the complex case.

$(\cdot)^H$ denotes Hermitian transposition (complex conjugate of transpose)

Messages will be parameterized

- either by mean vector m and covariance matrix $V \stackrel{\triangle}{=} W^{-1}$
- or by W and Wm.

General Linear State Space Model



$$
\begin{aligned}
X_k &= A_k X_{k-1} + B_k U_k \\
Y_k &= C_k X_k
\end{aligned}
$$

Example: **Equalization**

$$\tilde{Y}_k = \sum_{\ell=0}^{M} h_\ell U_{k-\ell} + Z_k,$$

with

U_k	real-valued transmitted symbols
$h_0, \ldots, h_M$	known real channel coefficients
Z_k	white Gaussian noise

State space formulation:

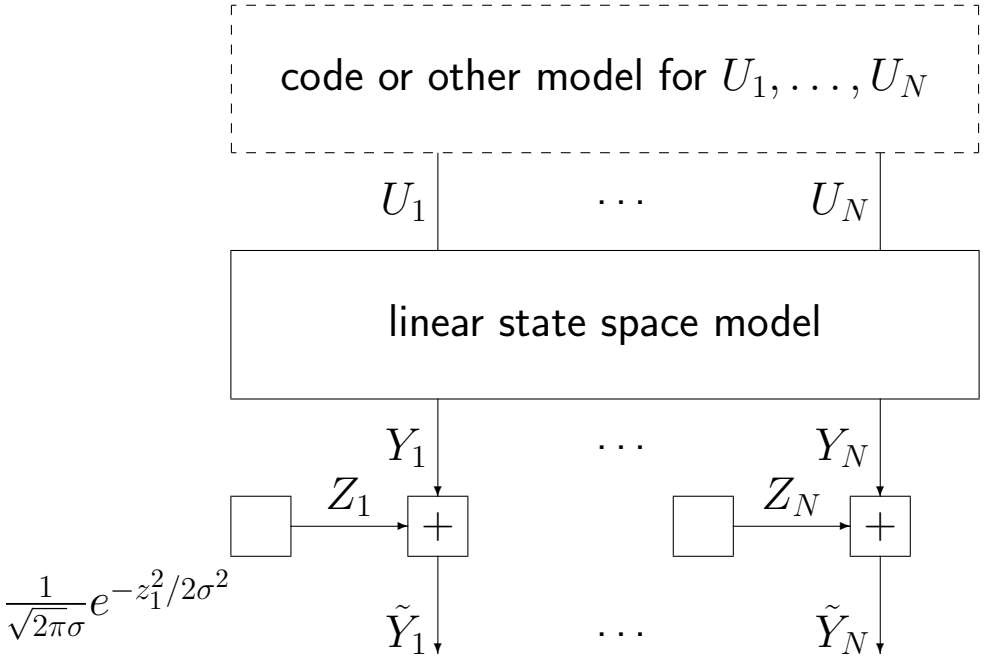
$$\begin{aligned} X_k &= A_k X_{k-1} + B_k U_k \\ Y_k &= C_k X_k \end{aligned}$$

with

$X_k \triangleq (U_k, \ldots, U_{k-M})^T$
$B_k \triangleq (1, 0, \ldots, 0)^T$
$C_k \triangleq (h_0, h_1, \ldots, h_M)$
$Y_k \triangleq \tilde{Y}_k - Z_k$ (noise-free output)

$$A_k \triangleq \begin{pmatrix} 0 & 0 \\ I_M & 0 \end{pmatrix}$$

Equalization: Cycle-Free Factor Graph



Forward-backward Gaussian message passing

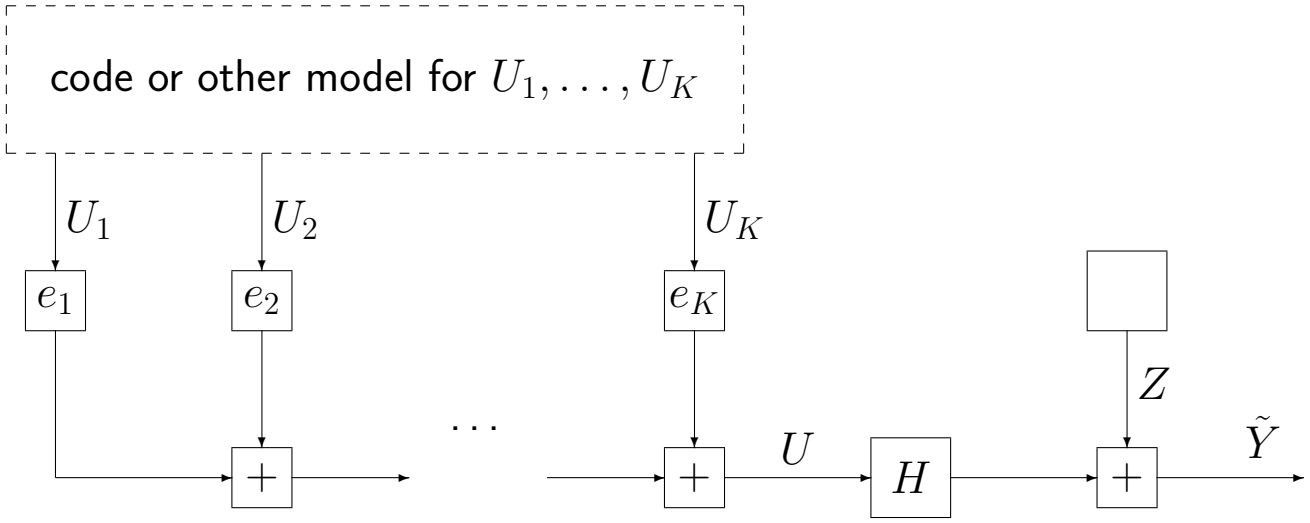
$\Longrightarrow$ LMMSE/Kalman equalization (many versions!)

Example: **Multi-User Communication** (Wang/Poor, Boutros/Caire)

$$\tilde{Y} = HU + Z$$

with

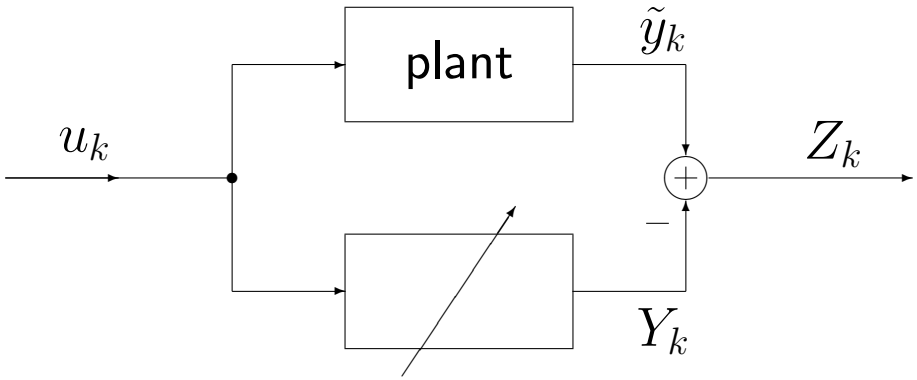
$U = (U_1, \ldots, U_K)^T$ signals transmitted by K-users
$\tilde{Y} = (\tilde{Y}_1, \ldots, \tilde{Y}_N)^T$ received signal
$Z = (Z_1, \ldots, Z_N)^T$ Gaussian noise
$H \in \mathbb{R}^{N \times K}$



with $e_1 \triangleq (1, 0, \ldots, 0)^T$, $e_2 \triangleq (0, 1, 0, \ldots, 0)^T$, etc.

A nonstochastic example:

FIR Filter Adaptation



FIR filter: $Y_k = \sum_{\ell=0}^{M} H_\ell u_{k-\ell}$.

Error: $Z_k \stackrel{\triangle}{=} \tilde{y}_k - Y_k$.

Known:

– input u_k, $k = -M+1, -M+2, \ldots, N$

– output $\tilde{y}_k$, $k = 1, 2, \ldots, N$.

Problem:

find coefficients $H_0, \ldots, H_M$ that minimize $\sum_{k=1}^{N} Z_k^2$.

FIR Filter Adaptation cont'd

Equivalent problem formulation:
Maximize

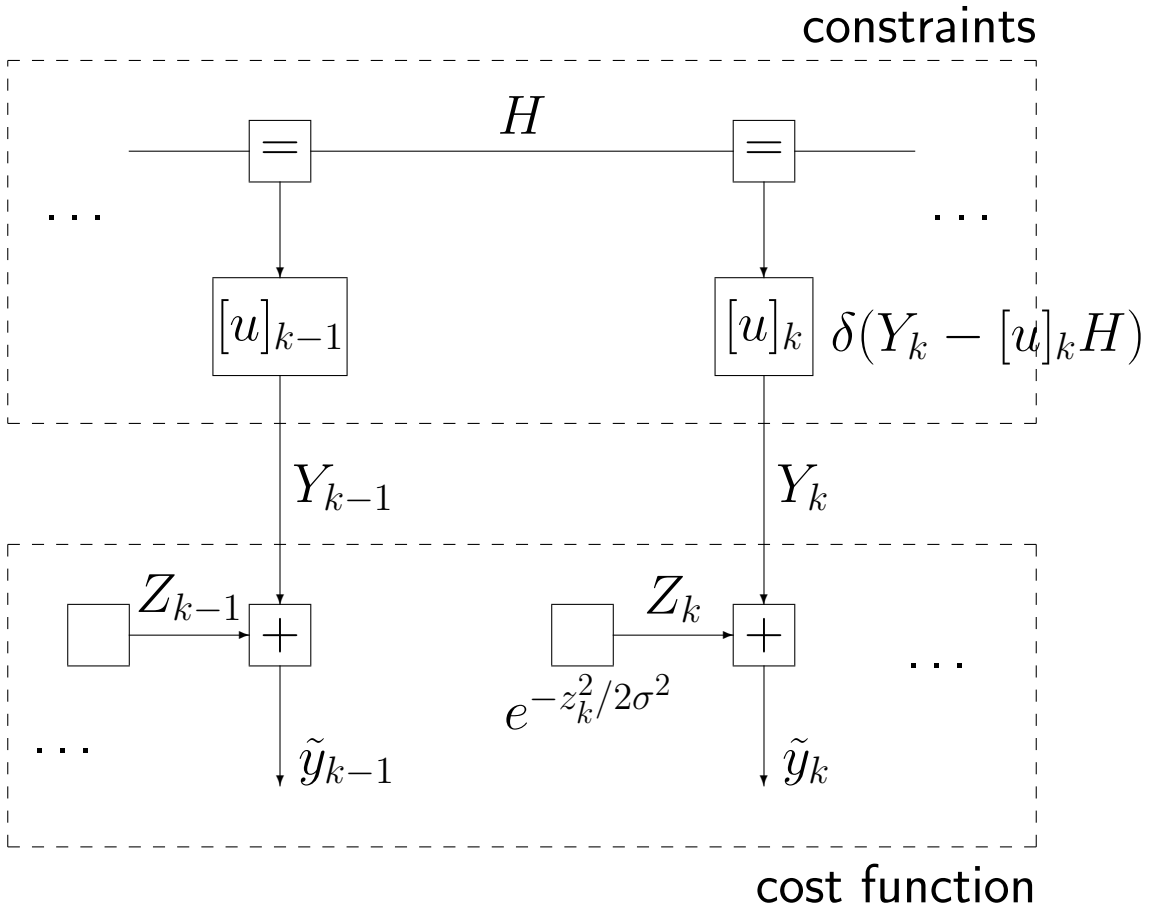
$$\prod_{k=1}^{N} e^{-Z_k^2/2\sigma^2} = e^{-\sum_{k=1}^{N} Z_k^2/2\sigma^2}$$

(for arbitrary $\sigma > 0$) subject to the constraints

$$Y_k = [u]_k H$$
$$Z_k = \tilde{y}_k - Y_k$$

with column vector $H \stackrel{\triangle}{=} (H_0, \ldots, H_M)^T$
and row vector $[u]_k \stackrel{\triangle}{=} (u_k, u_{k-1}, \ldots, u_{k-M})$.

FIR Filter Adaptation: Factor Graph



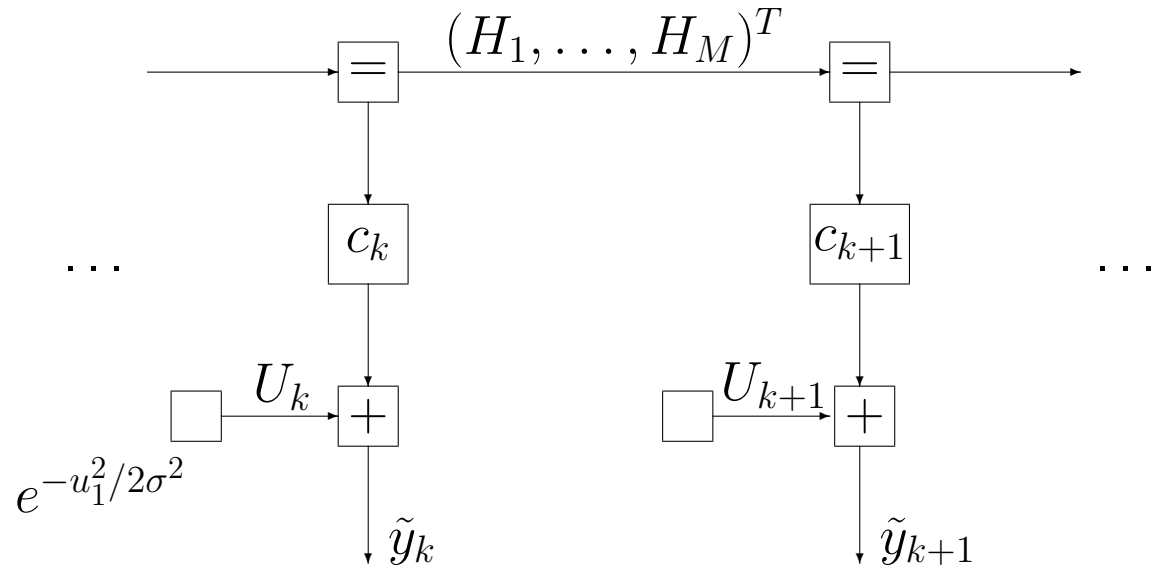
Problem: maximization of global function over all configurations.

Adaptive Autoregressive Filter—LPC Analysis

For known signal $\tilde{y}_k$ with

$$\tilde{y}_k = \sum_{\ell=1}^{M} H_\ell \tilde{y}_{k-\ell} + u_k$$

$k = 1, 2, \ldots$, determine filter coefficients $H_1, \ldots, H_M$ to minimize $\sum_k u_k^2$ or, equivalently, to maximize $\prod_k e^{-u_k^2/2\sigma^2}$.



with $c_k \stackrel{\triangle}{=} (\tilde{y}_{k-1}, \ldots, \tilde{y}_{k-M})$.

Gaussian Message Passing in Linear Models

encompasses much of classical signal processing and appears often as a component of more complex problems/algorithms.

Note:

1. Gaussianity of messages is preserved in linear models.
2. Includes Kalman filtering and recursive least-squares algorithms.
3. For Gaussian messages, the sum-product (integral-product) algorithm coincides with the max-product algorithm.
4. For jointly Gaussian random variables,
   MAP estimation $=$ MMSE estimation $=$ LMMSE estimation.
5. Even if X and Y are not jointly Gaussian, the LMMSE estimate of X from $Y = y$ may be obtained by pretending that X and Y are jointly Gaussian (with their actual means and second-order moments) and forming the corresponding MAP estimate.

Vector Gaussian Messages

Messages of the form

$$\mu(x) \propto \exp\left(-\beta(x-m)^H W(x-m)\right)$$

with mean vector m and positive (semi-) definite weight matrix W;
$\beta = 1/2$ in the real case and $\beta = 1$ in the complex case.

$(\cdot)^H$ denotes Hermitian transposition (complex conjugate of transpose)

Messages will be parameterized

- either by mean vector m and covariance matrix $V \stackrel{\triangle}{=} W^{-1}$
- or by W and Wm.

Vector Gaussian Messages cont'd

Arrow notation:

Message $\overrightarrow{\mu}_X$ parameterized by $\overrightarrow{m}_X$ and $\overrightarrow{V}_X$ or by $\overrightarrow{W}_X$ and $\overrightarrow{W}_X\overrightarrow{m}_X$.

Marginal: $\overrightarrow{\mu}_X(x)\overleftarrow{\mu}_X(x)$ is the Gaussian with mean m_X and covariance matrix $V_X = W_X^{-1}$ given by

$$W_X = \overrightarrow{W}_X + \overleftarrow{W}_X$$

$$W_X m_X = \overrightarrow{W}_X\overrightarrow{m}_X + \overleftarrow{W}_X\overleftarrow{m}_X.$$

Sometimes also useful:

$$\tilde{W}_X \stackrel{\triangle}{=} (\overrightarrow{V}_X + \overleftarrow{V}_X)^{-1},$$

which is dual to

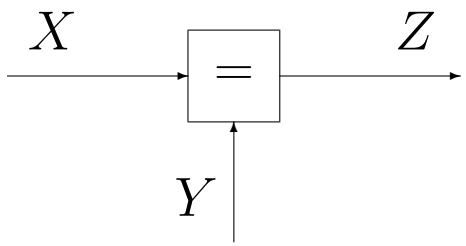
$$V_X = W_X^{-1} = (\overrightarrow{W}_X + \overleftarrow{W}_X)^{-1}.$$

Transformations Among Single-Edge Quantities

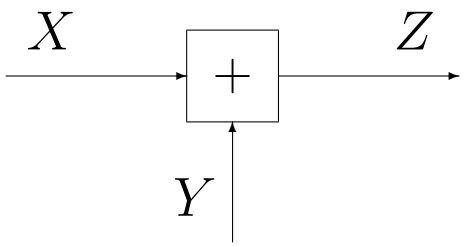
$$
\begin{aligned}
\tilde{W}_X &= \overrightarrow{W}_X V_X \overleftarrow{W}_X \\
&= \overrightarrow{W}_X - \overrightarrow{W}_X V_X \overrightarrow{W}_X \\
V_X &= \overrightarrow{V}_X \tilde{W}_X \overleftarrow{V}_X \\
&= \overrightarrow{V}_X - \overrightarrow{V}_X \tilde{W}_X \overrightarrow{V}_X \\
m_X &= V_X \overrightarrow{W}_X \overrightarrow{m}_X + V_X \overleftarrow{W}_X \overleftarrow{m}_X \\
&= \overrightarrow{m}_X - \overrightarrow{V}_X \tilde{W}_X \overrightarrow{m}_X + V_X \overleftarrow{W}_X \overleftarrow{m}_X .
\end{aligned}
$$

The direction of the arrows may be reversed in all these relations.

Computation Rules for Gaussian Messages I

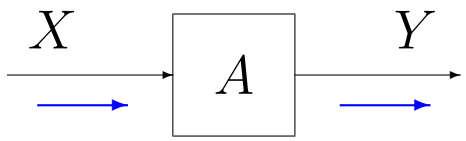


$$\overrightarrow{W}_Z = \overrightarrow{W}_X + \overrightarrow{W}_Y$$
$$\overleftarrow{W}_X = \overleftarrow{W}_Z + \overrightarrow{W}_Y$$
$$\overrightarrow{W}_Z \overrightarrow{m}_Z = \overrightarrow{W}_X \overrightarrow{m}_X + \overrightarrow{W}_Y \overrightarrow{m}_Y$$
$$\overleftarrow{W}_X \overleftarrow{m}_X = \overleftarrow{W}_Z \overleftarrow{m}_Z + \overrightarrow{W}_Y \overrightarrow{m}_Y$$
$$m_X = m_Y = m_Z$$
$$V_X = V_Y = V_Z$$

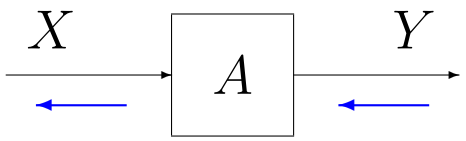


$$\overrightarrow{V}_Z = \overrightarrow{V}_X + \overrightarrow{V}_Y$$
$$\overleftarrow{V}_X = \overleftarrow{V}_Z + \overrightarrow{V}_Y$$
$$\overrightarrow{m}_Z = \overrightarrow{m}_X + \overrightarrow{m}_Y$$
$$\overleftarrow{m}_X = \overleftarrow{m}_Z - \overrightarrow{m}_Y$$
$$m_X + m_Y - m_Z = 0$$
$$\tilde{W}_X = \tilde{W}_Y = \tilde{W}_Z$$

Computation Rules for Gaussian Messages II



$$\overrightarrow{V}_Y = A\overrightarrow{V}_X A^H$$
$$\overrightarrow{m}_Y = A\overrightarrow{m}_X$$
$$m_Y = Am_X$$
$$V_Y = AV_X A^H$$



$$\overleftarrow{W}_X = A^H\overleftarrow{W}_Y A$$
$$\overleftarrow{W}_X\overleftarrow{m}_X = A^H\overleftarrow{W}_Y\overleftarrow{m}_Y$$
$$\tilde{W}_X m_X = A^H\tilde{W}_Y m_Y$$
$$\tilde{W}_X = A^H\tilde{W}_Y A$$
$$\tilde{W}_X\overleftarrow{m}_X = A^H\tilde{W}_Y\overleftarrow{m}_Y$$

Message computations for vector Gaussians:

More to Come

Using only the basic message computation rules may require frequent transformations of W into $V = W^{-1}$ and vice versa.

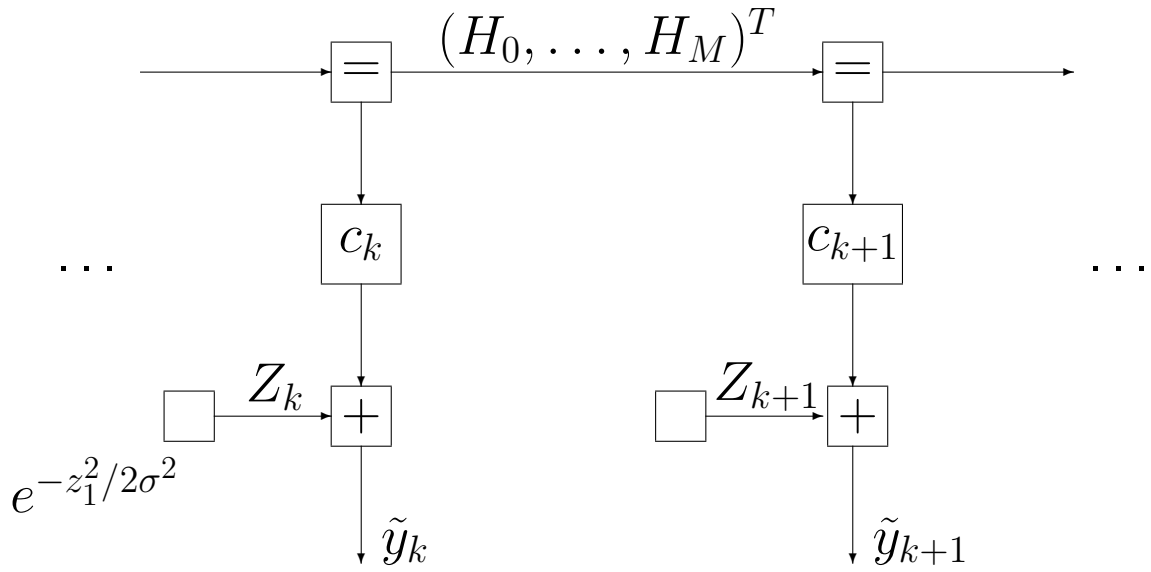
Such matrix inversions can often be avoided by using message computation rules for composite blocks.

Recall example: **Adaptive FIR Filter**

For known inputs u_k and known noisy outputs $\tilde{y}_k$ with

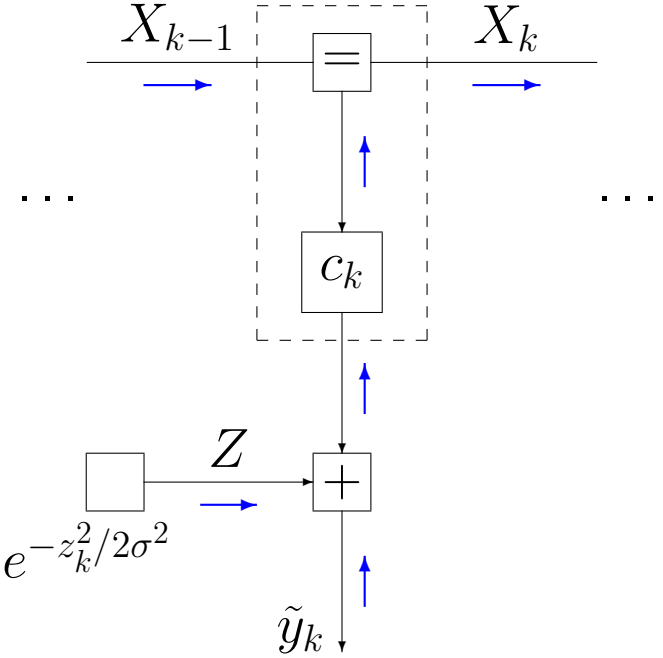
$$\tilde{y}_k = \sum_{\ell=0}^{M} H_\ell u_{k-\ell} + z_k$$

$k = 1, 2, \ldots$, determine filter coefficients $H_0, \ldots, H_M$ to minimize $\sum_k z_k^2$ or, equivalently, to maximize $\prod_k e^{-z_k^2/2\sigma^2}$.



with $c_k \stackrel{\triangle}{=} (u_k, u_{k-1}, \ldots, u_{k-M})$.

Adaptive FIR Filter: RLS Algorithm



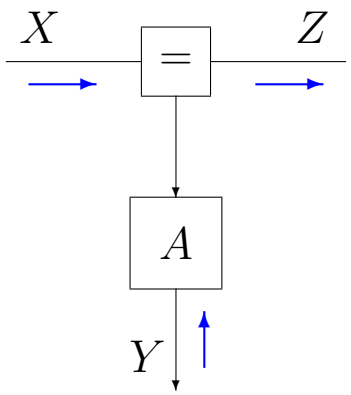
Two versions:

- Forward-only recursion with $\overrightarrow{W}_{X_k}$ and $\overrightarrow{W}_{X_k}\overrightarrow{m}_{X_k}$.
- Forward-only recursion with $\overrightarrow{V}_{X_k}$ and $\overrightarrow{m}_{X_k}$.

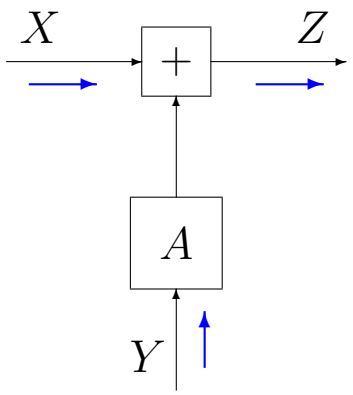
Extra twist of classical RLS algorithm:
at each step, multiply $\overrightarrow{V}_{X_k}$ with forgetting factor $\gamma > 1$, $\gamma \approx 1$.

Computation Rules for Gaussian Messages III

(The Kalman trick, based on the Matrix Inversion Lemma.)



$$\overrightarrow{m}_Z = \overrightarrow{m}_X + \overrightarrow{V}_X A^H G \left(\overleftarrow{m}_Y - A\overrightarrow{m}_X\right)$$
$$\overrightarrow{V}_Z = \overrightarrow{V}_X - \overrightarrow{V}_X A^H G A \overrightarrow{V}_X$$
$$\text{with } G \triangleq \left(\overleftarrow{V}_Y + A\overrightarrow{V}_X A^H\right)^{-1}$$

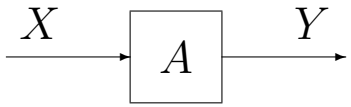


$$\overrightarrow{m}_Z = \overrightarrow{m}_X + A\overrightarrow{m}_Y$$
$$\overleftarrow{m}_X = \overleftarrow{m}_Z - A\overrightarrow{m}_Y$$
$$\overrightarrow{W}_Z = \overrightarrow{W}_X - \overrightarrow{W}_X A H A^H \overrightarrow{W}_X$$
$$\text{with } H \triangleq \left(\overrightarrow{W}_Y + A^H \overrightarrow{W}_X A\right)^{-1}$$
$$\overrightarrow{W}_Z \overrightarrow{m}_Z = \overrightarrow{W}_X \overrightarrow{m}_X + \overrightarrow{W}_X A H \cdot \left(\overrightarrow{W}_Y \overrightarrow{m}_Y - A^H \overrightarrow{W}_X \overrightarrow{m}_X\right)$$

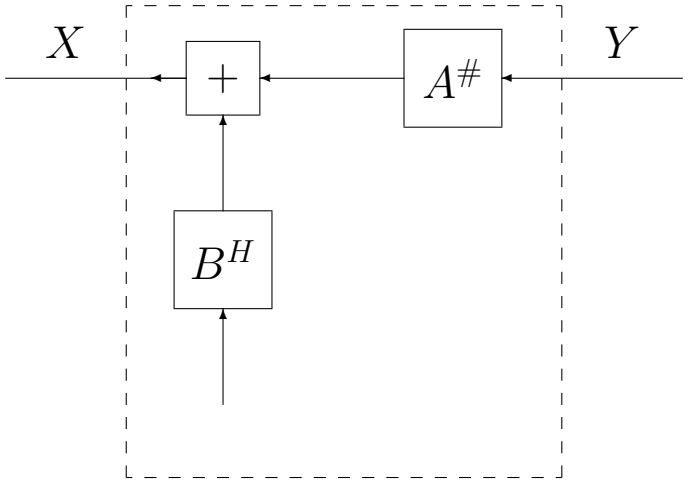
No matrix inversion if Y is scalar!

Reversing a Matrix Multiplication I

If $\operatorname{rank} A =$ number of rows $<$ number of columns:



is equivalent to:

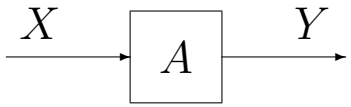


where $A^{\#} = A^H(AA^H)^{-1}$ and where B is a matrix such that $\begin{pmatrix} A \\ B \end{pmatrix}$ is a nonsingular square matrix and $AB^H = 0$.

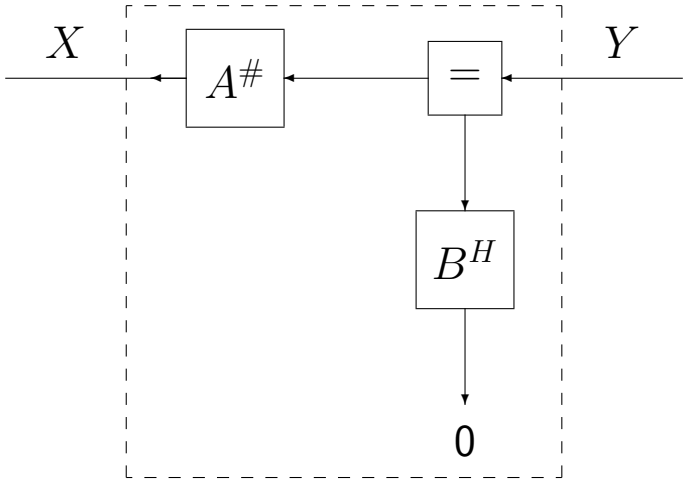
Allows to propagate $\overrightarrow{W}$ and $\overrightarrow{W}\overrightarrow{m}$.

Reversing a Matrix Multiplication II

If $\operatorname{rank} A =$ number of columns $<$ number of rows:



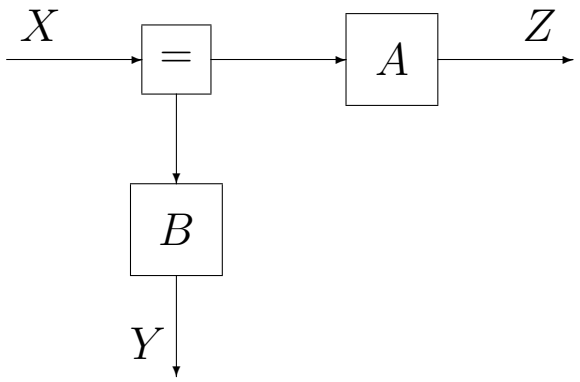
is equivalent to:



where $A^{\#} = (A^H A)^{-1} A^H$ and where B is a matrix such that (A, B) is a nonsingular square matrix and $B^H A = 0$.

Allows to propagate $\overleftarrow{V}$ and $\overleftarrow{m}$.

Nonsingular Matrix from Rectangular Matrices I

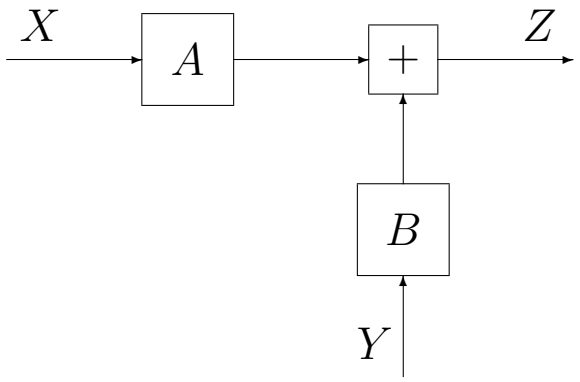


If $\begin{pmatrix} A \\ B \end{pmatrix}$ is a nonsingular square matrix:

$$\overleftarrow{m}_X = \begin{pmatrix} A \\ B \end{pmatrix}^{-1} \begin{pmatrix} \overleftarrow{m}_Z \\ \overleftarrow{m}_Y \end{pmatrix}$$

$$\overleftarrow{V}_X = \begin{pmatrix} A \\ B \end{pmatrix}^{-1} \begin{pmatrix} \overleftarrow{V}_Z & 0 \\ 0 & \overleftarrow{V}_Y \end{pmatrix} \left(A^H, B^H\right)^{-1}$$

Nonsingular Matrix from Rectangular Matrices II

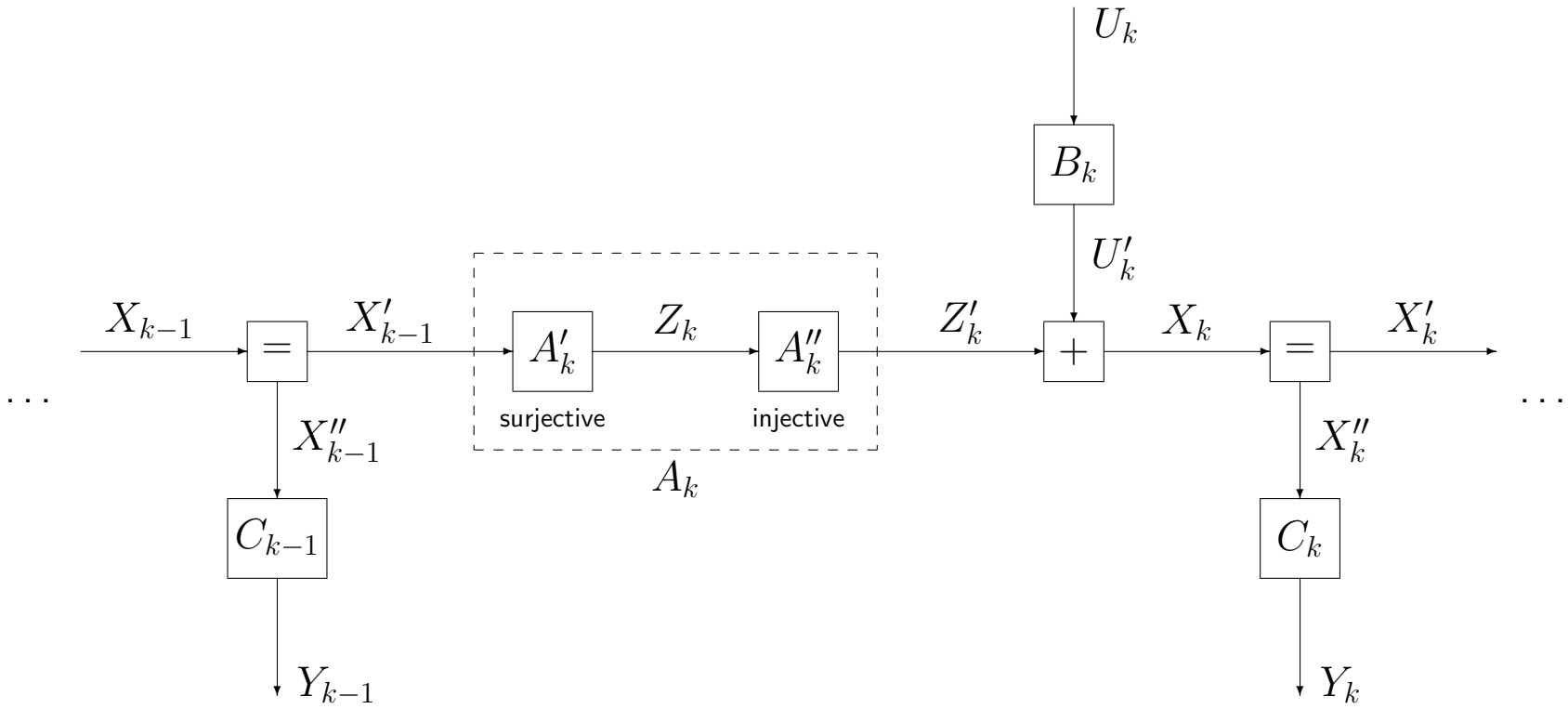


If (A, B) is a nonsingular square matrix:

$$\overrightarrow{W}_Z \overrightarrow{m}_Z = \begin{pmatrix} A^H \\ B^H \end{pmatrix}^{-1} \begin{pmatrix} \overrightarrow{W}_X \overrightarrow{m}_X \\ \overrightarrow{W}_Y \overrightarrow{m}_Y \end{pmatrix}$$

$$\overrightarrow{W}_Z = \begin{pmatrix} A^H \\ B^H \end{pmatrix}^{-1} \begin{pmatrix} \overrightarrow{W}_X & 0 \\ 0 & \overrightarrow{W}_Y \end{pmatrix} (A, B)^{-1}$$

General Linear State Space Model (again)



Can now propagate also $\overleftarrow{V}$ and $\overleftarrow{m}$ as well as $\overrightarrow{W}$ and $\overrightarrow{W}\overrightarrow{m}$ without matrix inversion (if U_k and Y_k are scalars).

Intermediate Conclusion

- The classical topic of Gaussian models (including least squares) with linear constraints is naturally developed in the factor graph setting.
- Much knowledge about algorithms may be expressed as "local" message computation rules or as "local" manipulations of nodes/factors.
- Algorithms can be composed from tabulated message computation rules.

Outline

The Factor Graph Approach to Signal Processing

1. Choose a factor graph to represent the system model.

   Options include:

   - Choice of auxiliary variables (state variables).
   - Grouping and splitting of variables and factors.
   - Adding redundant constraints.

2. Choose the message types and suitable message computation rules. Use and maintain tables of such computation rules.

3. Choose a message update schedule.

   For graphs with cycles, we obtain iterative algorithms, often without performance guarantee, but often with outstanding performance vs. complexity tradeoff.

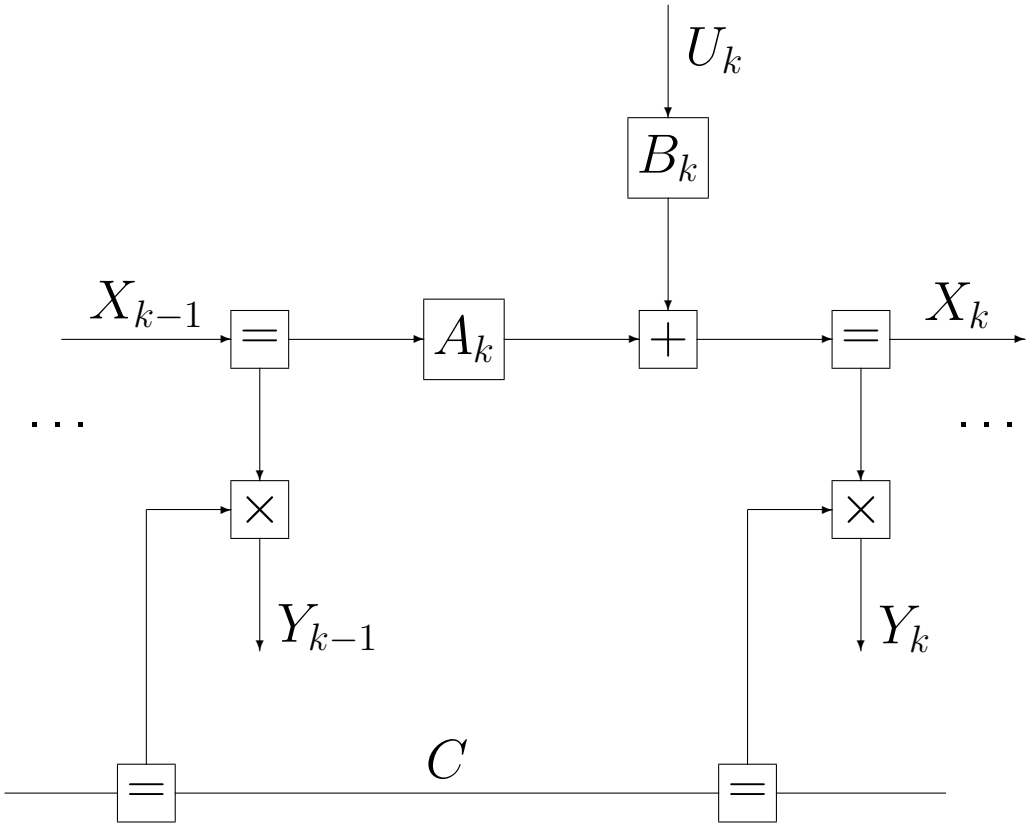
Continuous Variables: Message Types

The following message types are widely applicable.

- Quantization of messages into discrete bins. Infeasible in higher dimensions.
- Single point: the message $\mu(x)$ is replaced by a temporary or final estimate $\hat{x}$.
- Function value and gradient at a point selected by the receiving node. Allows steepest-ascent (descent) algorithms.
- Gaussians. Works for Kalman filtering.
- Gaussian mixtures.
- List of samples: a pdf can be represented by a list of samples. This data type is the basis of particle filters, but it can be used as a general data type in a graphical model.
- Compound messages: the "product" of other message types.

Example:

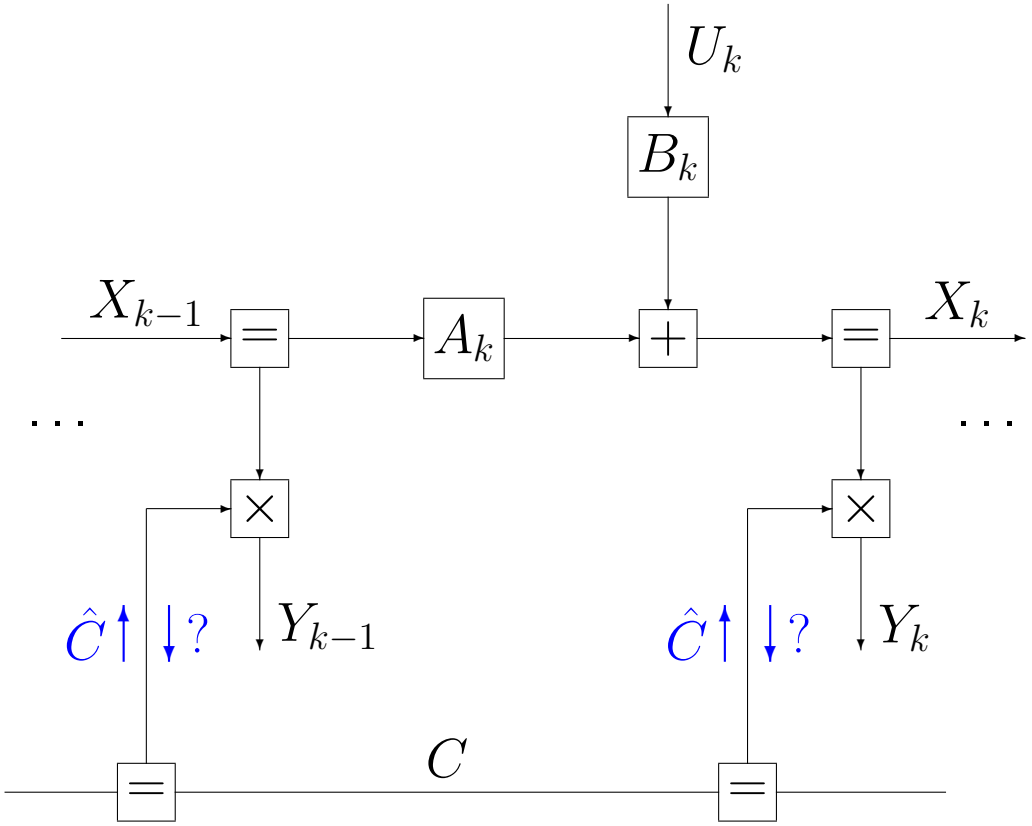
Linear State Space Model with Unknown Vector C



How cope with the multiplier node?

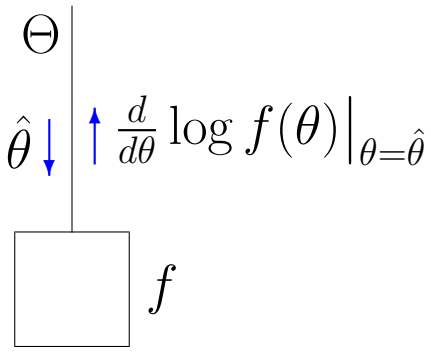
Example cont'd:

Popular Approach: **Tentative Decision** for coefficient(s)



What about the other direction?

Steepest Ascent



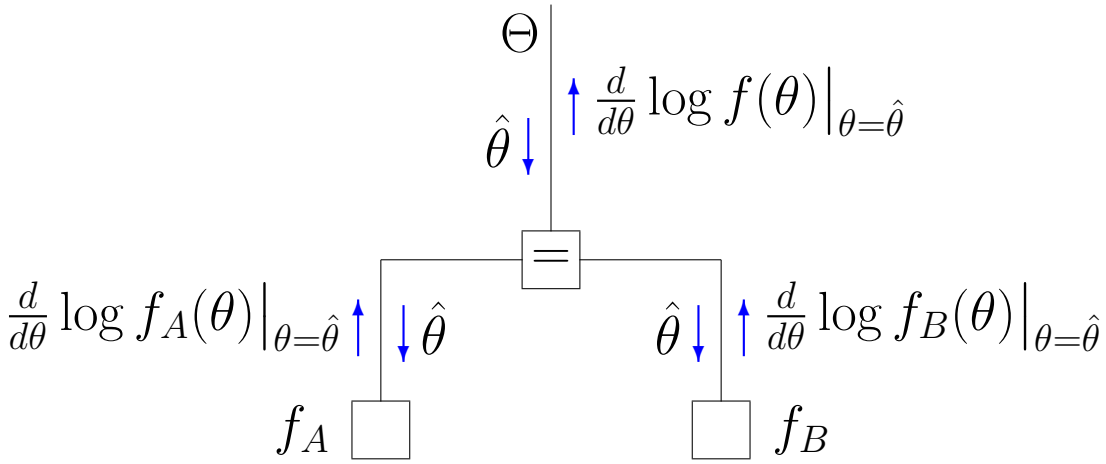
Try to find $\operatorname{argmax}_\theta f(\theta)$ by the iteration

$$\hat{\theta}_{\text{new}} = \hat{\theta}_{\text{old}} + s \cdot \frac{d}{d\theta} \log f(\theta)\Big|_{\theta=\hat{\theta}_{\text{old}}}$$

where s is a positive step size parameter.

Steepest Ascent as Message Passing

Example: $f(\theta) = f_A(\theta) f_B(\theta)$.



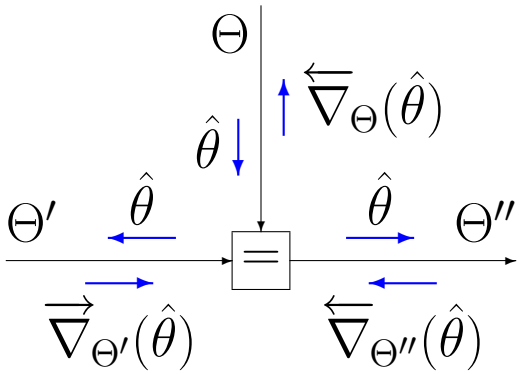
$$\frac{d}{d\theta} \log\big(f_A(\theta) f_B(\theta)\big) = \frac{d}{d\theta} \log f_A(\theta) + \frac{d}{d\theta} \log f_B(\theta)$$

Gradient Messages: Equality Constraint Node

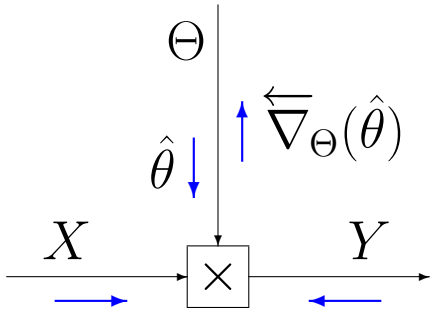
Using the notation

$$\overleftarrow{\nabla}_{\Theta}(\theta) \triangleq \frac{d}{d\theta} \log \overleftarrow{\mu}_{\Theta}(\theta)$$

where $\overleftarrow{\mu}_{\Theta}$ is the sum-product message: $\overleftarrow{\nabla}_{\Theta}(\hat{\theta}) = \overrightarrow{\nabla}_{\Theta'}(\hat{\theta}) + \overleftarrow{\nabla}_{\Theta''}(\hat{\theta})$



Gradient Messages: Multiplier Node



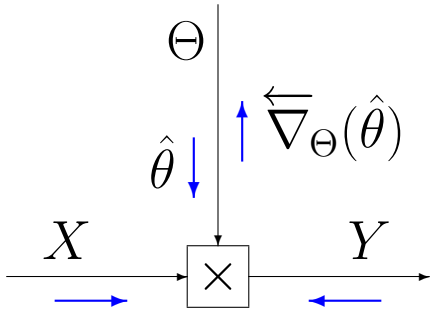
Recall:

$$\overleftarrow{\nabla}_\Theta(\theta) \stackrel{\triangle}{=} \frac{d}{d\theta} \log \overleftarrow{\mu}_\Theta(\theta)$$

with sum-product message

$$\begin{aligned}\overleftarrow{\mu}_\Theta(\theta) &= \int_x \int_y \overrightarrow{\mu}_X(x) \overleftarrow{\mu}_Y(y) \delta(y - \theta x)\, dx\, dy \\ &= \int_x \overrightarrow{\mu}_X(x) \overleftarrow{\mu}_Y(\theta x)\, dx.\end{aligned}$$

Gradient Messages: Multiplier Node cont'd



If both $\overrightarrow{\mu}_X$ and $\overleftarrow{\mu}_Y$ are Gaussians (scalar case):

$$\overleftarrow{\nabla}_\Theta(\hat{\theta}) = \frac{\overleftarrow{m}_Y}{\overleftarrow{\sigma}_Y^2}\tilde{m} - \frac{\hat{\theta}}{\overleftarrow{\sigma}_Y^2}\left(\tilde{\sigma}^2 + \tilde{m}^2\right)$$

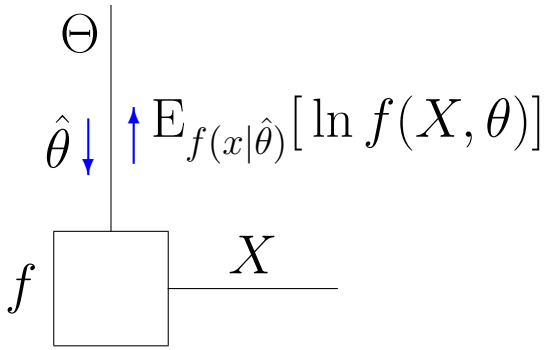
with $\tilde{m}$ and $\tilde{\sigma}^2$ given by

$$\frac{1}{\tilde{\sigma}^2} = \frac{1}{\overrightarrow{\sigma}_X^2} + \frac{\hat{\theta}^2}{\overleftarrow{\sigma}_Y^2}$$

and

$$\frac{\tilde{m}}{\tilde{\sigma}^2} = \frac{\overrightarrow{m}_X}{\overrightarrow{\sigma}_X^2} + \frac{\hat{\theta}\overleftarrow{m}_Y}{\overleftarrow{\sigma}_Y^2}$$

Expectation Maximization



Let $\bar{f}(\theta) \stackrel{\triangle}{=} \int_x f(x,\theta)\,dx$ for some $f(x,\theta) > 0$.

Try to find $\operatorname{argmax}_\theta \bar{f}(\theta)$ by the iteration

$$\hat{\theta}_{\text{new}} = \operatorname*{argmax}_\theta \mathrm{E}_{f(x|\hat{\theta}_{\text{old}})}[\ln f(X,\theta)]$$

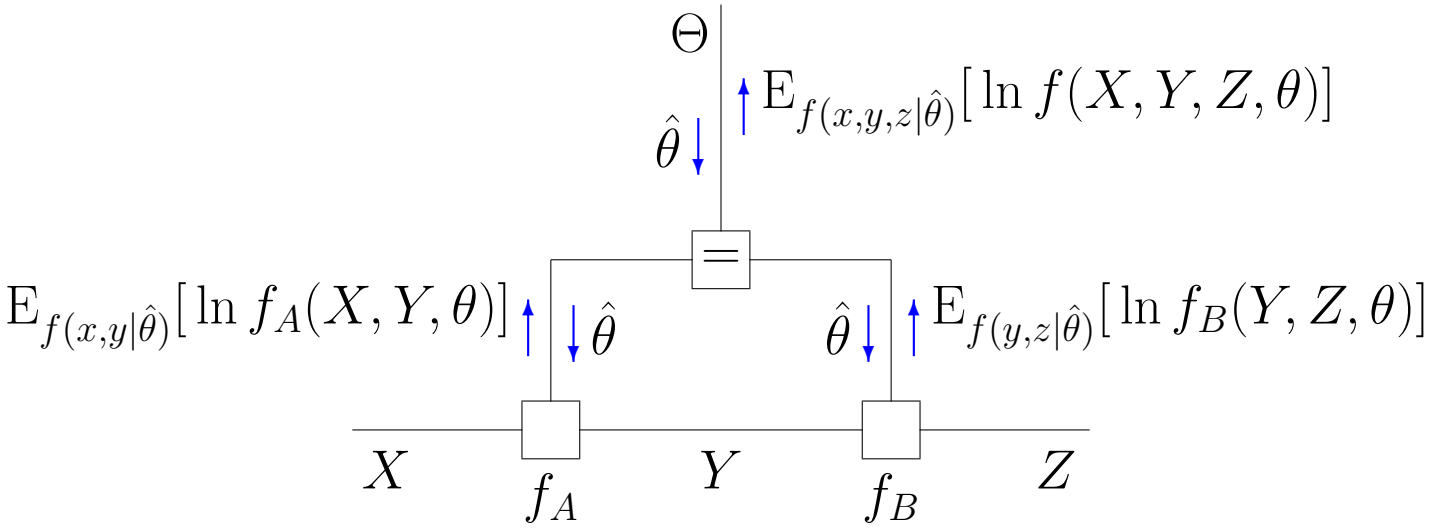
with

$$f(x|\theta) \stackrel{\triangle}{=} \frac{f(x,\theta)}{\int_x f(x,\theta)\,dx}$$

EM Theorem: $\bar{f}(\hat{\theta}_{\text{new}}) \geq \bar{f}(\hat{\theta}_{\text{old}})$.

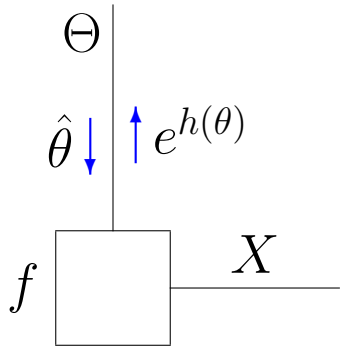
Expectation Maximization as Message Passing

Example: $f(x, y, z, \theta) = f_A(x, y, \theta) f_B(y, z, \theta)$.



$$\mathrm{E}_{f(x,y,z|\hat{\theta})}[\ln f(X, Y, Z, \theta)]$$
$$= \mathrm{E}_{f(x,y|\hat{\theta})}[\ln f_A(X, Y, \theta)] + \mathrm{E}_{f(y,z|\hat{\theta})}[\ln f_B(Y, Z, \theta)] .$$

Expectation Maximization cont'd



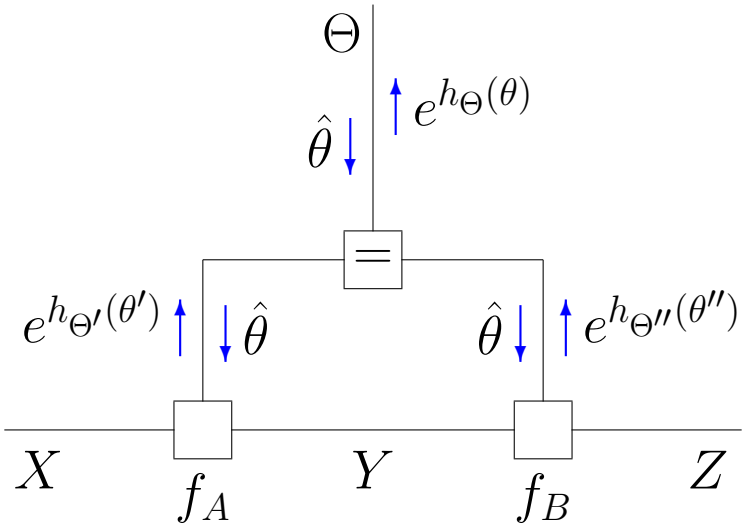
Try to find $\operatorname{argmax}_\theta \bar{f}(\theta)$ by the iteration

$$\hat{\theta}_{\text{new}} = \operatorname*{argmax}_\theta \underbrace{\mathrm{E}_{f(x|\hat{\theta}_{\text{old}})}[\ln f(X,\theta)]}_{h(\theta)}$$

$$= \operatorname*{argmax}_\theta e^{h(\theta)}.$$

Expectation Maximization cont'd

Example: $f(x, y, z, \theta) = f_A(x, y, \theta) f_B(y, z, \theta)$.

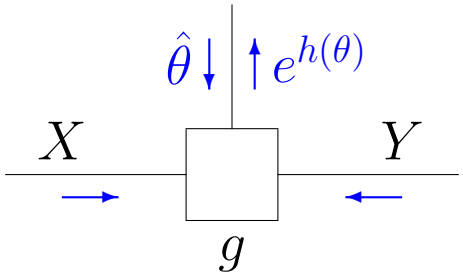


$$h_{\Theta'}(\theta) = \mathrm{E}_{f(x,y|\hat{\theta})}[\ln f_A(X, Y, \theta)] \qquad \text{etc.}$$

$$e^{h_\Theta(\theta)} = e^{h_{\Theta'}(\theta)} e^{h_{\Theta''}(\theta)},$$

consistent with factor graph interpretation (not just for "=" -node).

General Computation Rule for EM-Message



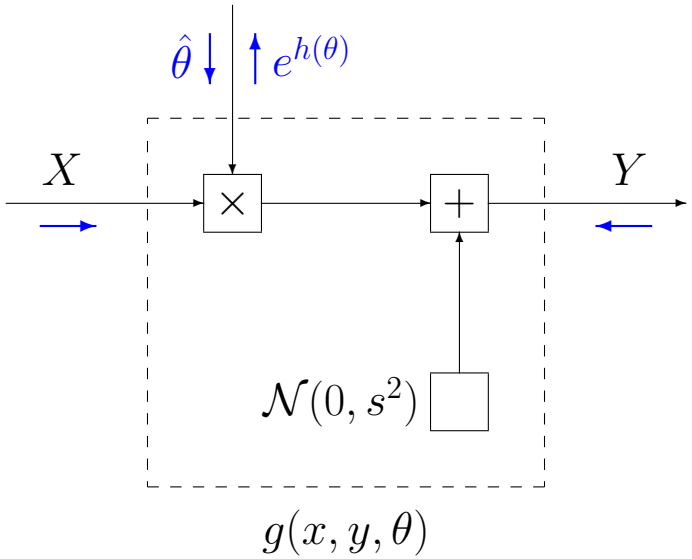
$$h(\theta) = \mathrm{E}_{p_{\text{local}}}\big[\log g(X, Y, \theta)\big]$$

with

$$p_{\text{local}}(x, y|\hat{\theta}) \propto g(x, y, \hat{\theta}) \underbrace{\overrightarrow{\mu}_X(x)\,\overleftarrow{\mu}_Y(y)}_{\text{sum-product messages}}$$

Looks complicated, but it sometimes yields nicer expressions than the sum-product rule.

EM Example: **Multiplier Node**

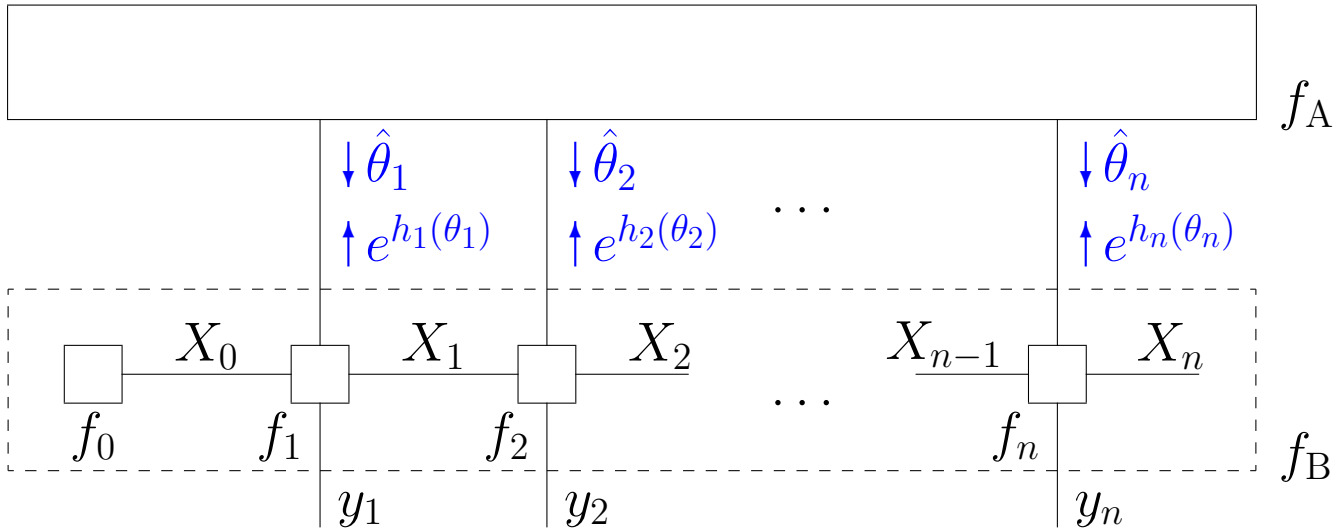


If both $\overrightarrow{\mu}_X$ and $\overleftarrow{\mu}_Y$ are Gaussians, then $e^{h(\theta)}$ is Gaussian with mean and variance (scalar case)

$$m_{\text{EM}} \triangleq \frac{\tilde{\sigma}_{X,Y} + \tilde{m}_X \tilde{m}_Y}{\tilde{\sigma}_X^2 + \tilde{m}_X^2} \qquad \text{and} \qquad \sigma_{\text{EM}}^2 \triangleq \frac{s^2}{\tilde{\sigma}_X^2 + \tilde{m}_X^2}$$

where $\tilde{m}_X$, $\tilde{\sigma}_X^2$, etc., are means and variances with respect to p_{local}.

More Complex Factor Graphs



1. Make some initial guesses $\hat{\theta}_k$.
2. Forward-backward sum-product sweep through f_{B}.
3. Compute upwards messages $e^{h(\theta_k)}$.
4. Compute new estimates $\hat{\theta}_k$ by max-product sweep through f_{A}.
5. Repeat 2–4 until convergence or until the available time is over.

EM theorem applies if edges Θ_k form a cut set.

EM as Message Passing: Benefits I

- EM can be used to break cycles in a factor graph (Eckford 2000, 2004).
- On the other hand, the EM messages can be used "locally", and may work just fine, if the "global" conditions for the EM theorem are not satisfied.
- The EM messages are "nice" expressions in some cases where the standard sum-product rule yields intractable expressions.
- As a full member of the family of message passing algorithms, it is easy to seamlessly combine EM with all the other message passing algorithms in interesting ways.

EM as Message Passing: Benefits II

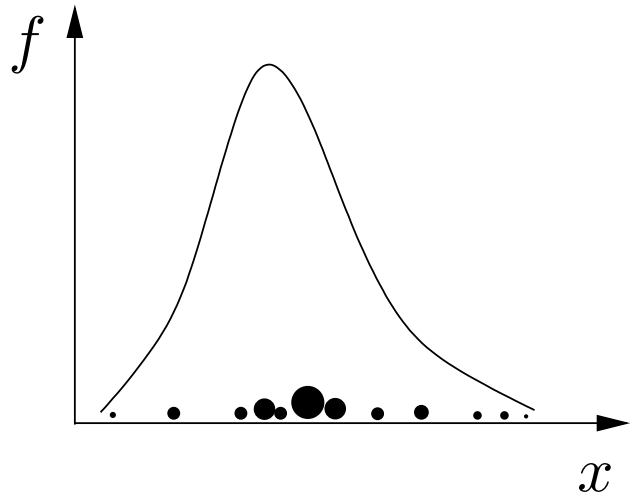
- EM messages (like all messages) may be represented in many different ways, leading to quite different actual computations.
- The freedom (or the necessity) to choose some definite message update schedule leads to different algorithms with different performance (ECM, SAGE, ...).
- The maximization step amounts to applying the max-product algorithm to the corresponding subgraph, which in turn may be carried out by many (exact or approxiate) message passing algorithms. For example, in some important applications, the maximization step can be done by Kalman filtering/smoothing.
- The expectation step relies on plain sum-product messages. However, depending on the involved nodes and message types, the sum-product algorithm may be realized (exactly or approximately) in many different ways.

Particle Methods as Message Passing

Basic idea: represent a probability density f by a list

$$\mathcal{L} = \left((\hat{x}^{(1)}, w^{(1)}), \ldots (\hat{x}^{(L)}, w^{(L)}) \right)$$

of weighted samples ("particles"):



- Versatile data type for sum-product, max-product, EM, ...
- Not sensitive to dimensionality.

Outline

Fourier Transform and Duality

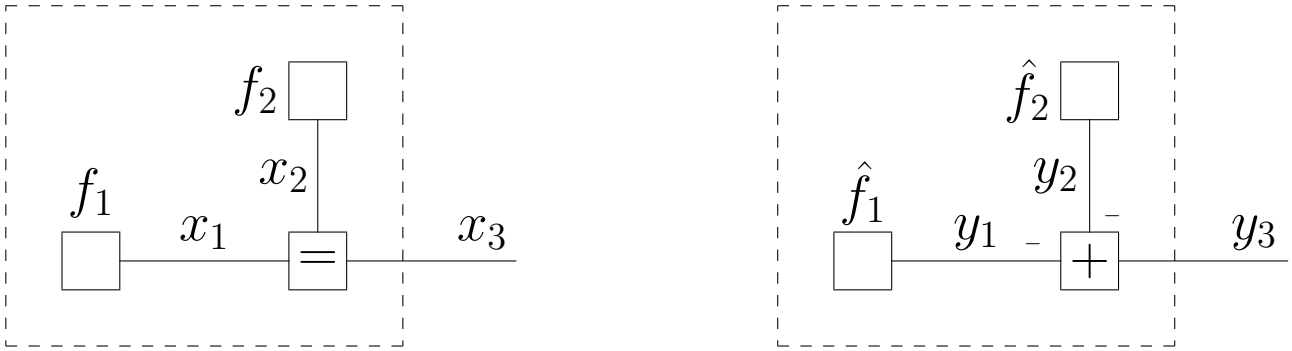
(Forney; Kschischang / Mao)

The Fourier transform of a multi-variable function can be carried out directly in the (Forney-style) factor graph (which may have cycles!):

- Replace each variable by its dual ("frequency") variable.
- Replace each local function by its Fourier transform. If some local function is the membership indicator function $\delta_V(.)$ of a vector space V, its "Fourier transform" is the membership indicator function $\delta_{V^\perp}(.)$ of the orthogonal complement $V^\perp$.
- For each edge, introduce a minus sign into one of the two adjacent factors.

For this recipe to work, all variables of interest must be external, i.e., represented by half edges.

Fourier Transform: Example



The Fourier transform of the pointwise multiplication

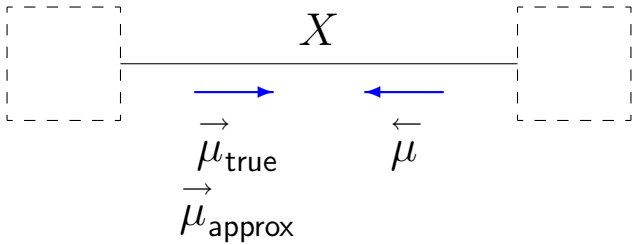
$$\begin{aligned} f(x_3) &= \sum_{x_1,x_2} f_1(x_1) f_2(x_2) \delta(x_1 - x_3) \delta(x_2 - x_3) \\ &= f_1(x_3) f_2(x_3) \end{aligned}$$

is the convolution

$$\begin{aligned} \hat{f}(y_3) &= \sum_{y_1,y_2} \hat{f}_1(y_1) \hat{f}_2(y_2) \delta(y_3 - y_2 - y_1) \\ &= \sum_{y_2} \hat{f}_1(y_3 - y_2) \hat{f}_2(y_2). \end{aligned}$$

Approximate Messages: a General Principle

(Simplification / generalization of an idea due to Minka, "expectation propagation", 2001.)



Assume that we wish to (or need to) replace the "true" summary/message $\vec{\mu}_{\text{true}}$ by some approximation $\vec{\mu}_{\text{approx}}$. This will change the marginal/belief

$$f(x) \stackrel{\triangle}{=} \vec{\mu}_{\text{true}}(x)\,\overleftarrow{\mu}(x)$$

into

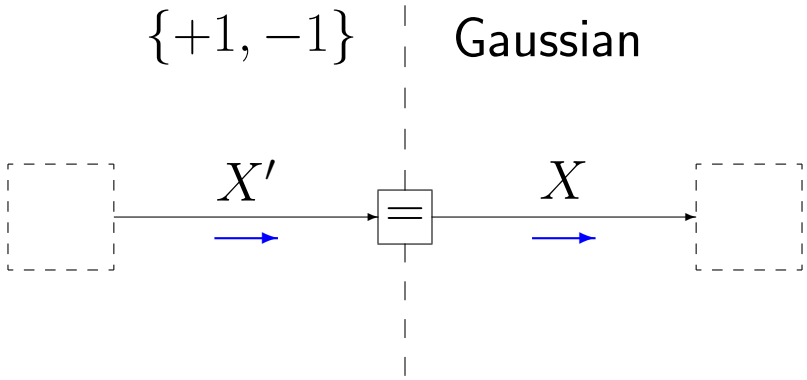
$$\tilde{f}(x) \stackrel{\triangle}{=} \vec{\mu}_{\text{approx}}(x)\,\overleftarrow{\mu}(x). \qquad (1)$$

This suggests to first compute

$$\tilde{f}(x) = \text{some approximation of } \vec{\mu}_{\text{true}}(x)\,\overleftarrow{\mu}(x)$$

(approximation of marginal) and then to compute $\vec{\mu}_{\text{approx}}$ from (1).

Minka Rule: Binary $\rightarrow$ Gaussian (Allerton 2006)



Approximate true marginal

$$f(x) = \sum_{x'} \overrightarrow{\mu}_{X'}(x)\, \delta(x' - x)\, \overleftarrow{\mu}_{X}(x)$$

by Gaussian $\tilde{f}(x)$ (mean and variance matching), then obtain Gaussian

$$\overrightarrow{\mu}_{\text{approx}}(x) \propto \tilde{f}(x) / \overleftarrow{\mu}_{X}(x).$$

May be non-neutral even if $\overrightarrow{\mu}_{X'}(x)$ is neutral.

Fine print: mean and variance of $\overrightarrow{\mu}_{\text{approx}}$; negative "variance"; damping...

Conclusion

- Factor graphs are useful for developing and teaching practical algorithms for detection, estimation, optimization, etc.
- Iterative message passing ("turbo") algorithms often work well (even if convergence cannot be guaranteed) when "optimal" algorithms are infeasible.
- A wealth of nontrivial algorithms can be composed from tabulated message computation rules. The extension and refinement of such tables is an ongoing research activity.
- Factor graphs encourage to mix and match a wide variety of algorithmic techniques.