PROBABILISTIC SIGNAL PROCESSING ON GRAPHS

Francesco A. N. Palmieri

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

Graduate Students:
Amedeo Buonanno
Francesco Castaldo
Outline:

- Why graphs
- Manage uncertainties
- Types of graphs
- Examples: single block, small network, continuous densities «loopy graph»
- Learning in a graph – ML learning (EM)
- Application to learning non linear functions
- Application to Camera tracking
- Application to Deep multi-layer network
- The inference on the graph as a probabilistic computing machine
- Open Issues and future developments
Why Graphs?:

We think on graphs!

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

If everything were connected to everything:

“spaghetti”
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....

Model dependencies:

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

$$p(X_1X_2\ldots X_N) \quad \text{(joint pdf)}$$

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

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

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

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

Use the chain rule

$$p(X_1X_2\ldots X_N) = \prod_{j=1}^{N} p(X_j|X_{j+1}\ldots X_N)$$

drop some conditioning variables using conditional independence assumptions. There are $N!$ ways of rearranging the variables.
What kind of graph:

- **Undirected graph**
- **Directed graph**
- **Factor graph**

**Normal Graph** (Forney’s style)

- More workable model:
  - Much easier message propagation
  - Unique rules for learning

(this example has a loop)
**Example 1:**  (to see how message propagation works)

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

Possible use: observe \( X_2 = x_2^0 \) and infer on \( X_1 \)

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

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

\[ f_{X_2}(x_2) = \int_{x_1} p_{X_2|X_1}(x_2|x_1) \underbrace{p_{X_1}(x_1)}_{f_{X_1}(x_1)} dx_1 \]
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 \( X_1 \)

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

Sum-Product rule
Example 2:  \[ p_{X_1 X_2 X_3}(x_1 x_2 x_3) = p_{X_3|X_2}(x_3|x_2)p_{X_2|X_1}(x_2|x_1)p_{X_1}(x_1) \]

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

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

One latent variable and three children (Bayesian clustering)

A tree with 8 variables

Three parents and a child
A numerical example:

$C = A + B$ (arithmetic sum); $A \in \{0, 1\}$; $B \in \{0, 1\}$; $C \in \{0, 1, 2\}$

deterministic function

$P_1 = \frac{3}{4} I_2 \otimes 1^T_2 = \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} & 0 & 0 \\
0 & 0 & \frac{1}{2} & \frac{1}{2}
\end{bmatrix}$; $P_3 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}$;

$P_2 = \frac{3}{4} 1^T_2 \otimes I_2 = \begin{bmatrix}
\frac{1}{2} & 0 & \frac{1}{2} & 0 \\
0 & \frac{1}{2} & 0 & \frac{1}{2}
\end{bmatrix}$

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

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

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: (embedded)

- The factor graph in normal form reduces the system to one-in/one-out blocks
- Each block “sees” only local messages
- \( P(Y/X) \) is here a discrete-variable stochastic matrix
- EM approach on N training examples

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

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

\[
\ell(\theta) = \log(L(\theta)) = \sum_{n=1}^{N} \log \left( f_{X[n]}^{T} \theta b_{Y[n]} \right) + \sum_{n=1}^{N} \log \left( K_{f_{X[n]}} K_{b_{Y[n]}} \right)
\]

\[
\begin{align*}
\min_{\theta} & - \sum_{n=1}^{N} L[n] \log \left( f_{X[n]}^{T} \theta b_{Y[n]} \right), \\
\theta & \ \text{row - stochastic,}
\end{align*}
\]

**ML learning**

\[
\begin{align*}
\min_{\theta} & \sum_{n=1}^{N} L[n] \sum_{i=1}^{M_y} b_{Y[n]}(j) \log \frac{b_{Y[n]}(j)}{\sum_{i=1}^{M_X} \theta_{ij} f_{X[n]}(i)}.
\end{align*}
\]

**Minimum KL-divergence learning**
**EM learning:**

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

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

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

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

1. Simulations on a single block;
2. Varying sharpness \(^{\delta}E: 1-10\)
3. Similar behaviour for more complicated architectures
4. Greedy search: Local minima (multiple restarts)

Application 1: Learning a Nonlinear Function

1. Soft quantization/dequantization
   (triangular likelihoods with entropic priors)

2. Map input variables to an embedding space

3. Minimize $KL(bY || fY)$
   
   - Not to challenge techniques for nonlinear adaptive filters (SVM, NN, RBF, ...);
   
   - Provide a technique for fusing categorical discrete data into a unique framework;
   
   - Numerous applications in signal processing


UConn - Feb 21, 2014
Application 1: Learning a Nonlinear Function (cont.)

Bidirectional quantizer

\[ \pi_X^E \propto \left( e^{H(X|1)}, e^{H(X|2)}, \ldots, e^{H(X|M)} \right) \]

Entropic priors

Application 1: Learning a Nonlinear function (cont.)

\[ Y = \frac{1}{20}(X - \frac{1}{2})^3; \ X \in [0, 1]; \ M_X = 20, \ M_Y = 5; \]

- Difficulties with the inverse map in the flat area (very small first derivative).

![Graphs showing data points and trend lines](image)

- Grayscale picture of \( f_{Y_a[n]} \) (estimated points superimposed)
Application 1: Learning a Nonlinear function (cont.)

\[ Y_a = \sqrt{X_{a1}^2 + X_{a2}^2} + \sin X_{a1} + 1, \quad M_1 = M_2 = M_Y = 4 \]

\[ N = 100 \text{ random independent pairs from } [0, 1]^2 \text{ for } X_{a1} \text{ and } X_{a2}. \quad M_1 = M_2 = M_Y = 4. \]

0 - backward
* - forward
Application 2: Tracking objects with cameras

Gaussian messages (means and covariances):

\[
\begin{align*}
    f_{s_k} &= \{ Amf_{s_{k-1}}, A^T \Sigma f_{s_{k-1}} A \}, \\
    f_{b_k} &= \{ m_f_{s_{k-1}}, \Sigma f_{s_{k-1}} \}, \\
    b_{s_{k-1}} &= \{ (A^T \Sigma b_{s_{k-1}} A)^{-1} A^T \Sigma b_{s_{k-1}} m_f_{s_{k-1}}, (A^T \Sigma b_{s_{k-1}} A)^{-1} \}.
\end{align*}
\]

(Kalman filter equations “pipelined”)

Application 2: Tracking objects with cameras (cont.)

Pinhole model

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

World coordinates \( \rightarrow \) Image coordinates

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

Image coordinates \( \rightarrow \) 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)

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

- Local first-order approximations for Gaussian pdf propagation;
- Gaussian noise on the homography matrix.
Application 2: Tracking objects with cameras (cont.)

Salerno (Italy) harbour (3 commercial cameras)


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

Background subtraction algorithm

- No calibration error (covariances amplified $10^6$)
- With calibration error ($10^{-3}$; $10^{-4}$)

With forward and backward propagation

Only forward propagation
Application 3: Multi-layer convolution graphs

- 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.

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

\[ CF_1 \simeq P_1; \]
\[ CF_2 \simeq P_2; \]
\[ CF_3 \simeq P_3; \]

C row-stochastic

UConn - Feb 21, 2014
Fig. 6. The three-layer normal graph approximating the convolution graph of Figure 1 for $N = 6$. 
Application 3: Multi-layer convolution graph (cont.)

Matlab/Simulink implementation using bi-directional ports assembled graphically
Simulation on controlled environment and limited computational complexity:
- \(d = 27\) character set from The Waste Land by T. S. Eliot:

\[
\text{i think we are in rats alley where the dead men lost their bones}
\]
- Used HMM approximation
- EM algorithm and manual the triplets from the text.
- Embedding variables thave sizes \(M_S = 59, M_G = 100\) and \(M_H = 100\).
- Belief propagation in the system is run for 30 steps In learning we have used 10 epoques and the greedy ML algorithm

(1) present a set of 8 characters \(X_{-2}\) through \(X_6\) to the bottom;
(2) perform belief propagation on the first layer with no connections to the second layer;
(3) collect the forward messages for \(S_1\) through \(S_6\) and use them to learn the latent model on triplets for the second layer;
(4) back to (1) with a new set of 8 characters at the bottom by sliding on the text.

- Tested the system at \(X_{-2}, X_{-1},...,X_6\) for various layer configurations by in-jecting our queries through the backward messages and collecting the results as the forward messages.
Applicaton 3: Multi-layer convolution graph (cont.)

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~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

----→Extension to Larger datasets and images
• Very consistent results on inference and learning with Bayesian networks;

• Many successful applications are based on Bayesian paradigms;

• Will the probability pipelines scale in complexity?

• New architectures/languages that include uncertainties?
Probabilistic computers???

Traditional architecture

Addressable MEMORY

LOAD/STORE

Address Data

Data ALU
Arithmetic Logic Unit

Data

Probabilistic architecture (?!)

Content-addressable MEMORY

INFER/LEARN

b f

Probability Distributions

Pr. Distr.
Pr. Distr.

b f

b f

Pr. Distr.
Probabilistic computers???

Complex environment

SENSSORY DATA

ACTIONS

Content-addressable MEMORY

Probability Distributions

INFER/LEARN

Bidirectional Function

Pr. Distr.

Pr. Distr.

Pr. Distr.
Conclusions and future directions:

The Bayesian framework is effective in a number of signal processing applications

Beyond hard logic

Bidirectional probability propagation shows promising impact on the applications

Complexity scale in dealing with unstructured environments

Probabilistic computers

Extensions of signal processing to (stochastic) control of action in integration with uncertainty
Thanks for your kind attention.

francesco.palmieri@unina2.it