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
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....

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

\[
p(X_1X_2...X_N) \quad \text{(joint pdf)}
\]

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

Group the variables in \( M \) subsets \( \{S_c, c = 1, ..., M\} \) (cliques)

\[
p(X_1X_2...X_N) \propto \prod_{c=1}^{M} \Psi_c \left( \bigcap_{j \in S_c} X_j \right) \quad \text{(\( \Psi_c \) potential functions)}
\]

\[
p(X_1X_2...X_N) \propto \exp \left( \sum_{c=1}^{M} \phi_c \left( \bigcap_{j \in S_c} X_j \right) \right) \quad \text{(\( \phi_c \) energy functions)}
\]

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

\[
p(X_1X_2...X_N) = \prod_{j=1}^{N} p(X_j|X_{j+1}...X_N)
\]

drop some conditioning variables using conditional independence assumptions. There are \( N! \) ways of rearranging the variables.
What kind of Bayesian graph?

- **Undirected graph**
  - $X_1$ connected to $X_2$ and $X_3$

- **Directed graph**
  - $X_1$ connected to $X_2$ and $X_3$

- **Factor graph**
  - $p(X_1)$ to $X_1$
  - $p(X_2|X_1)$ to $X_2$
  - $p(X_3|X_2X_1)$ to $X_3$

**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)

---


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 \( X_1 \)

\[ p(x_1|X_2 = x_2^0) = \frac{p(X_2 = x_2^0|x_1)\pi_{X_1}(x_1)}{p(X_2 = x_2^0)} \propto p(X_2 = x_2^0|x_1)\pi_{X_1}(x_1) \]

\[ b_{X_1}(x_1) = \int_{X_2} p_{X_2|X_1}(x_2|x_1)\delta(x_2 - x_2^0) \, dx_2 \]

\[ f_{X_2}(x_2) = \int_{X_1} p_{X_2|X_1}(x_2|x_1)\pi_{X_1}(x_1) \, dx_1 \]
Example 1: (cont.)

\[ p_{X_1X_2}(x_1x_2) = p_{X_2|X_1}(x_2|x_1)p_{X_1}(x_1) \]

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

\[
p(x_1|\pi_{X_2}) \overset{def}{=} \int_{X_2} p(x_1|X_2 = x_2)\pi_{X_2}(x_2)dx_2 \propto \pi_{X_1}(x_1) \int_{X_2} p_{X_2|X_1}(x_2|x_1)\pi_{X_2}(x_2)dx_2
\]

average posterior

Sum-Product rule
**Example 2:**

\[ p_{X_1 X_2 X_3}(x_1 x_2 x_3) = p_{X_3|X_2}(x_3|x_2)p_{X_2|X_1}(x_2|x_1)\pi_{X_1}(x_1) \]

**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

\[
\begin{align*}
  f_{X_1}(x_1) &= \pi_{X_1}(x_1); \\
  f_{X_2}(x_2) &= \pi_{X_2}(x_2); \\
  b_{X_2}(x_2) &\propto \int_{X_3} p_{X_3|X_2}(x_3|x_2)b_{X_3}(x_3)dx_3; \text{ (sum)} \\
  b_{X_2}(x_2) &\propto f_{X_2}(x_2)b_{X_3}(x_2); \text{ (product)} \\
  b_{X_1}(x_1) &\propto \int_{X_2} p_{X_2|X_1}(x_2|x_1)b_{X_2}(x_2)dx_2; \text{ (sum)} \\
  p(x_1|\pi_{X_2}, X_3 = x_3^0) &\propto f_{X_1}(x_1)b_{X_1}(x_1) \text{ (product)}
\end{align*}
\]

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:

A tree with 8 variables:

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

\[ C = A + B \] (arithmetic sum - deterministic function);
\[ A \in \{0, 1\}; \quad B \in \{0, 1\}; \quad C \in \{0, 1, 2\} \]

\[
P_1 = \frac{2}{4} I_2 \otimes 1^T_2 = \begin{bmatrix} \frac{1}{2} & 0 & 0 \frac{1}{2} \frac{1}{2} \frac{1}{2} \end{bmatrix};
\]

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

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

INPUTS: \( b_{A^2} = \begin{bmatrix} .1 \\ .9 \end{bmatrix} \); \( b_{B^2} = \begin{bmatrix} .99 \\ .01 \end{bmatrix} \); \( b_{C} = \begin{bmatrix} .33 \\ .33 \\ .33 \end{bmatrix} \);

OUTPUTS: \( f_{A^2} = \begin{bmatrix} .5 \\ .5 \end{bmatrix} \); \( f_{B^2} = \begin{bmatrix} .5 \\ .5 \end{bmatrix} \); \( f_{C} = \begin{bmatrix} .01 \\ .98 \\ .01 \end{bmatrix} \);

INPUTS: \( b_{A^2} = \begin{bmatrix} .5 \\ .5 \end{bmatrix} \); \( b_{B^2} = \begin{bmatrix} .99 \\ .01 \end{bmatrix} \); \( b_{C} = \begin{bmatrix} .2 \\ .8 \\ .0 \end{bmatrix} \);

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 (Bidyu 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:
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

\[
P(XY A_1...A_U C_1...C_V; \theta) = P(C_1...C_V | Y) \underbrace{P(Y | X ; \theta) P(XA_1...A_U)}_{\text{to be learned}}
\]

\[
\begin{align*}
L(\theta) &= \prod_{n=1}^{N} \sum_x \sum_y P_{X[n]Y[n]}(xy ; \theta) = \prod_{n=1}^{N} \sum_x \sum_y f'_{X[n]}(x) p_{Y | X}(y|x ; \theta) b'_{Y[n]}(y), \\
\ell(\theta) &= \log(L(\theta)) = \sum_{n=1}^{N} \log \left( f^{T}_{X[n]} \theta b_{Y[n]} \right) + \sum_{n=1}^{N} \log \left( K_{f_{X[n]}} K_{b_{Y[n]}} \right)
\end{align*}
\]

\[
\begin{align*}
\left\{ \begin{array}{c}
\min_{\theta} - \sum_{n=1}^{N} \log \left( f^{T}_{X[n]} \theta b_{Y[n]} \right), \\
\theta \quad \text{row - stochastic}
\end{array} \right\} & \quad \begin{array}{c}
\min_{\theta} \sum_{n=1}^{N} \sum_{i=1}^{M_y} b_{Y[n]}(j) \log \frac{b_{Y[n]}(j)}{f_{Y[n]}(j)}, \\
\theta \quad \text{row - stochastic}
\end{array}
\end{align*}
\]

ML learning \quad Minimum KL-divergence learning
EM Algorithms:

ML Algorithm:
(1) \( \theta_{lm} \leftarrow \frac{\sum_{n=1}^{N} \theta_{lm}}{\sum_{n=1}^{N} f_{X[n]}(l)} \sum_{n=1}^{N} \frac{f_{X[n]}(l) b_{Y[n]}(m)}{f_{X[n]}(l) \theta b_{Y[n]}}, \)
(2) Row-normalize \( \theta \) and back to (1)

KL Algorithm:
(1) \( \theta_{lm} \leftarrow \frac{\sum_{n=1}^{N} \theta_{lm}}{\sum_{n=1}^{N} f_{X[n]}(l)} \sum_{n=1}^{N} \frac{f_{X[n]}(l) b_{Y[n]}(m)}{\sum_{i=1}^{M X} \theta_{lm} f_{X[n]}(i)}, \)
(2) Row-normalize \( \theta \) and back to (1).

VITERBI-like Algorithm:
(1) \( e_{X[n]} = I_{Max}(f_{X[n]}) + \delta 1_{M X \times 1}; \)
\( e_{Y[n]} = I_{Max}(b_{Y[n]}) + \delta 1_{M Y \times 1}; \)
(2) \( \theta = \sum_{n=1}^{N} e_{X[n]} e_{Y[n]}^{T}; \)
(3) Row-normalize \( \theta \).

VARIATIONAL Algorithm:
(1) \( \theta_{lm} \leftarrow \delta + \sum_{n=1}^{N} f_{X[n]}(l) b_{Y[n]}(m), \)
(2) Row-normalize \( \theta \).

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

Simulink Library

It uses two-way connection ports

Figure 1: FGrm 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

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!

Application 2: Tracking objects with cameras

Salerno (Italy) harbour (3 commercial cameras)

Typical views
Application 2: Tracking objects with cameras (cont.)

Pinhole model

Image coordinates $\begin{pmatrix} x^i_k \\ y^i_k \end{pmatrix}$ = World coordinates $\begin{pmatrix} X_k \\ Y_k \end{pmatrix}$

Homography matrix (learned from calibration points)

$\begin{align*}
x^i_k &= \frac{x^i_k}{1} = \frac{h_{11} X_k + h_{12} Y_k + h_{13}}{h_{31} X_k + h_{32} Y_k + h_{33}} = q^i_1(X_k, Y_k), \\
y^i_k &= \frac{y^i_k}{1} = \frac{h_{21} X_k + h_{22} Y_k + h_{23}}{h_{31} X_k + h_{32} Y_k + h_{33}} = q^i_2(X_k, Y_k), \\
\dot{x}^i_k &= \frac{dx^i_k}{dt} = q^i_3(X_k, Y_k, \dot{X}_k, \dot{Y}_k), \\
\dot{y}^i_k &= \frac{dy^i_k}{dt} = q^i_4(X_k, Y_k, \dot{X}_k, \dot{Y}_k).
\end{align*}$

World coordinates $\begin{pmatrix} X_k/\lambda^i_k \\ Y_k/\lambda^i_k \\ 1/\lambda^i_k \end{pmatrix}$ = Image coordinates $\begin{pmatrix} x^i_k \\ y^i_k \\ 1 \end{pmatrix}$

$\begin{align*}
X_k &= \frac{X_k}{\lambda^i_k} = \frac{r_{11} x_k^i + r_{12} y_k^i + r_{13}}{r_{31} x_k^i + r_{32} y_k^i + r_{33}} = g^i_1(x^i_k, y^i_k), \\
Y_k &= \frac{Y_k}{\lambda^i_k} = \frac{r_{21} x_k^i + r_{22} y_k^i + r_{23}}{r_{31} x_k^i + r_{32} y_k^i + r_{33}} = g^i_2(x^i_k, y^i_k), \\
\dot{X}_k &= \frac{dX_k^i}{dt} = g^i_3(x^i_k, y^i_k, \dot{x}_k^i, \dot{y}_k^i), \\
\dot{Y}_k &= \frac{dY_k^i}{dt} = g^i_4(x^i_k, y^i_k, \dot{x}_k^i, \dot{y}_k^i),
\end{align*}$

- 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):

\[
\begin{align*}
    f_{s_k} & = \{ A m_{s_k} + A \Sigma_{s_k} A^T \}, \\
    f_{\mathbf{s}_k^M} & = \{ m_{f_{s_k}^M} + m_{f_{w_k}} + \Sigma_{f_{s_k}^M} + \Sigma_{f_{w_k}} \}, \\
    b_{s_k^0} & = \{ m b_{s_k^M} - m f_{w_k} + \Sigma b_{s_k^M} + \Sigma f_{w_k} \}, \\
    b_{s_k-1} & = \{ (A^T \Sigma_{s_k}^{-1} A)^{-1} A^T \Sigma_{s_k}^{-1} m b_{s_k^M}, (A^T \Sigma_{b_{s_k^M}}^{-1} A)^{-1} \}.
\end{align*}
\]

\[
\begin{align*}
    f_{s_k} & = f_{s_k} \circ (\otimes_{j=1}^N f_{s_k}^j), \\
    b_{s_k^M} & = b_{s_k} \circ (\otimes_{j=1}^N f_{s_k}^j), \\
    b_{s_k^i} & = b_{s_k} \circ f_{s_k} \circ (\otimes_{j=1,j\neq i}^N f_{s_k}^j), \quad i = 1, \ldots, N.
\end{align*}
\]

(Kalman filter equations “pipelined”)

Application 2: Tracking objects with cameras (cont.)

No calibration error (covariances amplified $10^6$)

With calibration error ($10^{-3}$; $10^{-4}$)

With forward and backward propagation

Only forward propagation

Background subtraction algorithm

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)
- Convolutional paradigms in Bayesian factor graphs?
- Convolutional structures better than trees account for short distance chained dependences;
- Expansion to hierarchies to capture long-term dependence at a gradually increasing scale.

Many many loops!!

It appears intractable for message propagation;

Stationarity allows a transformation

Fig. 1. Example of three-layer convolution graph with overlap $M = 2$ on all layers.
Application 3: Multi-layer convolution graph (cont.)

Latent model

Explicit mapping to product space

Junction tree

\[ P = P(X_{n-1}X_nX_{n+1}|X_{n-2}X_{n-1}X_n) = 1_d \otimes I_d \otimes I_d \otimes \frac{1}{d} 1_d^T. \]

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


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

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

Embedding space size = 100

Bottom layer 8x8 patches

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)

Probabilistic computers

Traditional architecture

Addressable MEMORY

LOAD/STORE

Address | Data

BUS

ALU
Arithmetic Logic Unit

Data

Probabilistic architecture (?!)

Content-addressable MEMORY

INFER/LEARN

Probability Distributions

BUS

Pr. Distr.

Bidirectional Function

Pr. Distr.

Pr. Distr.
Probabilistic computers

Complex environment

Content-addressable MEMORY

Probability Distributions

Pr. Distr.

Bidirectional Function

Programmable hardware implementations

Programming framework (MIT group)
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