PROBABILISTIC SIGNAL PROCESSING ON NORMAL FACTOR GRAPH ARCHITECTURES

Francesco A. N. Palmieri

Dipartimento di Ingegneria Industriale e dell'Informazione
Seconda Università di Napoli (SUN) - Italy

Graduate Students:
Amedeo Buonanno
Francesco Castaldo
Anna Di Benedetto

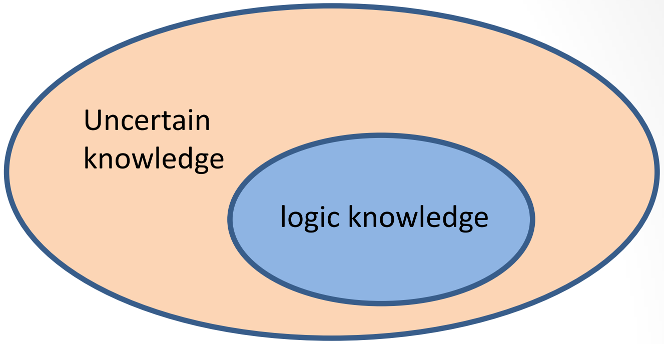
Master's Students:
Luigi Di Grazia
Pasquale Coscia

Stanford - April 2015

Outline:

- Motivation and introduction to Bayesian thinking
- Belief propagation in factor graphs in Normalized Reduced Form
- Localized Learning
- A Simulink architecture
- Applications:
  - Learning non linear functions
  - Tracking by fusing camera information
  - Multi-layer convolution graphs
  - Deep Belief quad-tree factor graphs
- Probabilistic computing machines
- Conclusions and trends

Intelligence = manage uncertainties



Smart fusion consists in providing the best answer with any available information, with both discrete and continuous variables, noise, erasures, errors, hard logic, weak syllogisms, etc.

….The "new'' perception amounts to the recognition that the mathematical rules of probability theory are not merely rules for calculating frequencies of "random variables"; they are also the unique consistent rules for conducting inference ***(i.e. plausible reasoning) of any kind****…*

…..each of his (Kolmogorov's) axioms turns out to be, for all practical purposes, derivable from the Polya-Cox desiderata of rationality and consistency. In short, ***we regard our system of probability as not contradicting Kolmogorov's; but rather seeking a deeper logical foundation*** *that permits its extension in the directions that are needed for modern applications….*

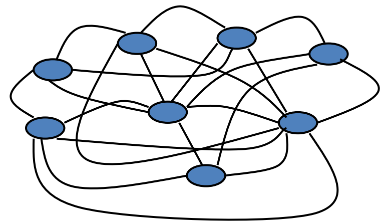
Jaynes E.T., Probability Theory: The Logic of Science, Cambridge University Press (2003)

Why use graphs ?

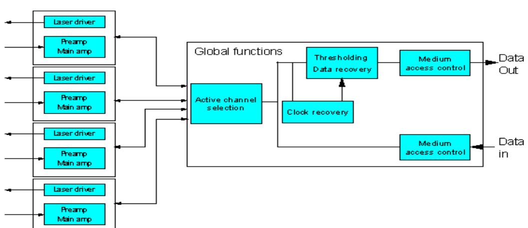
We think on graphs!

The graph represents most of our a priori knowledge about a problem.

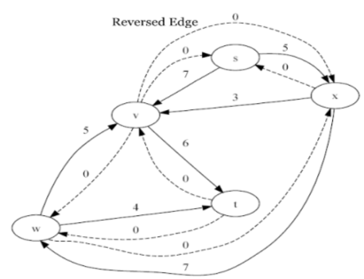
If everything were connected to everything:



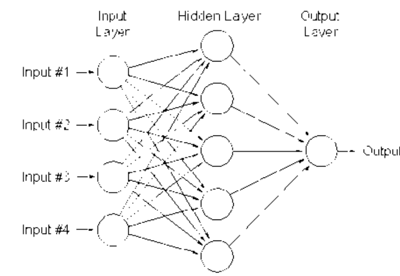
"spaghetti"



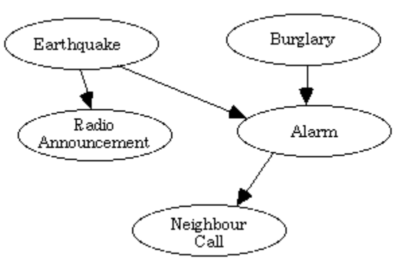
Signal flow diagram



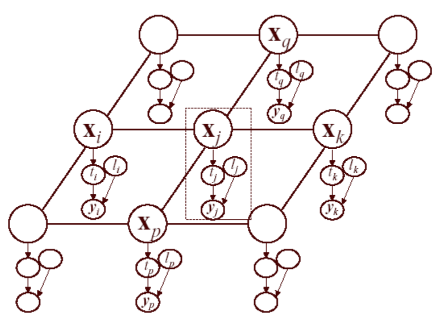
State transition graph



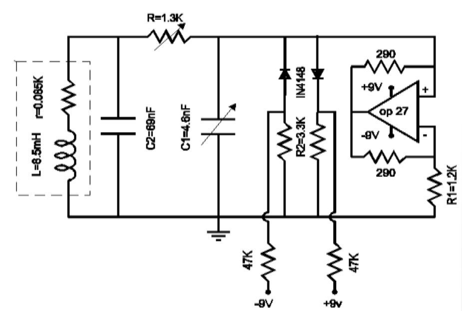
Neural network



Bayesian reasoning



Markov random field



Circuit diagram

The graph models dependencies

Given N (deterministic or random) variables $X_1, X_2, ..., X_N$, model all possible dependencies

$$p(X_1 X_2 ... X_N) \quad \text{(joint pdf)}$$

Knowledge of the structural dependencies is in the factorization (graph) of p

Group the variables in M subsets $\{\mathcal{S}_c, c =, 1, ..., M\}$ (cliques)

$$p(X_1 X_2 ... X_N) \propto \prod_{c=1}^{M} \Psi_c \left(\bigcap_{j \in \mathcal{S}_c} X_j \right) \qquad (\Psi_c \quad \text{potential} \quad \text{functions})$$

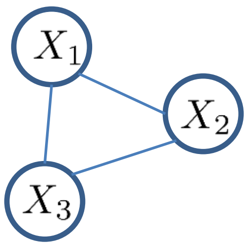
$$p(X_1 X_2 ... X_N) \propto exp \left(\sum_{c=1}^{M} \phi_c \left(\bigcap_{j \in \mathcal{S}_c} X_j \right) \right) \qquad (\phi_c \quad \text{energy} \quad \text{functions})$$

To see how many choices we have, use the chain rule

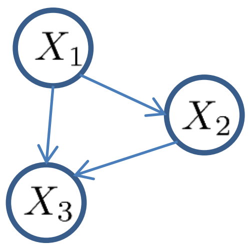
$$p(X_1 X_2 ... X_N) = \prod_{j=1}^{N} p(X_j | X_{j+1} ... X_N)$$

drop some conditioning variables using conditional independence assumptions. There are $N!$ ways of rearraging the variables.

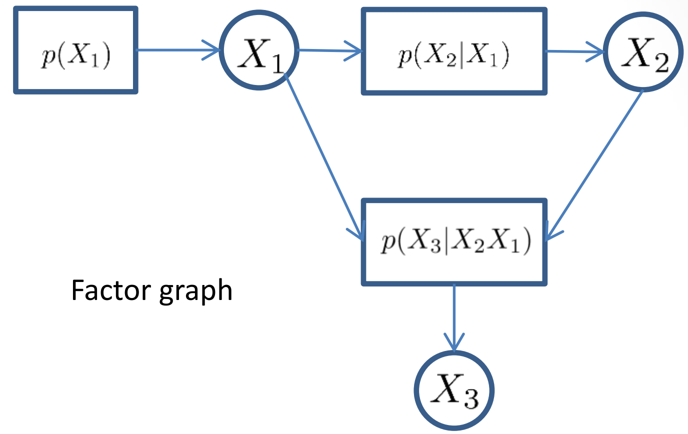
What kind of Bayesian graph ?



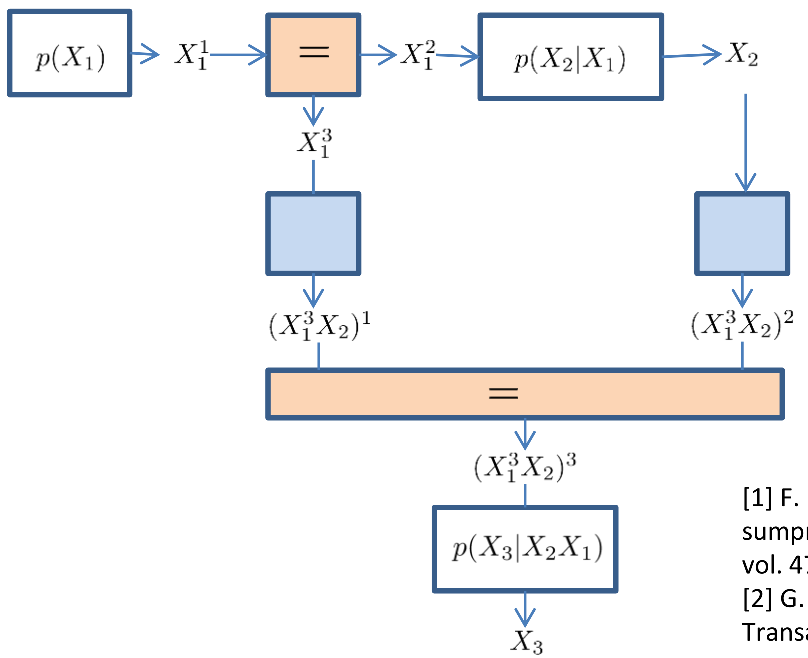
Undirected graph



Directed graph



Factor graph



Normal Graph (Forney's style [1][2]) in reduced form (see [3] for details): **factors, except for the replicators, have at most two terminations**

- Much easier message propagation
- Unique rules for learning

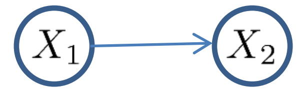
(this example has a loop)

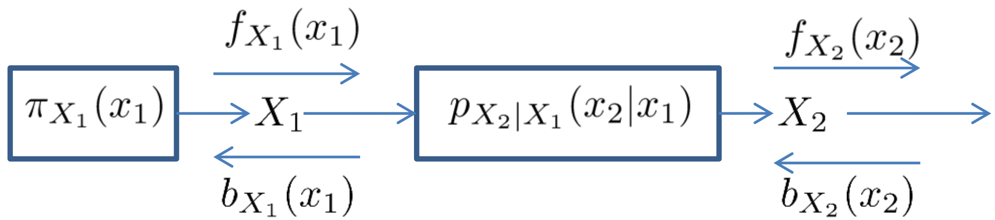
[1] F. R. Kschischang, B. Frey, and H. Loeliger, "Factor graphs and the sumproduct algorithm," IEEE Transactions on Information Theory, vol. 47, pp. 498–519, 2001.
[2] G. D. Forney, "Codes on graphs: normal realizations," IEEE Transactions on Information Theory, vol. 47, pp. 520–548, 2001.

[3] F. A. N. Palmieri, "A Comparison of Algorithms for Learning Hidden Variables in Normal Graphs", *submitted for journal publication, 2014, available on* arXiv: 1308.5576v1 [stat.ML]

Example 1: (to see how message propagation works)

$$p_{X_1X_2}(x_1x_2) = p_{X_2|X_1}(x_2|x_1)\pi_{X_1}(x_1)$$





Possible use: observe $X_2 = x_2^0$ and infer on X_1

$$p(x_1|X_2 = x_2^0) = \frac{p(X_2 = x_2^0|x_1)\pi_{X_1}(x_1)}{p(X_2 = x_2^0)} \propto \underbrace{p(X_2 = x_2^0|x_1)}_{b_{X_1}(x_1)}\underbrace{\pi_{X_1}(x_1)}_{f_{X_1}(x_1)}$$

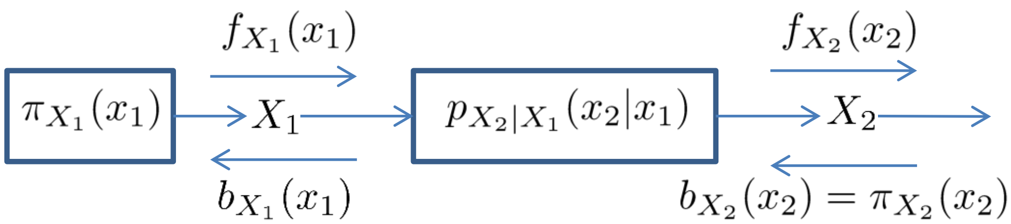
$$b_{X_1}(x_1) = \int_{\mathcal{X}_2} p_{X_2|X_1}(x_2|x_1)\underbrace{\delta(x_2 - x_2^0)}_{b_{X_2}(x_2)}dx_2$$

$$f_{X_2}(x_2) = \int_{\mathcal{X}_1} p_{X_2|X_1}(x_2|x_1)\underbrace{\pi_{X_1}(x_1)}_{f_{X_1}(x_1)}dx_1$$

Example 1: (cont.)

$$p_{X_1X_2}(x_1x_2) = p_{X_2|X_1}(x_2|x_1)\pi_{X_1}(x_1)$$

Possible use: use soft knowledge on X_2, $\pi_{X_2}(x_2)$ and infer on X_1

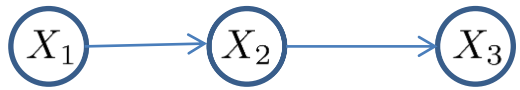


$$p(x_1|\pi_{X_2}) \stackrel{def}{=} \int_{\mathcal{X}_2} p(x_1|X_2 = x_2)\pi_{X_2}(x_2)dx_2 \propto \underbrace{\pi_{X_1}(x_1)}_{f_{X_1}(x_1)} \underbrace{\int_{\mathcal{X}_2} p_{X_2|X_1}(x_2|x_1)\underbrace{\pi_{X_2}(x_2)}_{b_{X_2}(x_2)}dx_2}_{b_{X_1}(x_1)}$$

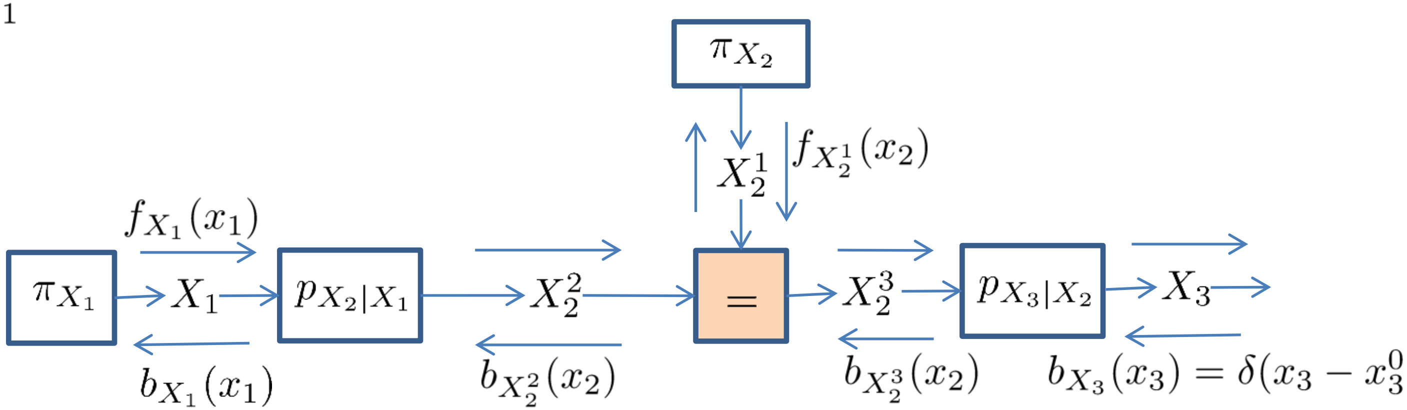
average posterior

Sum-Product rule

Example 2: $p_{X_1X_2X_3}(x_1x_2x_3) = p_{X_3|X_2}(x_3|x_2)p_{X_2|X_1}(x_2|x_1)\pi_{X_1}(x_1)$



Possible use: observe $X_3 = x_3^0$, use soft knowledge on X_2, $\pi_{X_2}(x_2)$ and infer on X_1



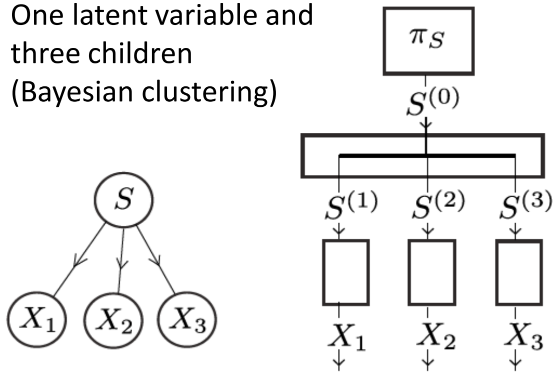
Insert a T-junction in the probability pipeline

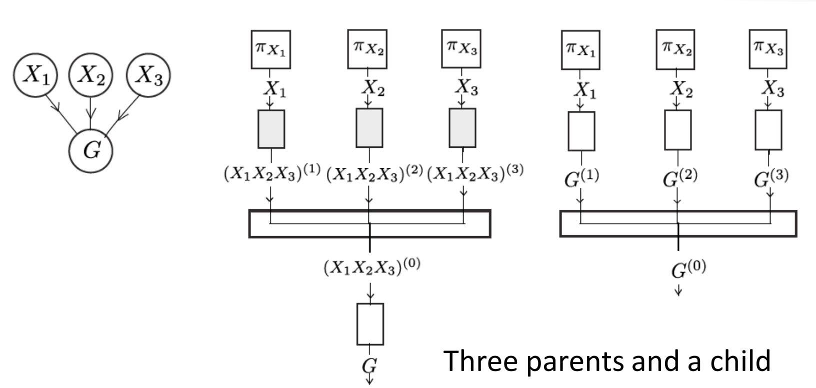
$f_{X_1}(x_1) = \pi_{X_1}(x_1);\ f_{X_2^1}(x_2) = \pi_{X_2}(x_2);$
$b_{X_2^3}(x_2) \propto \int_{\mathcal{X}_3} p_{X_3|X_2}(x_3|x_2)b_{X_3}(x_3)dx_3;$ (sum)
$b_{X_2^2}(x_2) \propto f_{X_2^1}(x_2)b_{X_2^3}(x_2);$ (product)
$b_{X_1}(x_1) \propto \int_{\mathcal{X}_2} p_{X_2|X_1}(x_2|x_1)b_{X_2^2}(x_2)dx_2;$ (sum)
$p(x_1|\pi_{X_2}, X_3 = x_3^0) \propto f_{X_1}(x_1)b_{X_1}(x_1)$ (product)

Message propagation rules are rigorous translation of Bayes theorem and marginalization

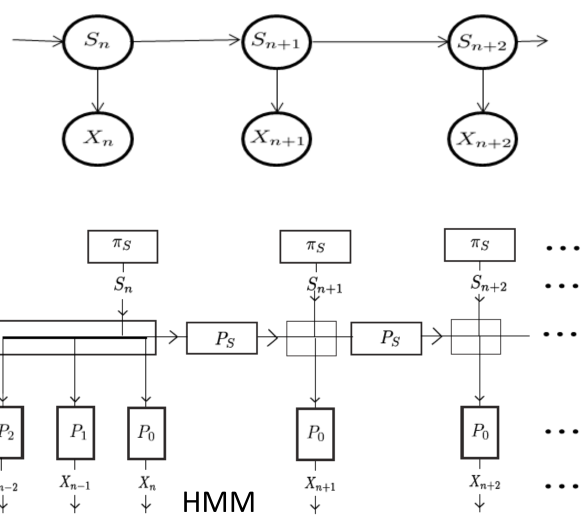
Some architectures reduced normal form :

One latent variable and
three children
(Bayesian clustering)



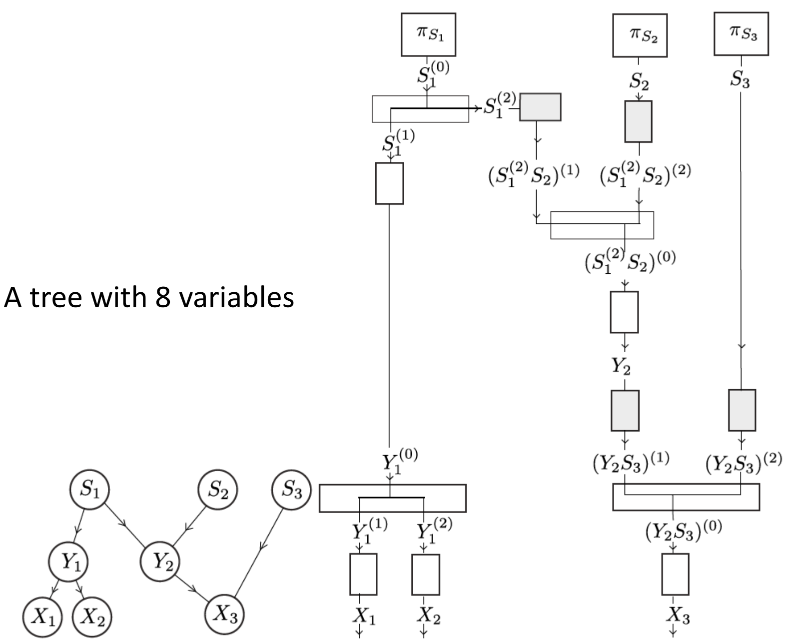


Three parents and a child

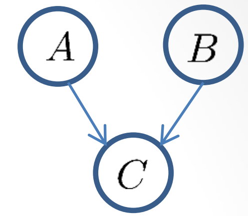


HMM

A tree with 8 variables



You will never look at a function in the same way !!



$C = A + B$ (arithmetic sum - deterministic function);

$A \in \{0,1\}$; $B \in \{0,1\}$; $C \in \{0,1,2\}$

$$P_1 = \tfrac{2}{4} I_2 \otimes 1_2^T = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix};$$

$$P_2 = \tfrac{2}{4} 1_2^T \otimes I_2 = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

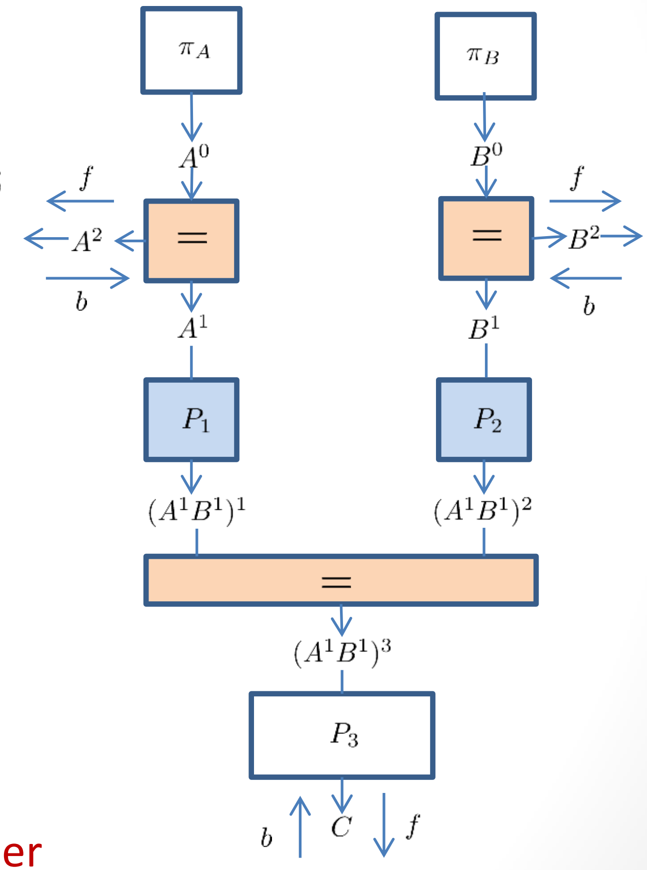
$$P_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

$$\text{INPUTS: } b_{A^2} = \begin{bmatrix} .1 \\ .9 \end{bmatrix};\ b_{B^2} = \begin{bmatrix} .99 \\ .01 \end{bmatrix};\ b_C = \begin{bmatrix} .33 \\ .33 \\ .33 \end{bmatrix};$$

$$\text{OUTPUTS: } f_{A^2} = \begin{bmatrix} .5 \\ .5 \end{bmatrix};\ f_{B^2} = \begin{bmatrix} .5 \\ .5 \end{bmatrix};\ f_C = \begin{bmatrix} .01 \\ .98 \\ .01 \end{bmatrix};$$

$$\text{INPUTS: } b_{A^2} = \begin{bmatrix} .5 \\ .5 \end{bmatrix};\ b_{B^2} = \begin{bmatrix} .99 \\ .01 \end{bmatrix};\ b_C = \begin{bmatrix} .2 \\ .8 \\ .0 \end{bmatrix};$$

$$\text{OUTPUTS: } f_{A^2} = \begin{bmatrix} .21 \\ .79 \end{bmatrix};\ f_{B^2} = \begin{bmatrix} .56 \\ .45 \end{bmatrix};\ f_C = \begin{bmatrix} .5 \\ .5 \\ .0 \end{bmatrix};$$



Easily extended to arbitrary-length N-bit adder

Issues:

1. Posterior calculation on trees is exact

(Pearl, 1988), (Lauritzen, 1996), (Jordan, 1998), (Loeliger, 2004), (Forney, 2001), (Bishop, 2006), (Barber, 2012),

……expressive power of trees if often limited, but a lot can be done with trees

2. "Loopy graphs" (Chertkov, Chernyak and Teodorescu, 2008), (Murphy, Weiss, and Jordan, 1999), (Yedidia, Freeman and Weiss, 2000, 2005), (Weiss, 2000), (Weiss and Freeman, 2001)

…….simple belief propagation can lead to inconsistencies

Junction Trees (Lauritzen, 1996); **Cutset Conditioning** (Bidyuk and R. Dechter, 2007); **Monte Carlo sampling** (see for ex. Koller and Friedman, 2010); **Region method** (Yedidia, Freeman and Weiss, 2005).; **Tree Re-Weighted (TRW)** algorithm (Wainwright, Jaakkola and Willsky, 2005);

…….sometimes using simple loopy propagation gives good results if the loops are wide

3. Parameter learning

EM-learning: (Heckerman, 1996), (Koller and Friedman, 2010), (Ghahramani, 2012); **Variational Learning: (**Winn and Bishop, 2005)

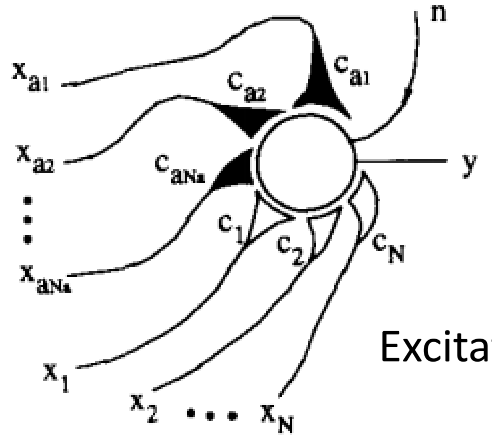
4. Structure Learning

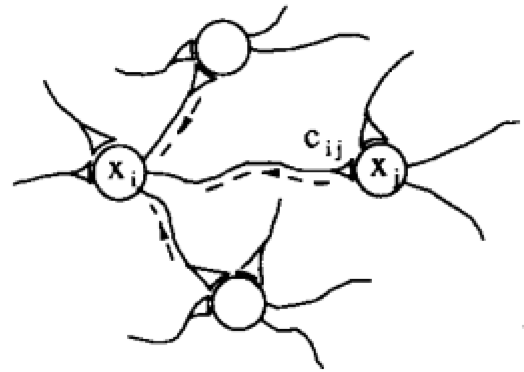
Learning trees: (Chow and Liu, 1968) ,(Zhang, 2004), (Harmeling and Williams, 2011), (Palmieri, 2010), (Choi, Anandkumar and Willsky, 2011); **Learning general architectures** (Koller and Friedman, 2010)

5. Applications

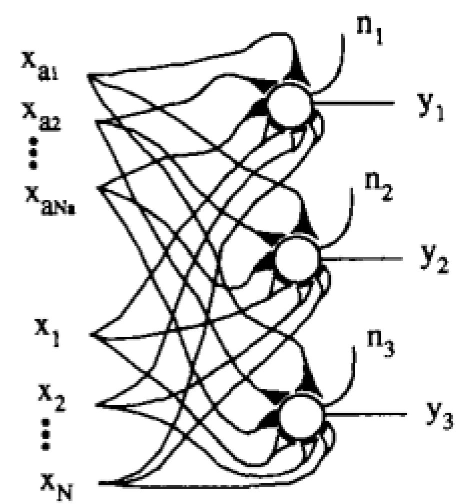
Coding; HMM; Complex scene analysis; Fusion of heterogeneous sources; ….opportunity of integrating more traditional signal processing with higher levels of cognition!

Localized learning:





Excitatory-inhibitory synapses

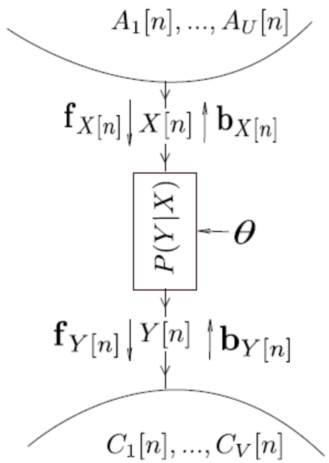


Hebbian hypothesis

There is a certain amount of bi-directionality in Hebbian algorithms!
Is there a single unique algorithm???? (Fei Fei talk)

SOME OLD WORK:
F. Palmieri, C. Catello and G. D'Orio, ``Inhibitory Synapses in Neural Networks with Sigmoidal Nonlinearities,"
IEEE Trans. on Neural Networks, Vol. 10, N. 3, pp. 635-644, May 1999.
F. Palmieri, J. Zhu, ``Self-Association and Hebbian Learning in Linear Neural Networks,"
IEEE Trans. on Neural Networks, Vol. 6, N. 5, pp. 1165-1183, Sept. 1995.
F. Palmieri, J. Zhu and C. Chang, ``Anti-Hebbian Learning in Topologically Constrained Linear Neural Networks: a Tutorial ,"
IEEE Trans. on Neural Networks, Vol. 4, N. 5, 748-761, Sept. 1993.

In the graph in Reduced normal form learning is totally localized to SISO blocks



- Each block “sees” only local messages
- P(Y/X) is a discrete-variable stochastic matrix
- EM approach on N training examples

$$P(XYA_1...A_UC_1...C_V;\theta) = P(C_1...C_V|Y)\, \underbrace{P(Y|X;\theta)}_{\text{to be learned}}\, P(XA_1...A_U).$$

$$L(\theta) = \prod_{n=1}^{N}\sum_x\sum_y p_{X[n]Y[n]\mathcal{E}[n]}(xy;\theta) = \prod_{n=1}^{N}\sum_x\sum_y f'_{X[n]}(x)p_{Y|X}(y|x;\theta)b'_{Y[n]}(y),$$

$$\ell(\theta) = \log(L(\theta)) = \sum_{n=1}^{N}\log\left(\mathbf{f}^T_{X[n]}\ \theta\ \mathbf{b}_{Y[n]}\right) + \sum_{n=1}^{N}\log\left(K_{f_{X[n]}}K_{b_{Y[n]}}\right)$$

$$\begin{cases} \min_\theta & -\sum_{n=1}^{N}\log\left(\mathbf{f}^T_{X[n]}\ \theta\ \mathbf{b}_{Y[n]}\right), \\ \theta & \text{row}-\text{stochastic}, \end{cases}$$

ML learning

$$\begin{cases} \min_\theta & \sum_{n=1}^{N}\sum_{i=1}^{M_y} b_{Y[n]}(j)\log\frac{b_{Y[n]}(j)}{f_{Y[n]}(j)}, \\ \theta & \text{row}-\text{stochastic} \end{cases}$$

Minimum KL-divergence learning

EM Algorithms:

ML Algorithm:

(1) $\theta_{lm} \longleftarrow \frac{\theta_{lm}}{\sum_{n=1}^{N} f_{X[n]}(l)} \sum_{n=1}^{N} \frac{f_{X[n]}(l) b_{Y[n]}(m)}{\mathbf{f}_{X[n]}^T \theta \mathbf{b}_{Y[n]}},$

(2) Row-normalize θ and back to (1)

KL Algorithm:

(1) $\theta_{lm} \longleftarrow \frac{\theta_{lm}}{\sum_{n=1}^{N} f_{X[n]}(l)} \sum_{n=1}^{N} \frac{f_{X[n]}(l) b_{Y[n]}(m)}{\sum_{i=1}^{M_X} \theta_{im} f_{X[n]}(i)},$

(2) Row-normalize θ and back to (1).

VITERBI-like Algorithm:

(1) $\mathbf{e}_{X[n]} = I_{Max}(\mathbf{f}_{\mathbf{X[n]}}) + \delta \mathbf{1}_{M_X \times 1};$

$\quad \mathbf{e}_{Y[n]} = I_{Max}(\mathbf{b}_{\mathbf{Y[n]}}) + \delta \mathbf{1}_{M_Y \times 1};$

(2) $\theta = \sum_{n=1}^{N} \mathbf{e}_{X[n]} \mathbf{e}_{Y[n]}^T;$

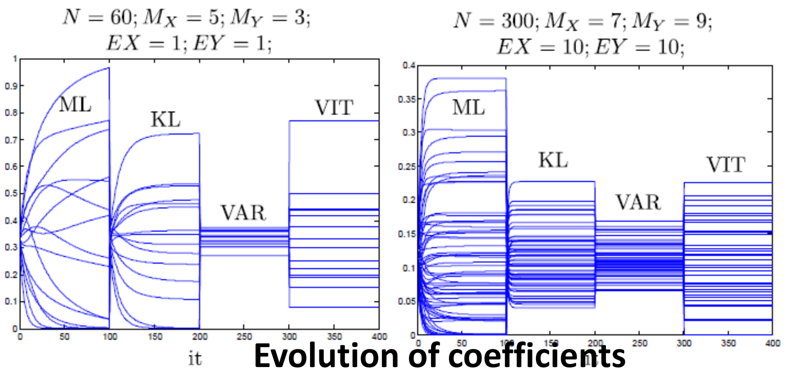
(3) Row-normalize θ.

VARIATIONAL Algorithm:

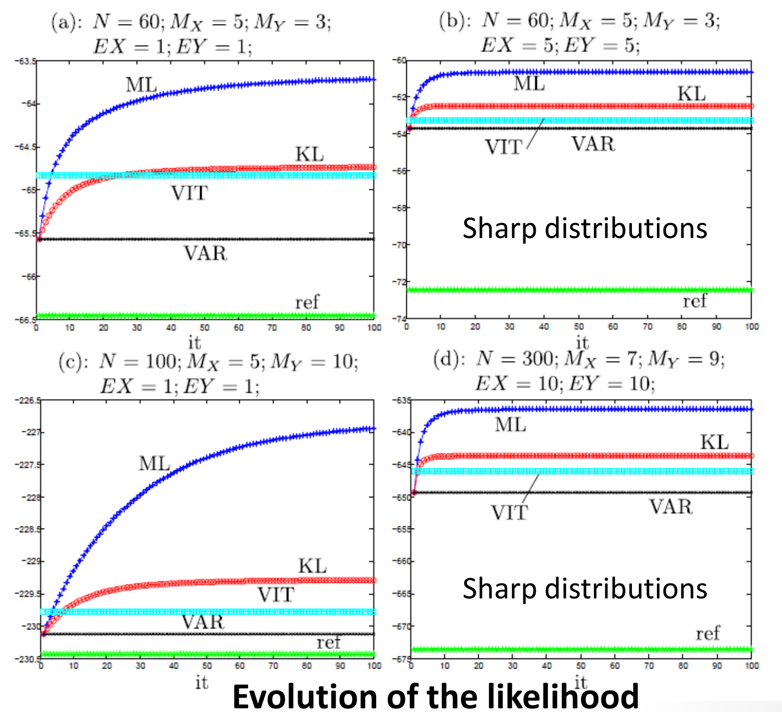
(1) $\theta_{lm} \longleftarrow \delta + \sum_{n=1}^{N} f_{X[n]}(l) b_{Y[n]}(m),$

(2) Row-normalize θ.

1. Simulations on a single block;
2. Varying sharpness
3. Similar behaviour for more complicated architectures
4. Greedy search: Local minima (multiple restarts)



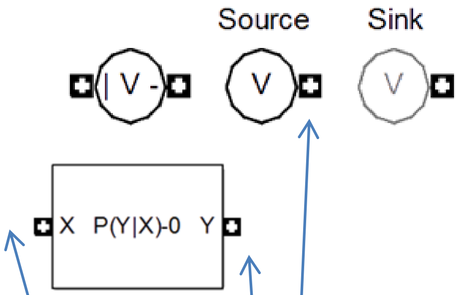
Evolution of coefficients

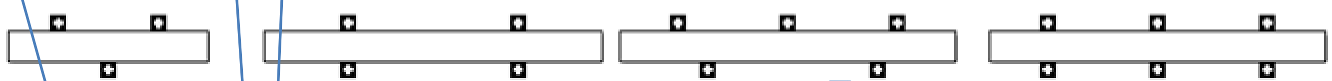


Evolution of the likelihood

F. A. N. Palmieri, "A Comparison of Algorithms for Learning Hidden Variables in Normal Graphs", *submitted for journal publication, Jan 2014,* arXiv: 1308.5576v1 [stat.ML]

Simulink Library





It uses two-way connection ports

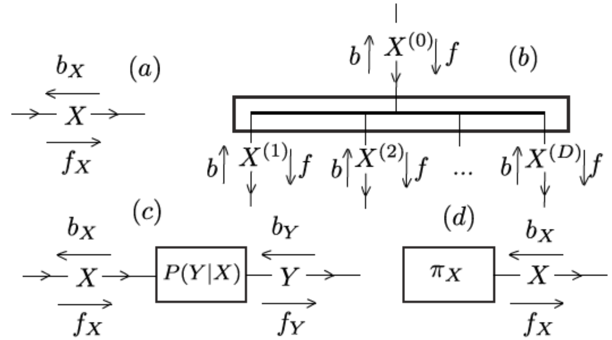
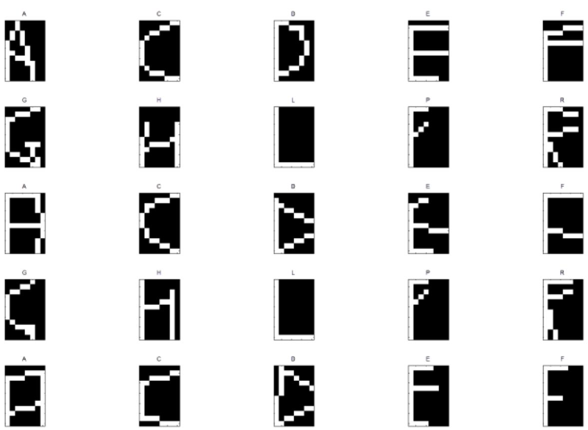
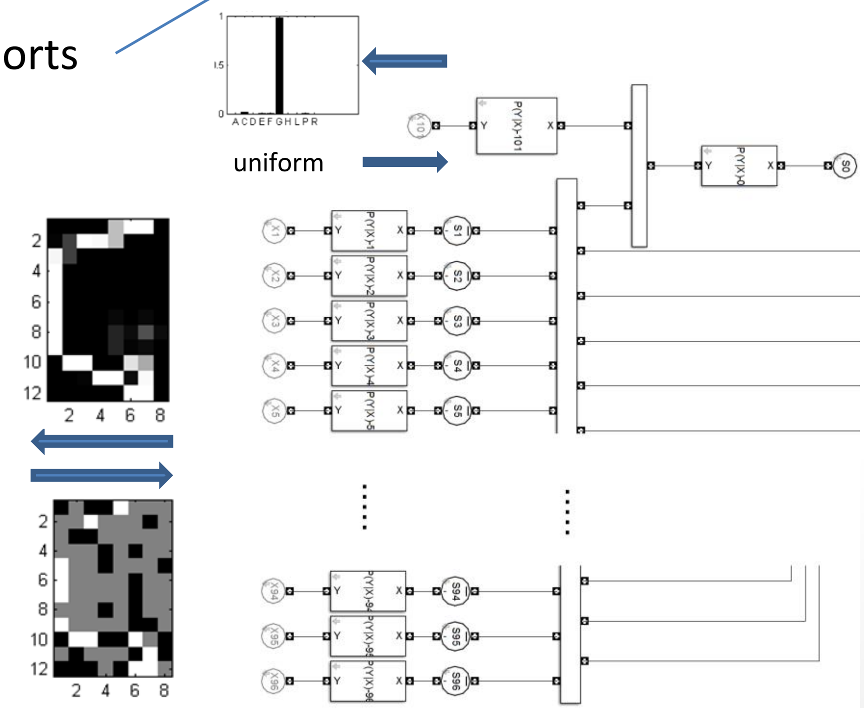
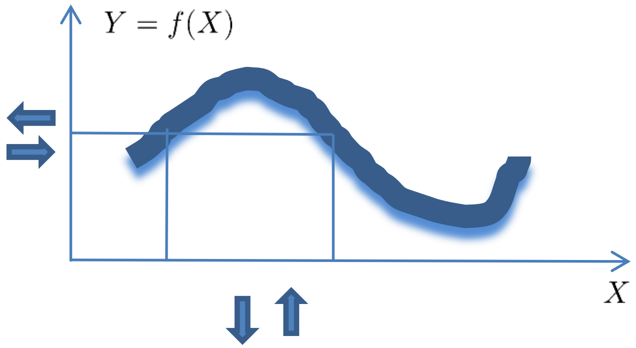


Figure 1: FGrn components: (a) a variable branch; (b) a diverter; (c) a SISO block; (d) a source block.



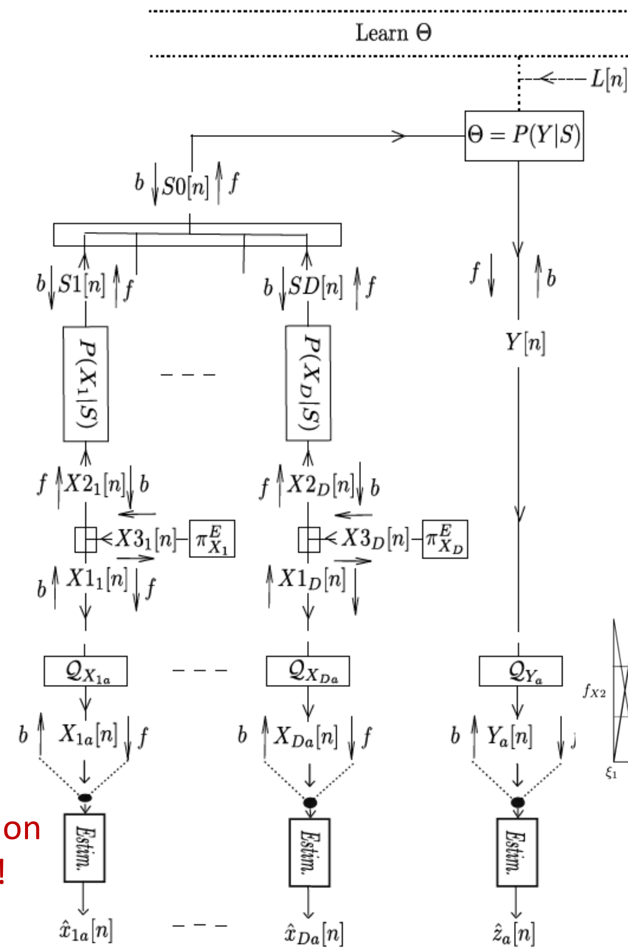


Application 1: Learning a Nonlinear Function

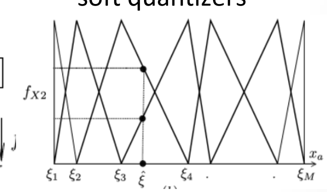


Think of the function as a joint density $p(X, Y)$

1. Map input variables to an embedding space
2. Minimize $KL(bY||fY)$
3. ~ tensor-product approximation



soft quantizers



The factor graph for learning the D-dimensional function $Y_a = g(X_{1a}, ..., X_{Da})$.

The objective is not to challenge in accuracy SVMs, MLPs, RBFs etc., but to see the function approximation problem as part of a unique architectural paradigm !

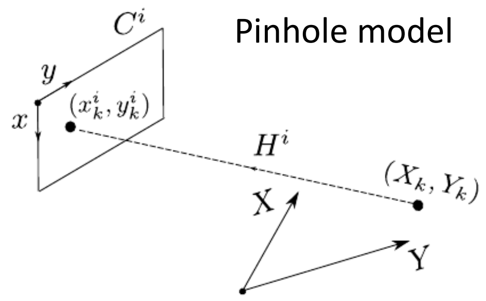
Francesco A. N. Palmieri, "Learning Non-Linear Functions with Factor Graphs," *IEEE Transactions on Signal Processing, Vol.61, N. 17, pp.* 4360 - 4371, 2013.

Application 2: Tracking objects with cameras

Salerno (Italy) harbour (3 commercial cameras)

Typical views

Application 2: Tracking objects with cameras (cont.)



Pinhole model

World coordinates ← Image coordinates

$$\begin{pmatrix} X_k/\lambda_k^i \\ Y_k/\lambda_k^i \\ 1/\lambda_k^i \end{pmatrix} = R^i \begin{pmatrix} x_k^i \\ y_k^i \\ 1 \end{pmatrix},$$

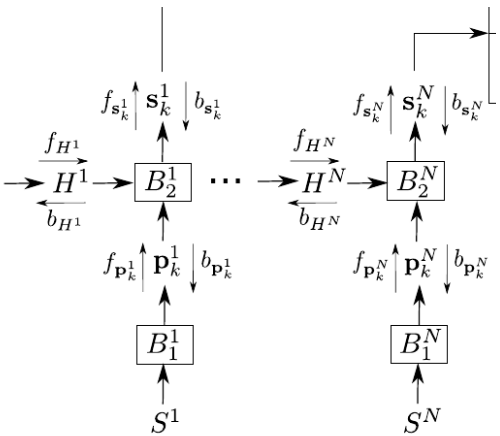
$$\begin{aligned}
X_k &= \frac{X_k/\lambda_k^i}{1/\lambda_k^i} = \frac{r_{11}^i x_k^i + r_{12}^i y_k^i + r_{13}^i}{r_{31}^i x_k^i + r_{32}^i y_k^i + r_{33}^i} = g_1^i(x_k^i, y_k^i),\\
Y_k &= \frac{Y_k/\lambda_k^i}{1/\lambda_k^i} = \frac{r_{21}^i x_k^i + r_{22}^i y_k^i + r_{23}^i}{r_{31}^i x_k^i + r_{32}^i y_k^i + r_{33}^i} = g_2^i(x_k^i, y_k^i),\\
\dot{X}_k &= \frac{dX_k^i}{dt} = g_3^i(x_k^i, y_k^i, \dot{x}_k^i, \dot{y}_k^i),\\
\dot{Y}_k &= \frac{dY_k^i}{dt} = g_4^i(x_k^i, y_k^i, \dot{x}_k^i, \dot{y}_k^i),
\end{aligned}$$

Image coordinates ← World coordinates

$$\lambda_k^i \begin{pmatrix} x_k^i \\ y_k^i \\ 1 \end{pmatrix} = H^i \begin{pmatrix} X_k \\ Y_k \\ 1 \end{pmatrix},$$

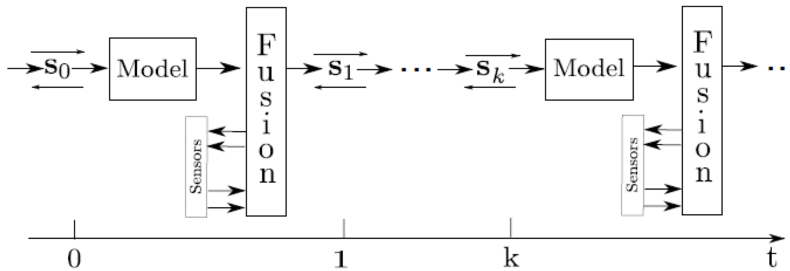
Homography matrix
(learned from calibration points)

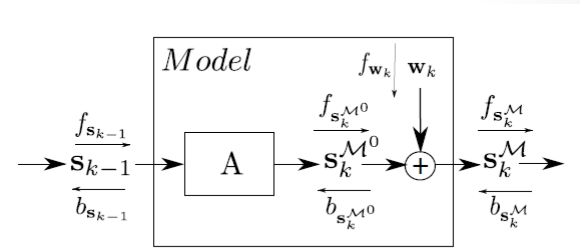
$$\begin{aligned}
x_k^i &= \frac{\lambda_k^i x_k^i}{\lambda_k^i} = \frac{h_{11}^i X_k + h_{12}^i Y_k + h_{13}^i}{h_{31}^i X_k + h_{32}^i Y_k + h_{33}^i} = q_1^i(X_k, Y_k),\\
y_k^i &= \frac{\lambda_k^i y_k^i}{\lambda_k^i} = \frac{h_{21}^i X_k + h_{22}^i Y_k + h_{23}^i}{h_{31}^i X_k + h_{32}^i Y_k + h_{33}^i} = q_2^i(X_k, Y_k),\\
\dot{x}_k^i &= \frac{dx_k^i}{dt} = q_3^i(X_k, Y_k, \dot{X}_k, \dot{Y}_k),\\
\dot{y}_k^i &= \frac{dy_k^i}{dt} = q_4^i(X_k, Y_k, \dot{X}_k, \dot{Y}_k).
\end{aligned}$$

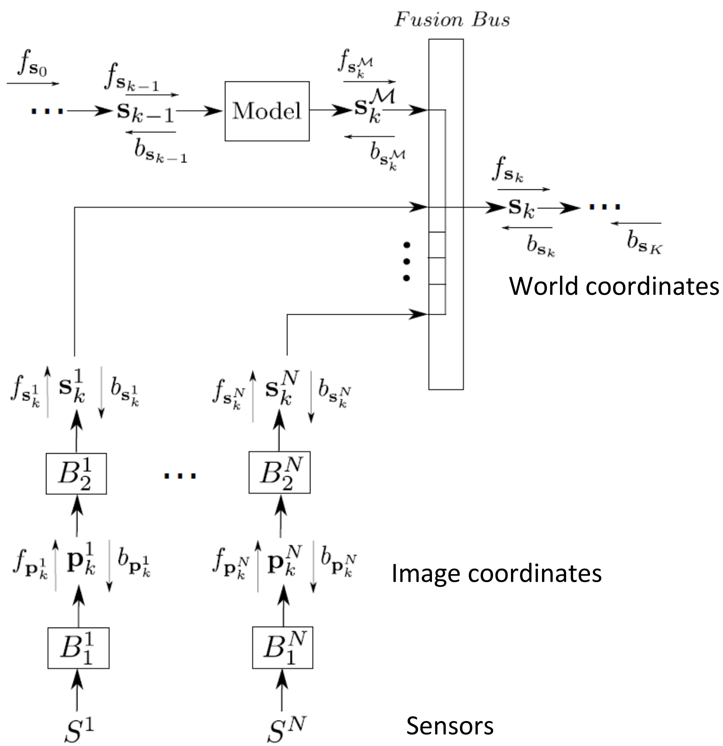


- Local first-order approximations for Gaussian pdf propagation;
- Gaussian noise on the homography matrix

Application 2: Tracking objects with cameras (cont)







Gaussian messages (means and covariances):

$$
\begin{aligned}
f_{\mathbf{s}_k^{\mathcal{M}0}} &= \{A m_{f_{\mathbf{s}_{k-1}}}, A\Sigma_{f_{\mathbf{s}_{k-1}}} A^T\},\\
f_{\mathbf{s}_k^{\mathcal{M}}} &= \{m_{f_{\mathbf{s}_k^{\mathcal{M}0}}} + m_{f_{\mathbf{w}_k}}, \Sigma_{f_{\mathbf{s}_k^{\mathcal{M}0}}} + \Sigma_{f_{\mathbf{w}_k}}\},\\
b_{\mathbf{s}_k^{\mathcal{M}0}} &= \{m_{b_{\mathbf{s}_k^{\mathcal{M}}}} - m_{f_{\mathbf{w}_k}}, \Sigma_{b_{\mathbf{s}_k^{\mathcal{M}}}} + \Sigma_{f_{\mathbf{w}_k}}\},\\
b_{\mathbf{s}_{k-1}} &= \{(A^T\Sigma^{-1}_{b_{\mathbf{s}_k^{\mathcal{M}0}}} A)^{-1} A^T \Sigma^{-1}_{b_{\mathbf{s}_k^{\mathcal{M}0}}} m_{b_{\mathbf{s}_k^{\mathcal{M}0}}}, (A^T\Sigma^{-1}_{b_{\mathbf{s}_k^{\mathcal{M}0}}} A)^{-1}\}.
\end{aligned}
$$

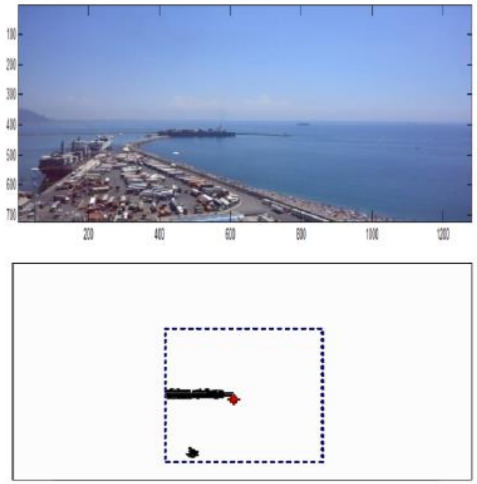
$$
\begin{aligned}
f_{\mathbf{s}_k} &= f_{\mathbf{s}_k^{\mathcal{M}}} \odot (\odot_{j=1}^{N} f_{\mathbf{s}_k^j}),\\
b_{\mathbf{s}_k^{\mathcal{M}}} &= b_{\mathbf{s}_k} \odot (\odot_{j=1}^{N} f_{\mathbf{s}_k^j}),\\
b_{\mathbf{s}_k^i} &= b_{\mathbf{s}_k} \odot f_{\mathbf{s}_k^{\mathcal{M}}} \odot (\odot_{j=1, j\neq i}^{N} f_{\mathbf{s}_k^j}),\ i = 1, ..., N.
\end{aligned}
$$

(Kalman filter equations ''pipelined'')

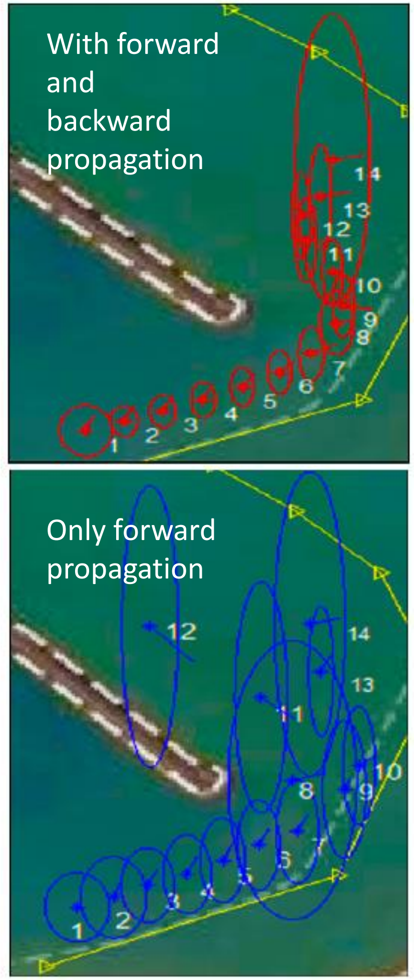
H.-A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. Kschischang, The factor graph approach to model-based signal processing, *Proceedings of the IEEE*, vol. 95, no. 6, pp. 12951322, 2007.

Stanford - April 2015

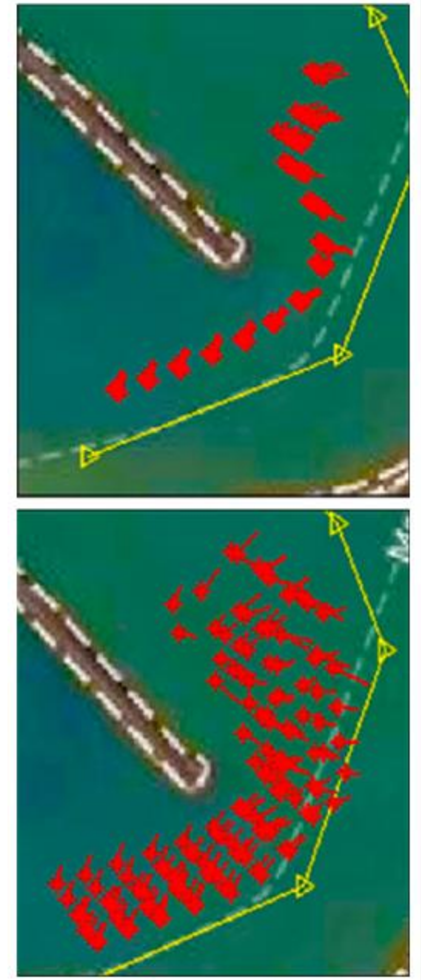
Application 2: Tracking objects with cameras (cont.)



Background subtraction algorithm



No calibration error
(covariances amplified 10^6)

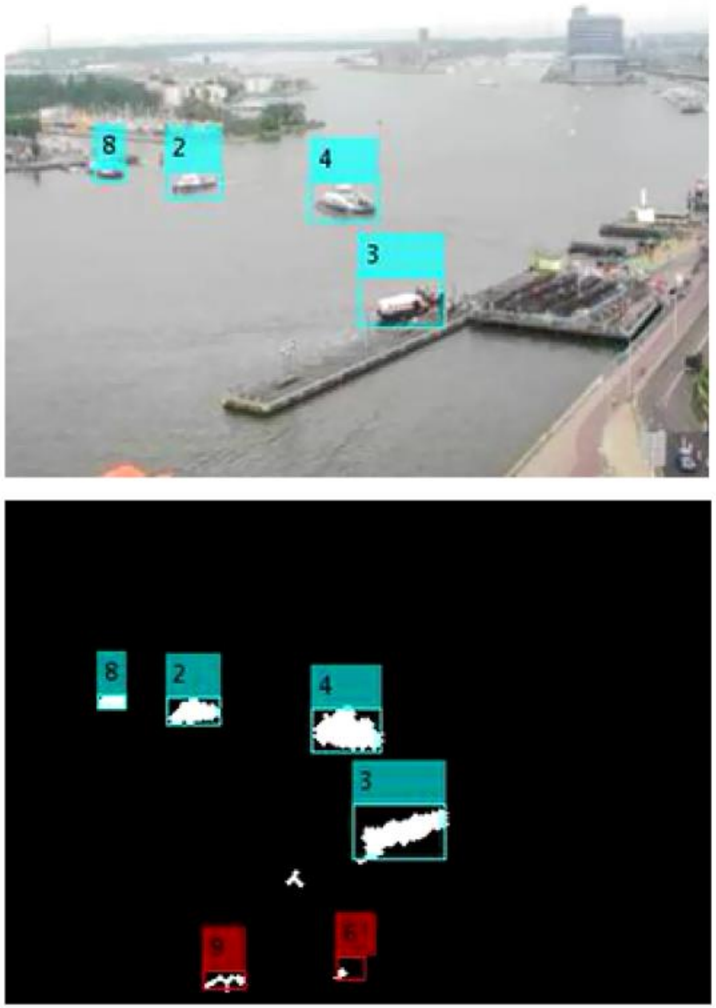


With calibration error
(10^-3; 10^-4)

F. Castaldo and F. A. N. Palmieri, "Target Tracking using Factor Graphs and Multi-Camera Systems", *IEEE Transactions on Aerospace and Electronic Systems (TAES),* 2015, in press.

Application 2: Tracking objects with cameras (cont.)

- Motion detection;
- Sea clutter (waves and wakes);
- Data association ;
- Birth and death of tracks;
- Multiple objects tracked using propagation of Gaussians mixtures.



Application 3: Multi-layer convolution graphs

- Striking achievements in “deep belief networks” rely on convolutional and recurrent structures in multi-layer neural networks (Hinton, Le Cun, Bengio, Ng)
- Convolutive paradigms in Bayesian factor graphs?
- Convolutive structures better than trees account for short distance chained dependences;
- Expansion to hierarchies to capture long-term dependence at a gradually increasing scale.

triplets

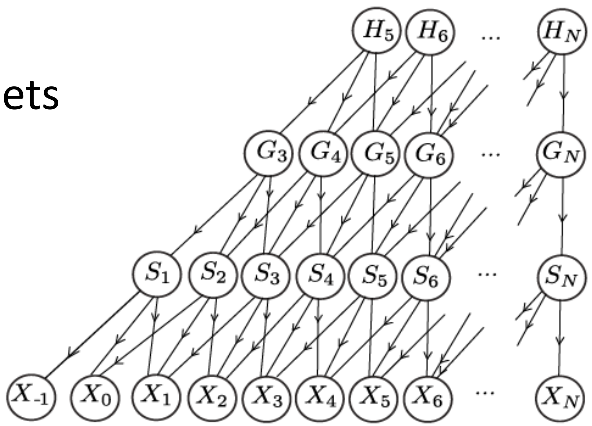


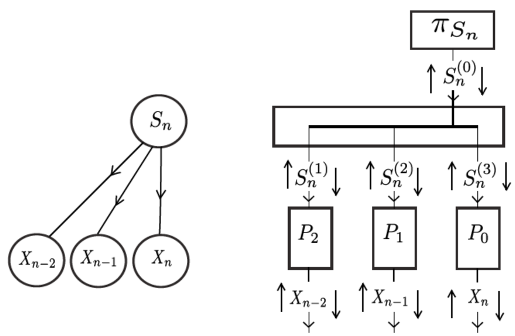
Fig. 1. Example of three-layer convolution graph with overlap $M = 2$ on all layers.

Many many loops!!

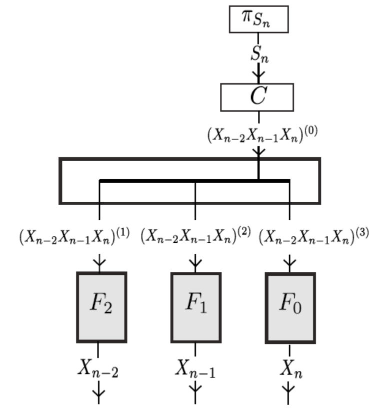
It appears intractable for message propagation;

Stationarity allows a transformation

Application 3: Multi-layer convolution graph (cont.)

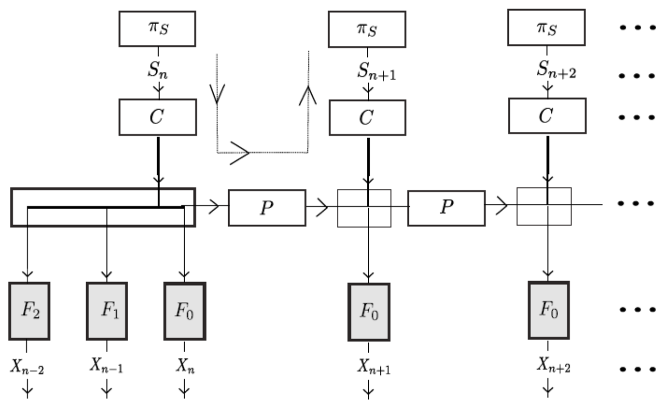


Latent model



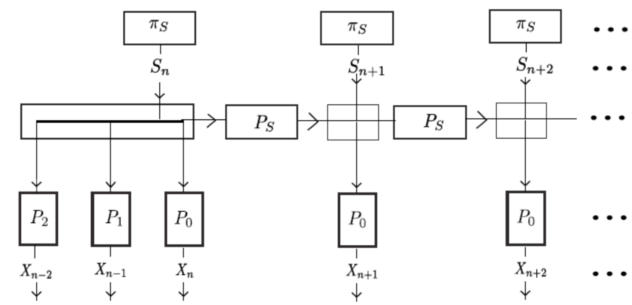
$CF_1 \simeq P_1$;
$CF_2 \simeq P_2$;
$CF_3 \simeq P_3$;
C row-stochastic

Explicit mapping to product space



Junction tree

$$P = P(X_{n-1}X_nX_{n+1}|X_{n-2}X_{n-1}X_n) = \mathbf{1}_d \otimes I_d \otimes I_d \otimes \tfrac{1}{d}\mathbf{1}_d^T.$$



HMM approximation

$P_S = CPC^T$;
Learned from data
if the product space is too large

Application 3: Multi-layer convolution graph (cont.)

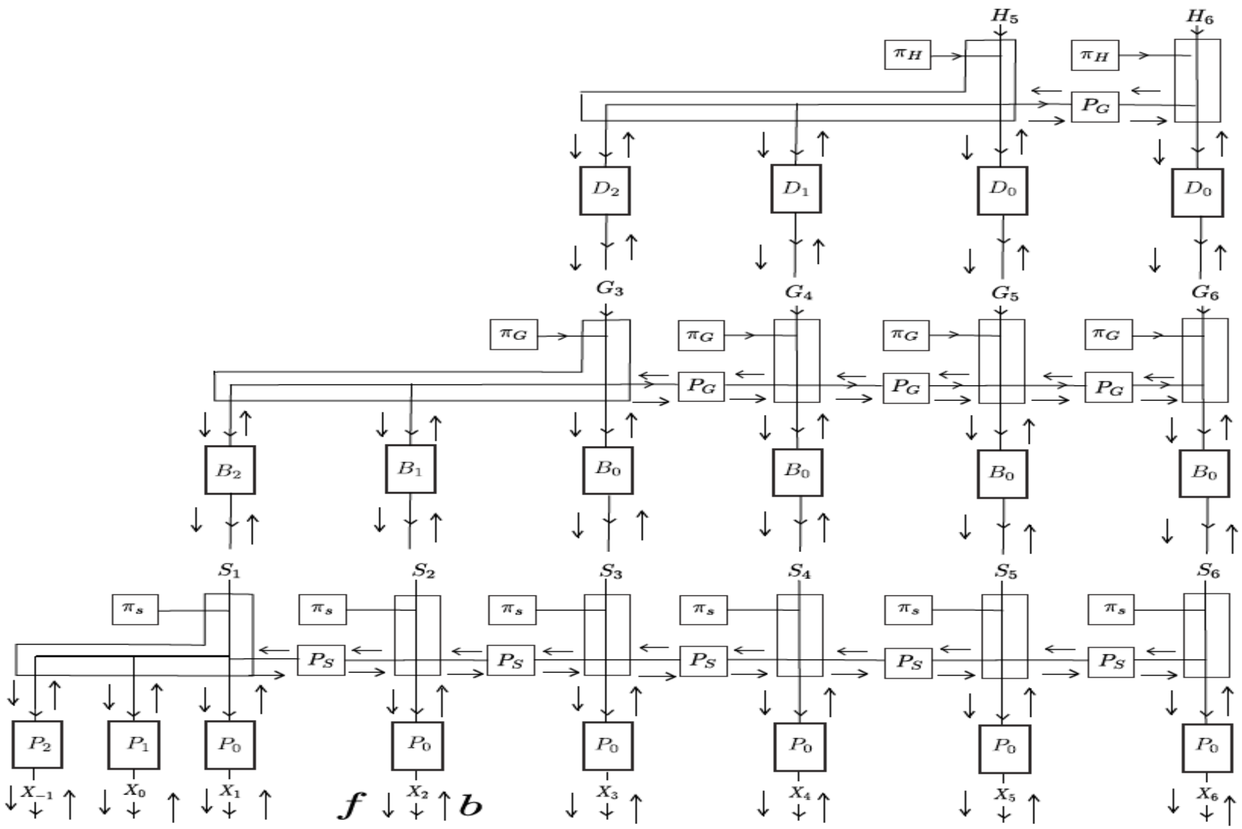
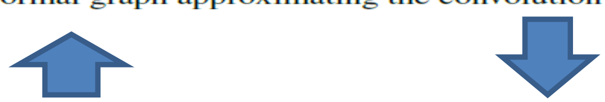


Fig. 6. The three-layer normal graph approximating the convolution graph of Figure 1 for $N = 6$.



Queries
(backward flow)

Answers
(forward flow)

Application 3: Multi-layer convolution graph (cont.)

- $d = 27$ character set - HMM approximation
- EM algorithm and manual the triplets from the text.
- Embedding variables $M_S = 59$, $M_G = 100$ and $M_H = 100$.

i think we are in rats alley where the dead men lost their bones
[T.S. Eliot, "The Waste Land"]

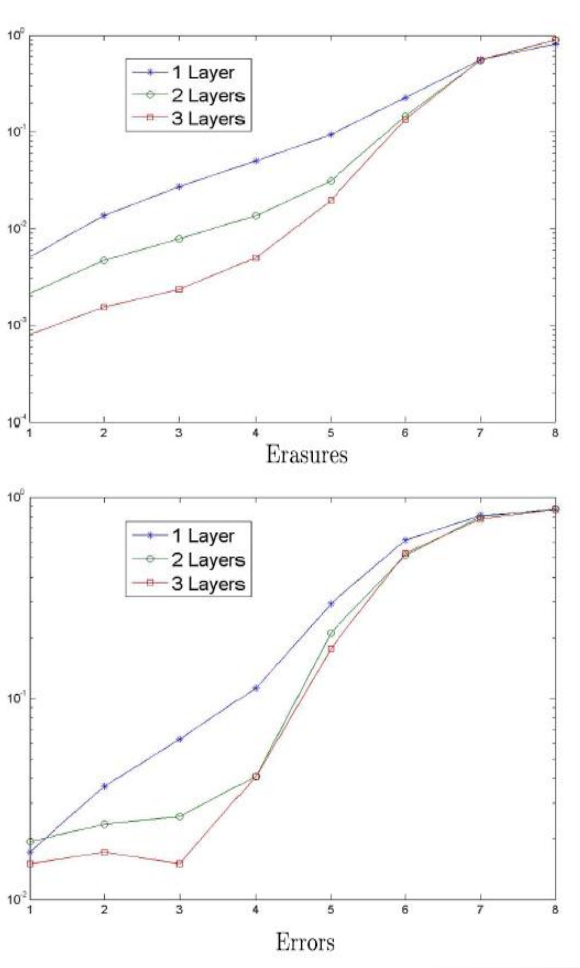
Incomplete input: **re~the??**
one- and two-layer graph, one error: **re~their**
three-layer graph, no error: **re~the~d**

Incomplete input: **o~~~the?**
one- and two- layers: **ost~the~**
even if in the two-layer response there is an equal maximum probability on both **~** and **i**
three-layers increase the probability on **i**

Wrong input: **re~tke~m**
One- two-layers, errors; three layers, no error: **re~the~d**

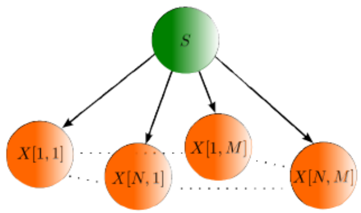
Input: **lbeherde**
one- two-layers, errors; three-layers, no error: **e~the~de**

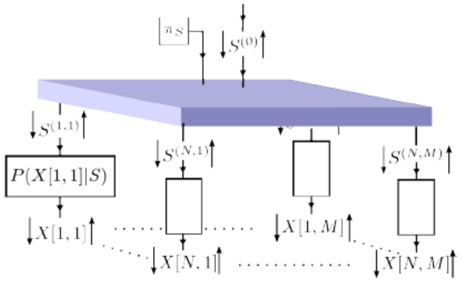
Arbitrary input: **asteland**
three-layers (getting closer to the dataset): **k~we~are**



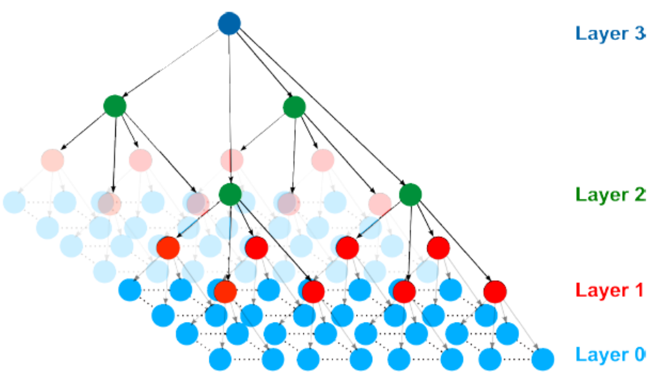
F. A. N. Palmieri and A. Buonanno, " Belief Propagation and Learning in Convolution Multi-layer Factor Graphs," *Proceedings of 4th International Workshop on Cognitive Information Processing, CIP2014*, May 26-28, 2014, Copenhagen, Denmark.

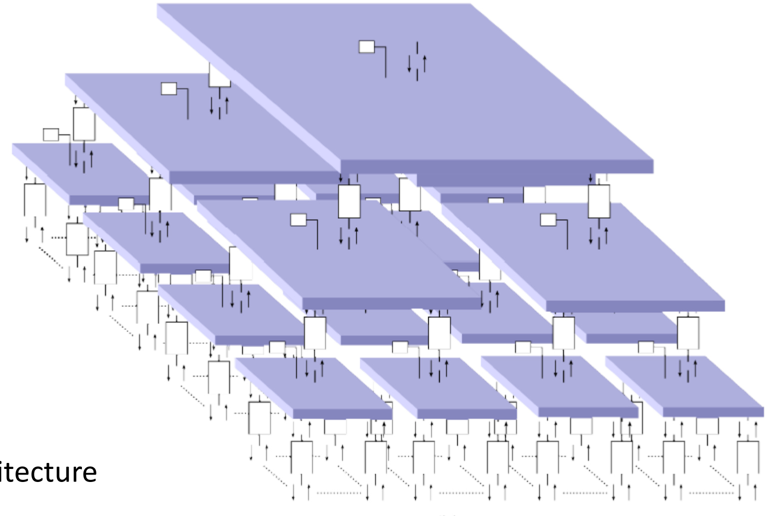
Application 4: Deep believe graph for images





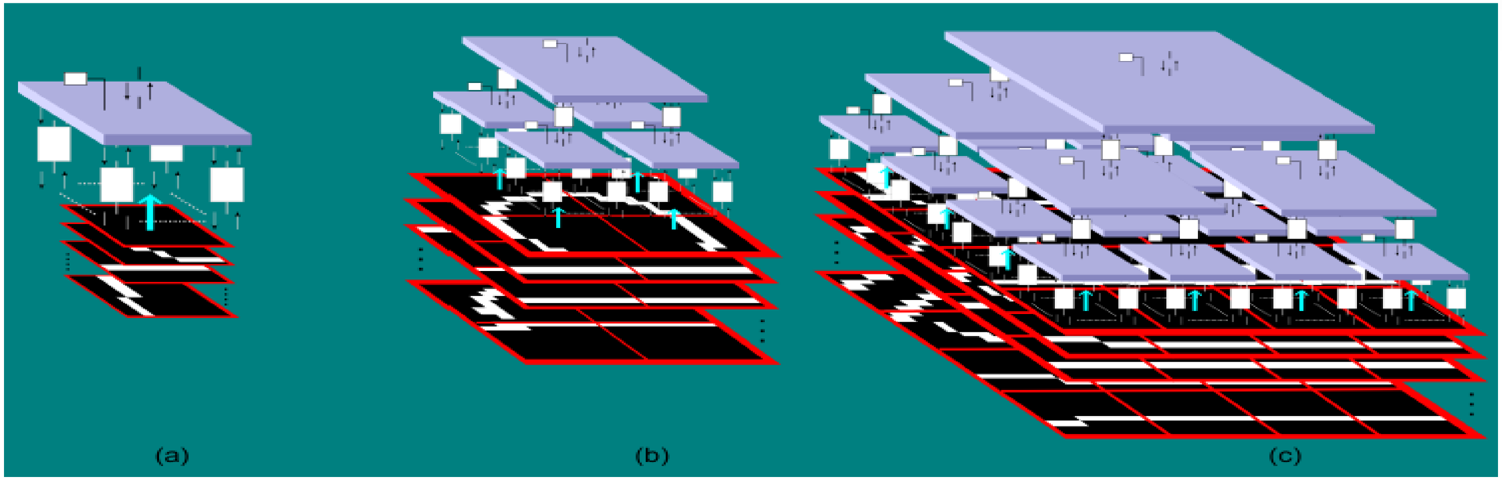
Latent Variable Model (LVM)



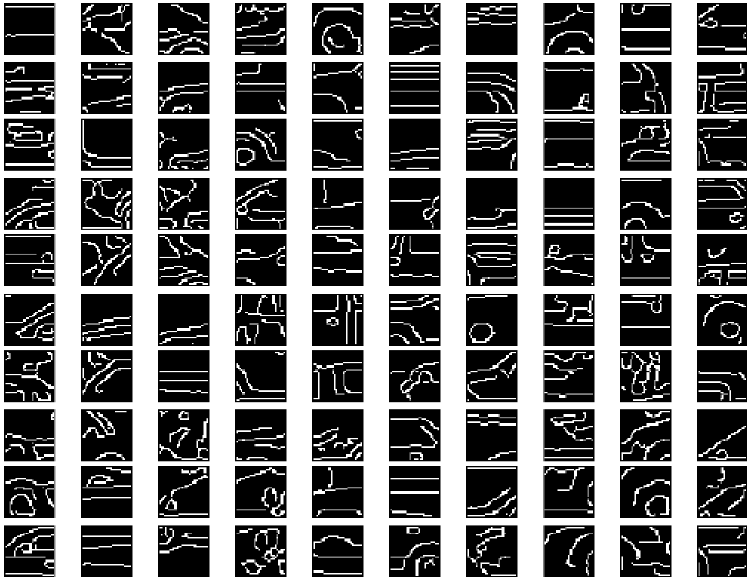


Quad-Tree Architecture

Application 4: Deep belief graph for images (cont)



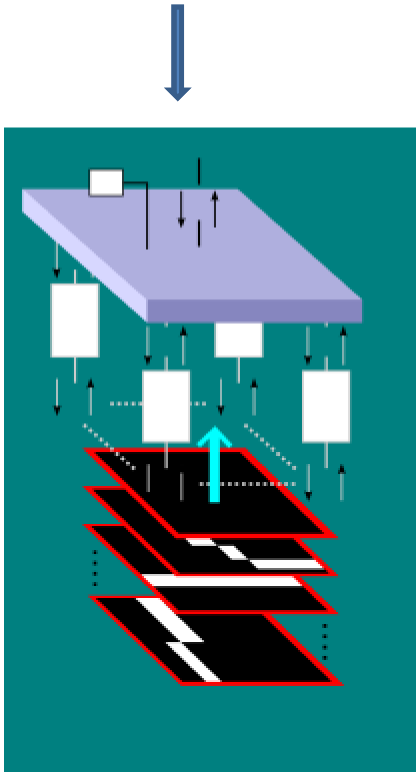
Learn one layer at a time (unsupervised)



32x32 images from the training set;
Filtered and reduced to BW
from Caltech101 (cars)

Application 4: Deep believe graph for images (cont)

Delta distribution



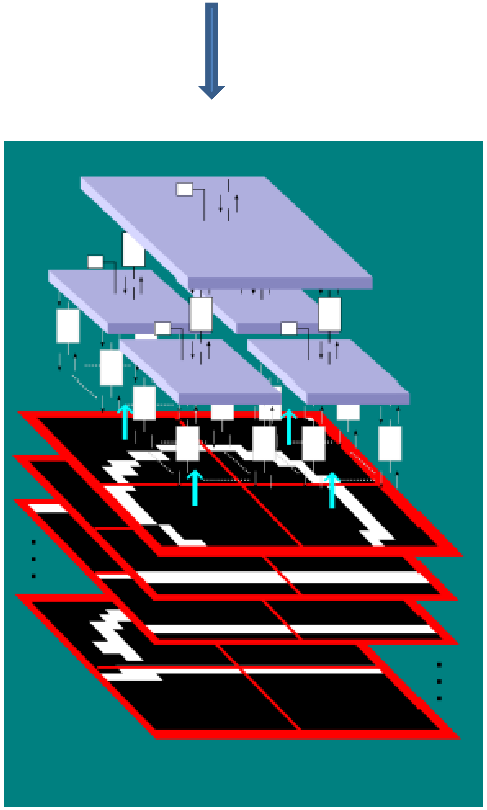
Bottom layer
8x8 patches

Embedding space size = 100

100 8x8 learned prototypes

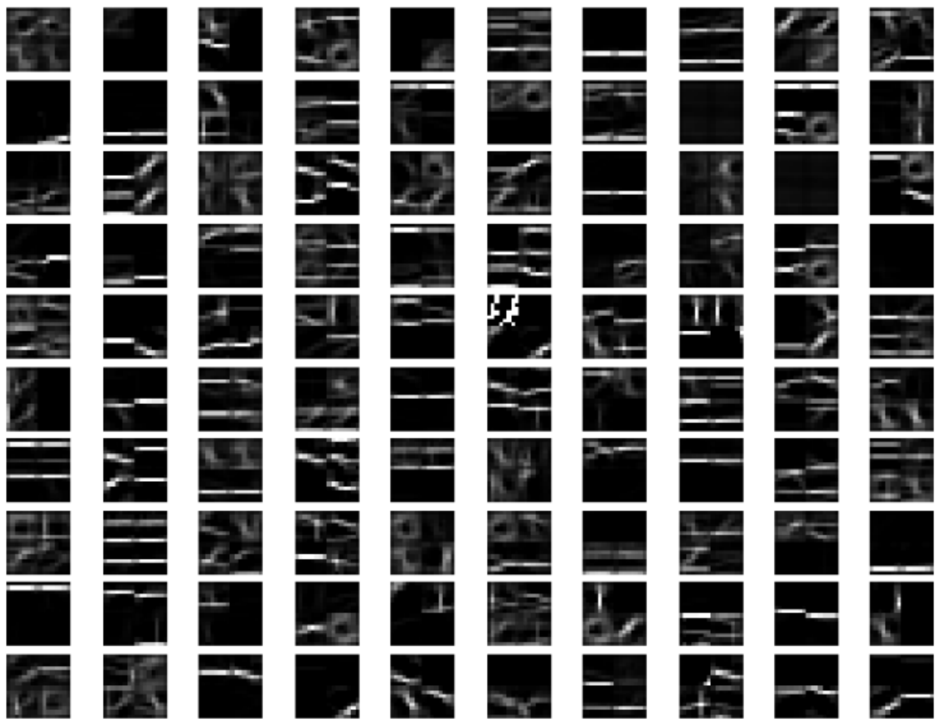
Application 4: Deep believe graph for images (cont)

Delta distribution



Two layers

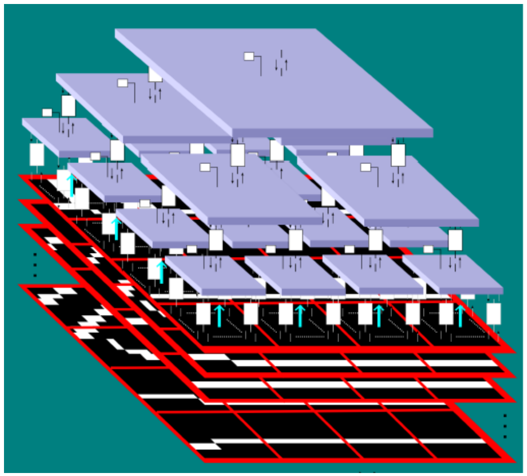
Embedding space size = 300



100 out of 300 16x16 learned prototypes

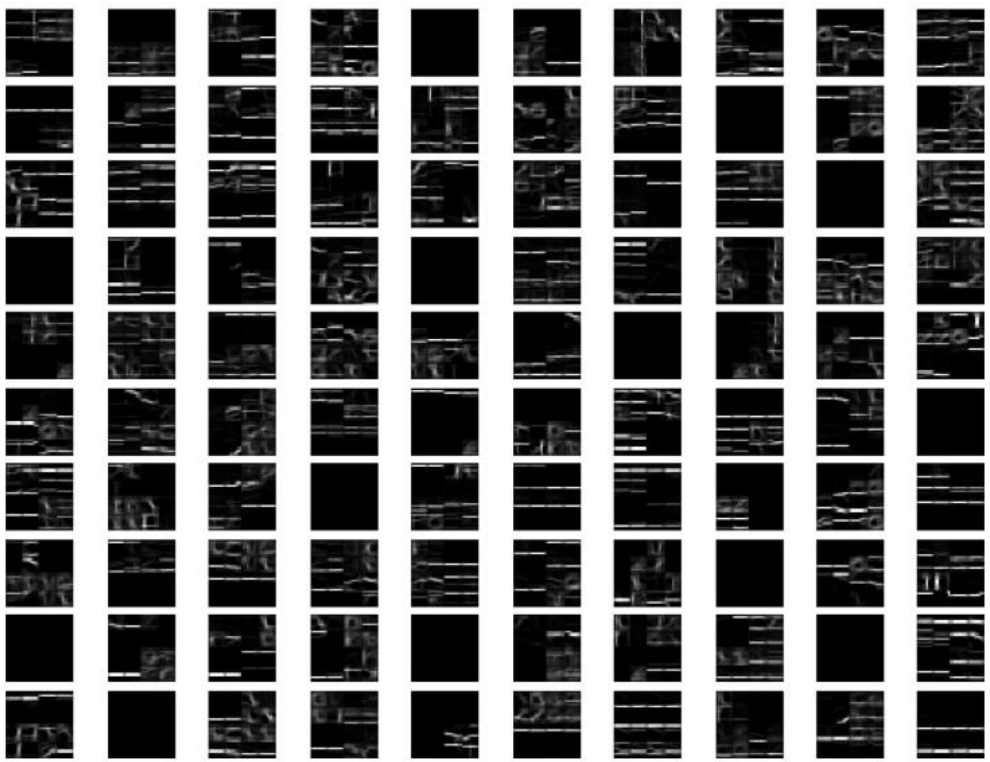
Application 4: Deep believe graph for images (cont)

Delta distribution



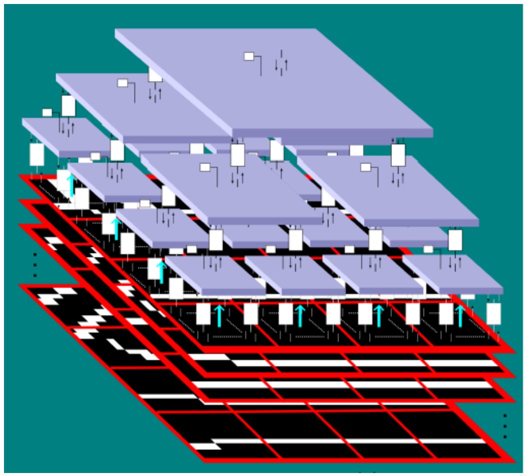
Three layers

Embedding space size = 300

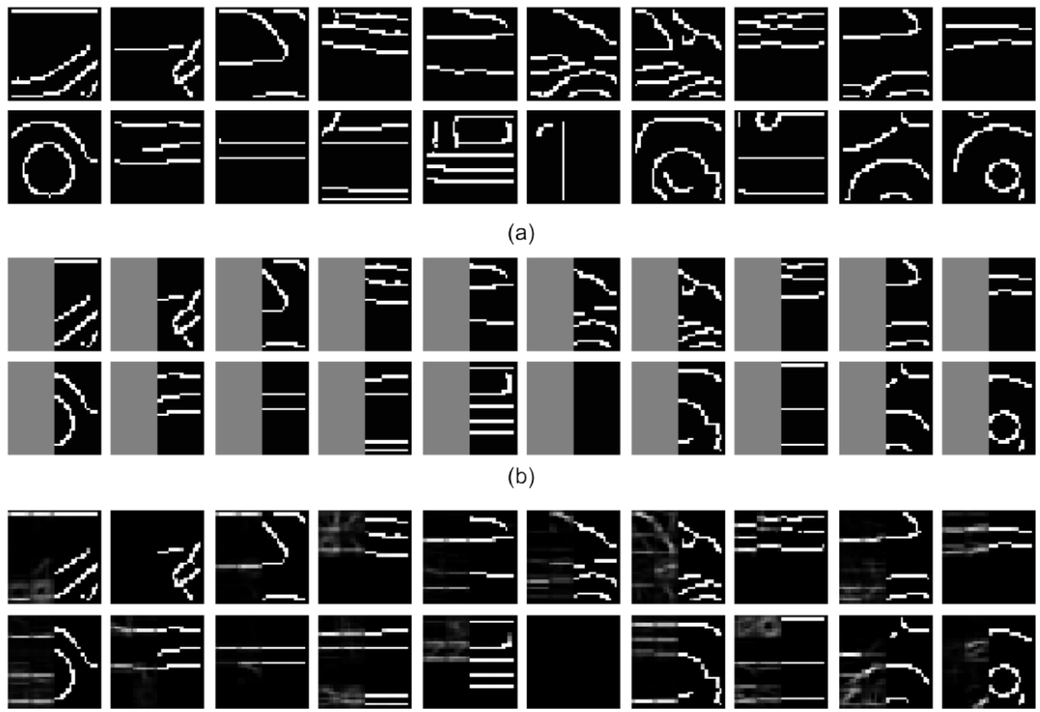


100 out of 300 32x32 learned prototypes

Application 4: Deep believe graph for images (cont)



Associative recall



(a)

(b)

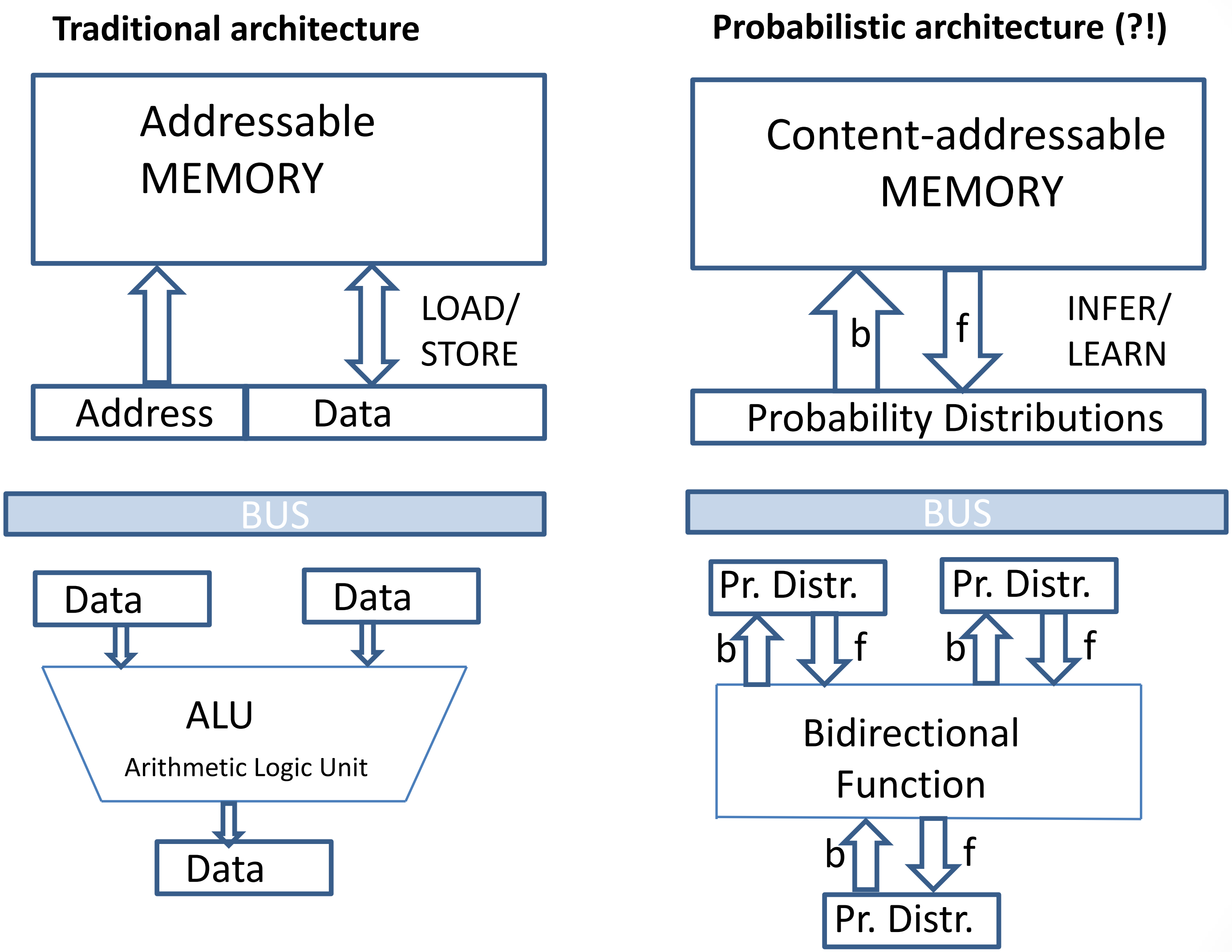
(c)

injected backward messages

Mean forward messages

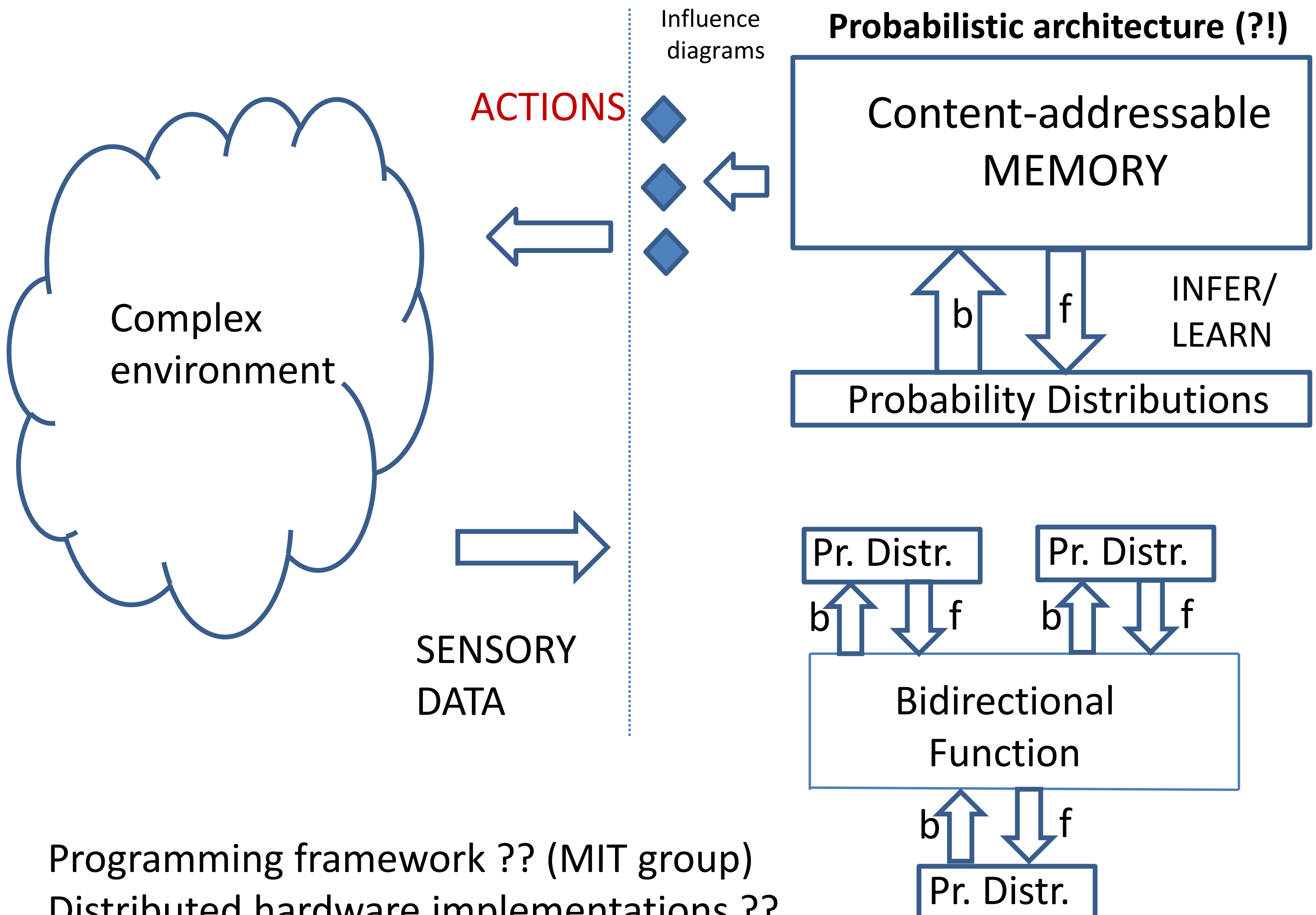
A. Buonanno and F.A.N. Palmieri, "Towards Building Deep Networks with Bayesian Factor Graphs", arXiv:1502.04492, Feb 2015, submitted for journal publication

Probabilistic computers ???



Stanford - April 2015

Probabilistic computers ???



Programming framework ?? (MIT group)
Distributed hardware implementations ??

Conclusions and open questions

- The Bayesian framework of factor graphs in reduced normal form appears to be very flexible to be applied as a general paradigm ;
- Bidirectional systems are very nice to work with and promise to change the way we think about signal processing;
- Multi-layer factor graphs architectures can be considered to build deep-believe networks if we address the computational issues;
- Beyond hard logic for probabilistic computers;

- Applications of this framework to model complex scenes;
- Study the proper architecture (grow the architectures adaptively);
- Make the probability pipelines scale in complexity;
- New hardware/languages to manipulate uncertainties
- Introduction of control with actions nodes (Influence Diagrams)

Thanks for your kind attention.

francesco.palmieri@unina2.it