Factor Graph Tutorial

HU, Pili

March 22, 2012

Abstract

Factor graph is one general problem solving technique (personally, I prefer to call it a technique rather than a model). It can solve marginalization problem very efficiently. Our traditional model, which involve probability marginalization, like Hidden Markov Model, Bayesian Network, Markov Random Field, etc, can all be translated into a factor graph representation. This article first constructs a toy motivating example to show the benefit of using factor graph. With the observation from the examples, factor graph is deduced. Notations, definitions, and descriptions mainly follow the original works, [4] [3]. Then we present two selected applications of FG, one for modeing underlying system dynamics[5] , and the other for modeling influence in social network[7]. We end this article by relating factor graphs to other existing graphical models and algorithms, together with some discussion.

Contents

1 A Motivating Example

1.1 Marginalization Problem

Consider a joint probability distribution:

$$p(\vec{x}) \tag{1}$$

where $\vec{x} = \{x_1, x_2, \ldots, x_n\}$.

Marginalization operator:

$$p_1(x_1) = \sum_{x_2} \cdots \sum_{x_{n-1}} \sum_{x_n} p(\vec{x}) \tag{2}$$

Assuming discrete variable. For continous ones, substitute sum with integral accordingly.

For simplicity of notation, introduce the shorthand "summary" notation:

$$\begin{aligned} p_1(x_1) &= \sum_{\sim\{x_1\}} p(\vec{x}) && (3)\\ &= \sum_{\{x_2, x_3, \ldots, x_n\}} p(\vec{x}) && (4)\\ &= \sum_{x_2} \cdots \sum_{x_{n-1}} \sum_{x_n} p(\vec{x}) && (5) \end{aligned}$$

The marginalization problem is defined as: Given $p(\vec{x})$, find $p_i(x_i)$. This problem can be generalized as summary for more than one variables.

1.2 Inference Problem

The inference problem is defined as: Given $p(\vec{x}, \vec{y})$, and the observed value of $\vec{y}$, say $\hat{\vec{y}} = \{\hat{y_1}, \hat{y_2}, \ldots, \hat{y_n}\}$, find the most probable configuration of $\vec{x}$:

$$\vec{x}^* = \arg\max_{\vec{x}} \{p(\vec{x}|\hat{\vec{y}})\} \tag{6}$$

The conditional probability can be rewritten as:

$$\begin{aligned} p(\vec{x}|\hat{\vec{y}}) &= \frac{p(\vec{x}, \hat{\vec{y}})}{p(\hat{\vec{y}})} && (7)\\ p(\vec{x}|\hat{\vec{y}}) &\propto p(\vec{x}, \hat{\vec{y}}) && (8) \end{aligned}$$

Thus eqn(6) can be rewritten as:

$$\vec{x}^* = \arg\max_{\vec{x}} \{p(\vec{x}, \hat{\vec{y}})\} \tag{9}$$

We'll bridge the gap between the inference problem and marginalization problem defined above later. Now, we start with a toy example.

1.3 Inference with Four Binary Variables

Assume we have a joint distribution $p(a, b, c, d)$. Without any knowledge of the internal structure of the distribution, we can always write it as:

$$p(a, b, c, d) = p(a)p(b|a)p(c|a, b)p(d|a, b, c) \tag{10}$$

Now assume the distribution can be factorized in the following way:

$$p(a, b, c, d) = p(a)p(b)p(c|b)p(d|c) \tag{11}$$

We'll compare two ways of commputing

$$\max_{abcd} p(abcd)$$

using the following data:

$$p(a) = \begin{bmatrix} 0.1 & 0.9 \end{bmatrix} \tag{12}$$

$$p(b) = \begin{bmatrix} 0.2 & 0.8 \end{bmatrix} \tag{13}$$

$$p(c|b) = \begin{bmatrix} 0.4 & 0.6 \\ 0.6 & 0.4 \end{bmatrix} \tag{14}$$

$$p(d|c) = \begin{bmatrix} 0.3 & 0.7 \\ 0.8 & 0.2 \end{bmatrix} \tag{15}$$

1.3.1 Search Max on Joint Distribution Directly

First, we pretend there is no structure information available. Thus a naive way to compute the max is to evaluate the joint distribution everywhere on tha complete alphabets of variables, and then get maximum by comparison.

$$p(abcd) = \left[\begin{array}{c|c} \text{abcd} & \text{Probability} \\ \hline 0000 & 0.0024 \\ 0001 & 0.0056 \\ 0010 & 0.0096 \\ 0011 & 0.0024 \\ 0100 & 0.0144 \\ 0101 & 0.0336 \\ 0110 & 0.0256 \\ 0111 & 0.0064 \\ 1000 & 0.0216 \\ 1001 & 0.0504 \\ 1010 & 0.0864 \\ 1011 & 0.0216 \\ 1100 & 0.1296 \\ 1101 & 0.3024 \\ 1110 & 0.2304 \\ 1111 & 0.0576 \end{array}\right] \tag{16}$$

$$
\begin{aligned}
p(abcd^*) &= \max_{abcd} p(abcd) && (17)\\
&= p(1101) && (18)\\
&= 0.3024 && (19)
\end{aligned}
$$

Corresponding computation complexity:

- Function evaluation: $16 \times 4 = 64$
- Product: $16 \times 3 = 48$
- Comparison(for max operator): 15

1.3.2 Search Max Intelligently

Indeed, eqn(11) conveys useful information by the factorization of the joint probability.

Let's expand the maximization $p(a, b, c, d)$

$$
\begin{aligned}
\max_{abcd}\{p(abcd)\} &= \max_{abcd}\{p(a)p(b)p(c|b)p(d|c)\} && (20)\\
&= \max_{a}\{p(a)\}\max_{bcd}\{p(b)p(c|b)p(d|c)\} && (21)\\
&= \max_{a}\{p(a)\}\max_{d}\{\max_{bc}\{p(b)p(c|b)p(d|c)\}\} && (22)\\
&= \max_{a}\{p(a)\}\max_{d}\{\max_{c}\{\max_{b}\{p(b)p(c|b)\}p(d|c)\}\} && (23)
\end{aligned}
$$

$$
\max_{a}\{p(a)\} = \max_{a} f_a(a) = 0.9 \quad (24)
$$

$$
\begin{aligned}
\max_{b}\{p(b)p(c|b)\} &= \max_{b} f_{bc}(bc) && (25)\\
&= \max_{b} \left[\begin{array}{c|c} \text{bc} & \text{Probability} \\ \hline 00 & 0.08 \\ 01 & 0.12 \\ 10 & 0.48(*) \\ 11 & 0.32(*) \end{array}\right] && (26)\\
&= \left[\begin{array}{c|c} \text{c} & \text{Probability} \\ \hline 0 & 0.48 \\ 1 & 0.32 \end{array}\right] && (27)
\end{aligned}
$$

Denote $\max_b\{p(b)p(c|b)\}$ by $\mu_{bc}(c)$.

$$\max_c\{\mu_{bc}(c)p(d|c)\} = \max_c f_{cd}(cd) \tag{28}$$

$$= \max_c \left[\begin{array}{c|c} \text{cd} & \text{Probability} \\ \hline 00 & 0.144 \\ 01 & 0.336(*) \\ 10 & 0.256(*) \\ 11 & 0.064 \end{array}\right] \tag{29}$$

$$= \left[\begin{array}{c|c} \text{d} & \text{Probability} \\ \hline 0 & 0.256 \\ 1 & 0.336 \end{array}\right] \tag{30}$$

Denote $\max_c\{\mu_{bc}(c)p(d|c)\}$ by $\mu_{cd}(d)$.

$$\max_d\{\max_c\{\max_b\{p(b)p(c|b)\}p(d|c)\}\} \tag{31}$$

$$= \max_d\{\mu_{cd}(d)\} \tag{32}$$

$$= 0.336 \tag{33}$$

Thus we get final result:

$$\max_{abcd}\{p(abcd)\} = \max_a\{p(a)\} \times \max_d\{\mu_{cd}(d)\} \tag{34}$$

$$= 0.3024 \tag{35}$$

Again, we calculate the computation complexity:

- Function evaluation:

$$2 + 4 \times 2 + 4 \times 2 + 2 = 20$$

- Product:

$$0 + 4 \times 1 + 4 \times 1 + 0 + 1 = 9$$

- Comparison(for max operator):

$$1 + 2 \times 1 + 2 \times 1 + 1 = 6$$

1.3.3 Observation

We compare the complexity of two methods in table(1).

The observations:

- By properly using the structure of joint distribution, it's possible to reduce computation complexity.

Table 1: Comparison Between Two Methods

Items	Naive	Intelligent
Function	64	20
Product	48	9
Comparison	15	6

- The trick to reduce complexity in the second method is: "product" is distributive through "max". Thus we can separate some variables when evaluating the maximum of others. We'll address this issue in details later.
- How to reveal and utilize the structure in a systematic way is still a problem.

1.4 Inference with One Observed Variable

Here's another example. Assume c is observed to be 1 in the last example. What's the new most probable configuration of other variables?

1.4.1 Naive

As before, simply restrict the evaluation of functions only on points where $c = 1$.

$$p(abcd) = \left[\begin{array}{c|c} \text{abcd} & \text{Probability} \\ \hline 0010 & 0.0096 \\ 0011 & 0.0024 \\ 0110 & 0.0256 \\ 0111 & 0.0064 \\ 1010 & 0.0864 \\ 1011 & 0.0216 \\ 1110 & 0.2304 \\ 1111 & 0.0576 \end{array}\right] \tag{36}$$

The most probable configuration of the four variables is 1110, and corresponding probability is 0.2304(the joint probability, not the probability of $a = 1, b = 1, d = 0$ conditioned $c = 1$).

1.4.2 Intelligent

With the observation of $c = 1$, the joint distribution can be decomposed as:

$$\max_{abd}\{p(ab1d)\} = \max_{abd}\{p(a)p(b)p(c=1|b)p(d|c=1)\} \tag{37}$$

$$= \max_{a}\{p(a)\} \max_{b}\{p(b)p(c=1|b)\} \max_{d}\{p(d|c=1)\} \tag{38}$$

$$\max_a\{p(a)\} = \max_a f_a(a) = 0.9 \tag{39}$$

$$\max_b\{p(b)p(c=1|b)\} = \max_b f_{bc}(bc, c=1) \tag{40}$$

$$= \max_b \left[\begin{array}{c|c} \text{bc} & \text{Probability} \\ \hline 01 & 0.12 \\ 11 & 0.32 \end{array}\right] \tag{41}$$

$$= 0.32 \tag{42}$$

$$\max_d\{p(d|c=1)\} = \max_d \left[\begin{array}{c|c} \text{cd} & \text{Probability} \\ \hline 10 & 0.8 \\ 11 & 0.2 \end{array}\right] \tag{43}$$

$$= 0.8 \tag{44}$$

Thus the final maximum probability is given by:

$$\max_a\{p(a)\}\max_b\{p(b)p(c=1|b)\}\max_d\{p(d|c=1)\} \tag{45}$$

$$= 0.9 * 0.32 * 0.8 \tag{46}$$

$$= 0.2304 \tag{47}$$

1.4.3 Observation

Now that we validated the correctness of our intelligent method, we again compare the complexity as is in table(2).

Table 2: Comparison Between Two Methods

Items	Naive	Intelligent
Function	32	8
Product	24	4
Comparison	7	3

Besides previous observations on the value of "structure", we highlight one more thing:

- When the variable c is observed, the joint distribution function can be further decomposed! That is, in previous example, there is a relationship between b and d, so we evaluate max operator in the order of b, then c, then d. However, with the observation of c, the sub functions involving b and d are fully decoupled, this further reduces the complexity.
- This observation is indeed the notion of conditional independence, as is one major concern in some graphical models like MRF and BN.

2 Factor Graph

2.1 Specification

This section is illustrated on white board during the group meeting of MobiTeC.

Roadmap:

- Notion of semiring, distributive law. Generalize the building block operators. Unify marginalization problem and maximum problem introduced in the motivating example.
- Make graph representation of toy example and conclude the relationship.
- Rule1: connected component can be evaluated on its own.
- The rest part form a chain structure.
- Draw the "message", we denoted by μ in the toy example on corresponding edges.
- Draw one trivial message for $p(b)$ to unify the message passing like process.
- Conclude how function node processes: receive messages; multiply by local function; marginalization for destination variable; send out new message.
- Process of variable node is the same. Relationships: 1. local function is an identity function; 2. since the received messages are marginalized for itself, the marginalization at a variable node is trivial operation.
- Process of variable and function is unified.
- Conclude possibility to handle a chain.
- Conclude possibility to handle a tree with designated root.
- Conclude possibility to handle a tree without designated root.

This section ends by showing the computation ability of factor graph on acyclic graph. Next section deals with cycles.

2.2 Transformation

Transformation of loopy factor graph can result in tree structures. Thus ordinary sum-product algorithm can be used to obtain an exact solution.

Effect of clustering:

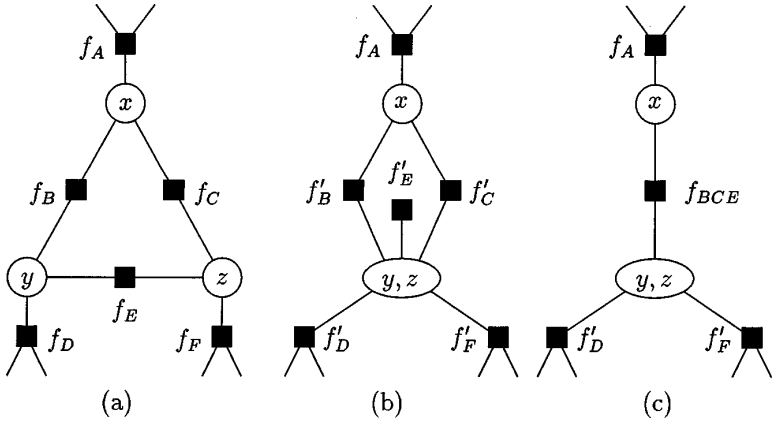

Figure 1: Transformation: Clustering[4]

- Cluster variable nodes: domain enlarged; no complexity increasing in local functions; increase complexity in sum-product algorithm (function evaluation).
- Cluster function nodes: do not increase complexity of variables; increase complxity in sum-product algorithm (sizes of messages are increased).

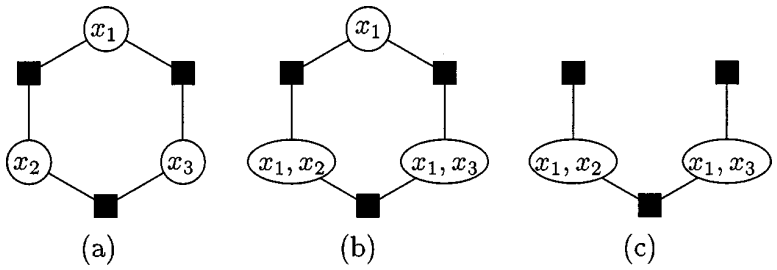

Figure 2: Transformation: Streching[4]

Effect of stretching:

- Simply stretching variables adds positional variables to some local functions. Local function complexity is not changed. Variable alphabets are enlarged.
- After stretching, redundant links or even redundant variables can be removed. By systematic stretching, all cycles can be eliminated.

2.3 Parallelization

The major advantage as we have already seen in previous examples is, the locality of problems structure is captured by Factor Graph. Besides lower-

ing the overall computation complexity, Factor Graph totally agrees with parallelization.

For example, the famous MapReduce framework can be utilized to implement factor graph. Here we provide a short description of how to map factor graph to MapReduce framework.

- Both function nodes and variable nodes are unified by one type of node, called unit. As we can see from the deduction of FG, the operations performed on them are also the same.
- Number of mappers is the same as number of reducers, and the same as number of units.
- For every mapper, it calculates the multiplication of all current messages together with the local function, and then summarize it for every neighbouring nodes. After that, it simply issue all new messages using the target node ID as the key.
- For every reducer, it collects the list of messages targeted for it and do nothing. Those collected messages can fit into next round of map.
- We run multiple round of MapReduce until a certain termination criterion is reached.
- To bootstrap the computation, the message list is initialized as an identity message.

3 Selected Applications

3.1 Model System Dynamics

This section follows the paper:
P. Mirowski and Y. LeCun. Dynamic factor graphs for time series modeling. *Machine Learning and Knowledge Discovery in Databases*, v:128–143, 2009.

Declaration: figures in this section are borrowed from the orignal paper[5]. We omit further citation for simplicity.

3.1.1 Quick Note of the Main Idea

Problem settings:

- System is composed of state variables and observation variables.
- Observation function maps state to observation.
- Dynamic function governs state transitions.

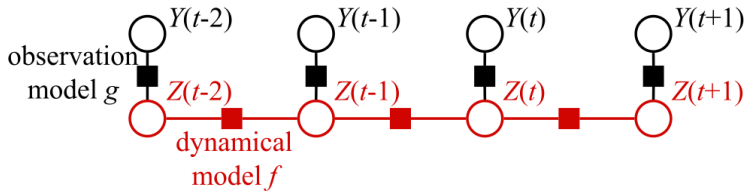

Figure 3: Single State Dependency(HMM)

- Aim at modeling high dimensional underlying dynamic, probably non-linear, but determinstc.

When current state is only dependent on previous one state, it is like an HMM model.

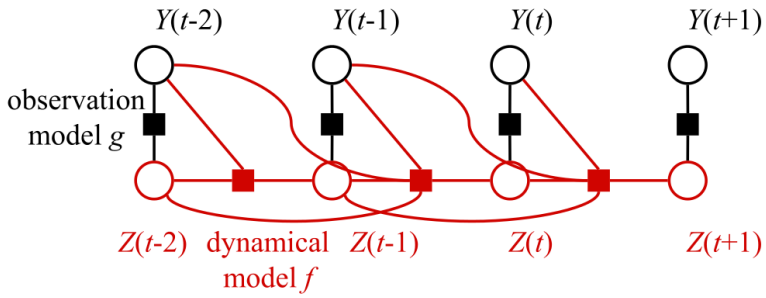

Figure 4: Multiple States and Observation Dependency

The state transition model can be more general that it may depend on several previous states, or even observations.

Formulation of paramterized FG:

$$L(\mathbf{W}, Y, Z) = E(\mathbf{W}, Y) + R_z(Z) + R(\mathbf{W}) \quad (48)$$

$$\tilde{Z} = \arg\max_{Z} L(\tilde{\mathbf{W}}, Y, Z) \quad (49)$$

$$\tilde{\mathbf{W}} = \arg\max_{\mathbf{W}} L(\mathbf{W}, Y, \tilde{Z}) \quad (50)$$

Since the paper operates in the negative log domain, they want to minimize L. The more likely some configuration is, the energy E is lower. As is depicted in fig(), energy function is the square error.

When parameters are fixed, eqn(49) acts as what ordinary factor graph does, namely, given all explicit function definitions, compute the marginalization problem. In ths application, paramters $\mathbf{W}$ is not determined. An usual way to tackle with this problem is by Expectation-Maximization schema (sometimes called Generalized Expectation Maximization algorithm). eqn(49) resembles the E-step, while eqn(50) resembles the M-step.

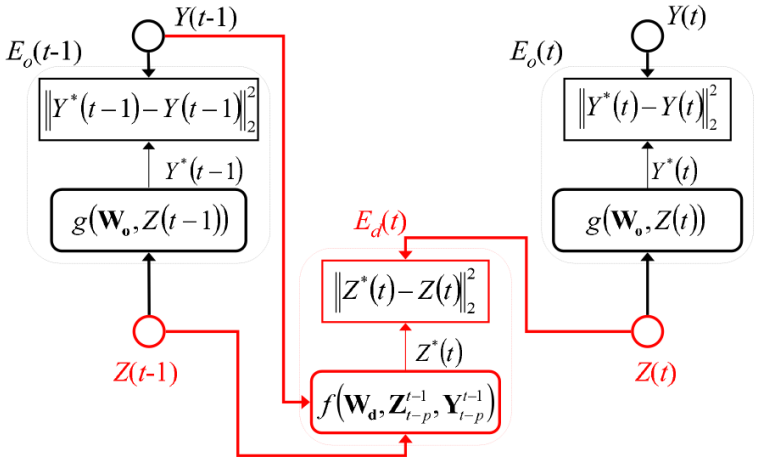

Figure 5: System Details

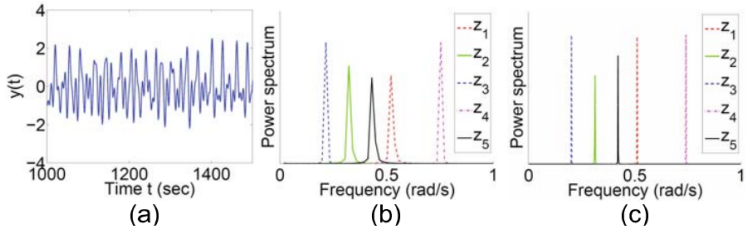

Figure 6: Training and Evaluation on Superimposed Sine Waves

To decouple 5 superimposed sine waves, the authors choose the architecture:

- 5 dimensional state variable.
- Dynamic function(f): 5 independent FIR filter of order 25.

To capture the observation of 1D variable whose underlying dynamic is 3D-Lorenz curve, the authors choose the architecture:

- 3 dimensional state variable.
- Dynamic function(f): 3 layered convelutional network.

Besides the two evaluation mentioned above, the original paper contains more. Interested readers can refer to [5] for more information.

3.1.2 Comments

- DFG(in this paper) is basically FG. "dynamic" only address the application aspects, rather than claiming an extension of FG.

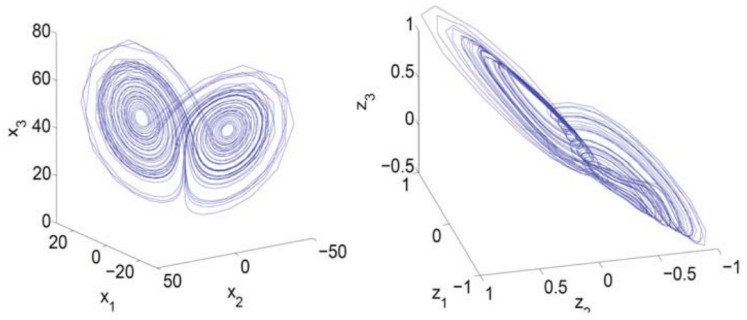

Figure 7: Training and Evaluation on Lorenz Curve

- The framework of FG is too general. Designing observation function and state transition function, choice of regularizer, setting coefficients are really tricky. There doesn't seem to be a rule of thumb.
- The introduction of unknown parameters and regularizers make the formulation deviate from ordinary factor graph. Before final marginalization, there is a training stage to obtain opimal paramters.

3.2 Solving Marginalization Problem in Social Network

This section follows the paper:
C. Wang, J. Tang, J. Sun, and J. Han. Dynamic social influence analysis through time-dependent factor graphs. *ASONAM*, v:p, 2011.

Declaration: figures in this section are borrowed from the orignal paper [7]. We omit further citation for simplicity.

3.2.1 Quick Note of the Main Idea

Problem settings:

- Graph $G^t =< V^t, E^t >$. V corresponds to people in a certain social network. E corresponds activity relationships. There is a real valued wegith function w defined on E.
- Graph G^t captures time varying relationships.
- Definition of w is problem(application) specific.
- Want to find the pairwise influence μ_{ij}. Denote the node to influence node i by y_i, a random variable . μ_{ij} is $P(y_i = j)$.

When defined on the static version, one μ_{ij} is output for each i, j. When defined on the dynamic version, a vectore $(\mu_{ij}^{(t)})$ is output.

Two definitions to facilitate further description:

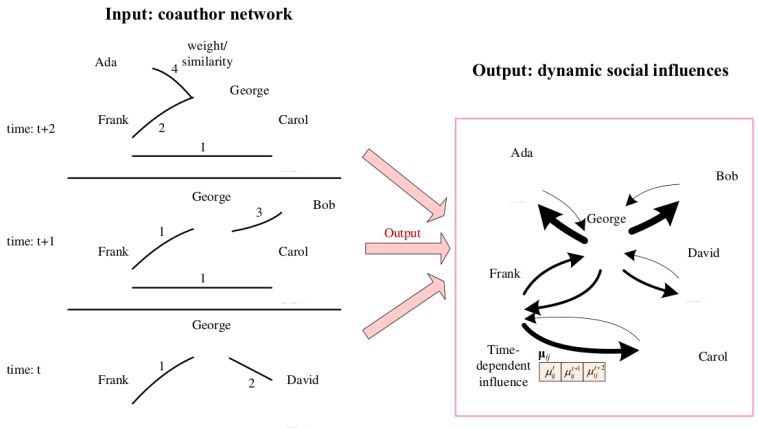

Figure 8: Problem Definition

- $NB(i)$: the neighbourhood of i. (More precisely, v_i, this is defined on the original graph)
- $SC(i) = NB(i) \cup i$.

More precisely, y_i should be linked to $v_j, \forall j \in NB(i)$ in the FG depicted in fig(). This can be justified through the definition of node factor function.

The definition of node factor function and link factor function, together with the observation of G removes complex links from y_i to V. The resultant FG can be viewed as one single layer with y_i vertices as variable nodes. Function nodes $g_i(y_i)$ are completely defined with the observation of G.

Note that there is one parameter u in the choice of link factor function. If not for this paramter, the FG in this paper was able to marginalize each y_i directly and give the influence using $\mu_{ij} = P(y_i = j)$. Again, the parameter estimation can be done using EM schema.

In order to capture the time varying property of social influence, the authors augmented original static FG to be time-dependent FG. Bridging function nodes are added between two time consecutive sub graphs. For detailed discussion, please refer to original paper.

Since the current section aims at modeling, details of algorithms are omitted. However, it's worth referring to the original paper for a specialized message passing algorithm named SIP(Social Influence Propagation) by the authors. That algorithm works more efficiently than general junction tree algorithms.

3.2.2 Comments

- Modeling an FG is rather easy. However, coming up with right choice of functions and paramters are difficult.

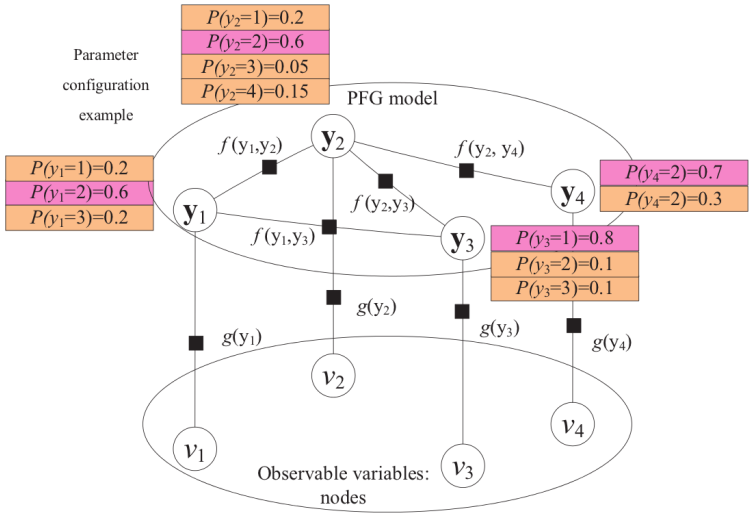

Figure 9: Visualization of the Factor Graph

$$g(y_i|V) = \begin{cases} \frac{w_{iy_i}}{\sum_{j\in NB(i)}(w_{ij}+w_{ji})} & y_i \neq i \\ \frac{\sum_{j\in NB(i)} w_{ji}}{\sum_{j\in NB(i)}(w_{ij}+w_{ji})} & y_i = i \end{cases}$$

Figure 10: Choice of Node Factor Function

- In this paper, the authors used heuristics to generate w_{ij}, which is a very important raw data. After the calculation of w_{ij}, the FG and so does the marginalizations are fully defined.
- The original itention(we've already shown that at the beginning of this article) of FG is to reduce complexity by utilizing graphical structures. However, the compuation in practice is still a concern. How to carefully design the algorithm to perform marginalization under the message passing framework according to physical meanings will be an interesting and chanllenging problem.

$$f(y_i, y_j|V) = \begin{cases} \frac{u}{|SC(i)\cap SC(j)|} & y_i = y_j \\ \frac{1-u}{|SC(i)||SC(j)|-|SC(i)\cap SC(j)|} & y_i \neq y_j \end{cases}$$

Figure 11: Choice of Link Factor Function

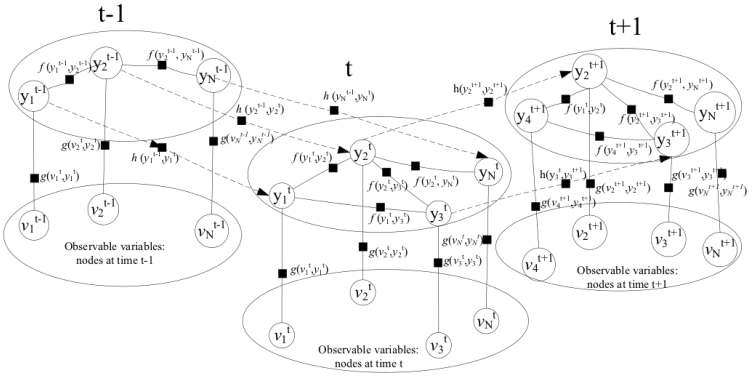

Figure 12: Factor Graph with Time Dependency

3.3 Short Summary on the Application of FG

- Framework and general algorithm is mature both in theory and in practice.
- Original FG can be used to solve marginalization problem directly. At this point, FG can be think of as a problem solver rather than a model. For this flavour, please refer to Bishop's book [1], Chapter 8. The author begins with BN and MRF which model directed and undirected independencies. In order to solve the inference problem, the author propose to convert BN and MRF to FG.
- According to different backgrounds, different authors come up with different Paramterized Factor Graph. Most of the works include two efforts:
  - Heuristics to determine paramters.
  - Iterative methods to determine paramters. (EM schema)

  Solving marginalization in FG is one sub problem then. Before giving the final marginalized answer, there is a training stage.
- Specific message passing algorithms are worth investigating. As is in [7], the authors propose SIP, specialized for their problem. The resulting algorithm is more tractable than general solvers.

4 Other Taste of Modeling

4.1 Coding

Besides modeling probability problems, people also use FG to model certain codes.

Consider a parity checking matrix given by H:

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{51}$$

In the corresponding factor graph, each factor node corresponds to one row and each variable node corresponds to one column. The type of the functions are indicators testing the predicate: $[x_{i_1} \oplus x_{i_2} \oplus x_{i_1} = 0]$. Note that the logic "$\cap$" operator can be regarded as "product". This kind of modeling is called behavioural modeling in [4].

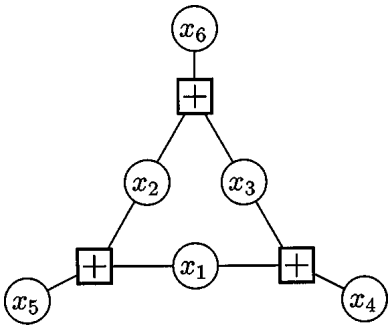

Figure 13: A Parity Code Modeled by FG[4]

4.2 Subgame Perfect Equilibrium

Interestingly, to search the subgame perfect equilibrium of an extensive game, we can use factor graph. For detailed course materials on extensive game, interested readers can refer to [6].

We know that to find the Nash Equilibrium of a game, one can enumerate the set of strategy profiles, and check for every player whether he can gain by deviation. This method is time-consuming, since it is no more than the definition of an equilibrium. However, subgame perfect equilibrium can be found much more efficiently with the help of One Deviation Property Lemma[6]. The lemma says, for every subgame of original game, we only need to check whether the first move is optimal or not. More intuitively, we can do backward deduction on the tree from leaves to root.

Note that in our introduction to FG, the example that we marginalize for a tree structure with a designated root fits this compuation very well.

5 Related Models/Algorithms

Models:

- Bayesian Network. Directed Graphical Model.

- Markov Random Field. Undirected Graphical Model.
- Chain Graph. Directed + Undirected.
- Factor Graph.

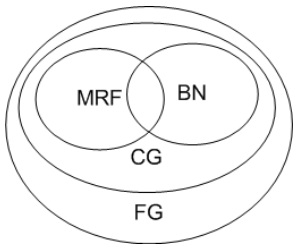

Figure 14: Venn Graph of Several Graphical Models

Justification of modeling ability:

- BN $\rightarrow$ MRF. The moralization process hides certain stucture previously available in BN.
- MRF. An example of square located 4 nodes. If two diagonal variables are observed, the other two variables become independent. BN can not express such information.
- CG. Since both directed and undirected, it forms the superset of the union of BN and MRF.
- FG. Converting CG to FG reveals more details, namely, how the joint distribution is factorized(usually, one explicit normalization function exhibits). BTW, FG can model not only probabilistic problems, but also other problems like system dynamics and coding. It is super general in the sense.

Algorithms:

- Junction tree.
- Bayesian ball.
- (Loopy)Belief propagation.
- Sum-product.

6 Discussions

Since the development of factor graph is boosted in the past decade, different authors come up with different description of similar problems. Not to distinguish right from wrong, I just regard those stuffs out there as inconsistent. My opinion on some parts of past literature:

- In Bishop's book[1], chapter 8.4.5, P411. The example is not good. Actually, when talking about that probability maximization problem, we should know "product" corresponds to product operator, and "sum" corresponds to max operator. In this case, the maginalization operation for a single varialbe is indeed the maximization for each instancde of that variable. Using local marginalized function(max), we can certainly get the global probability maximization point considering all variables.

- As for Dynamic Factor Graph, the author of this paper do not advocate the abuse of this term like an extension of factor graph. FG itself is able to model system dynamics, as we've already seen in those examples above. Other authors may use the term DFG [7] [5] , but their DFG is application specific. Those graphs are essentially FG. Not until we examine the physical meaning of some factor nodes do we realize their "dynamic" property.

Acknowledgements

References

[1] C.M Bishop. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[2] B.J. Frey. Extending factor graphs so as to unify directed and undirected graphical models. In *Proc. 19th Conf. Uncertainty in Artificial Intelligence*, pages 257–264, 2003.

[3] B.J. Frey, F.R. Kschischang, H.A. Loeliger, and N. Wiberg. Factor graphs and algorithms. In *PROCEEDINGS OF THE ANNUAL ALLERTON CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING*, volume 35, pages 666–680. UNIVERSITY OF ILLINOIS, 1997.

[4] F.R. Kschischang, B.J. Frey, and H.A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

[5] P. Mirowski and Y. LeCun. Dynamic factor graphs for time series modeling. *Machine Learning and Knowledge Discovery in Databases*, v:128–143, 2009.

[6] M.J. Osborne and A. Rubinstein. *A course in game theory*. The MIT press, 1994.

[7] C. Wang, J. Tang, J. Sun, and J. Han. Dynamic social influence analysis through time-dependent factor graphs. *ASONAM*, v:p, 2011.

[8] Moral Graph, Wikipedia, `http://en.wikipedia.org/wiki/Moral_graph`

[9] Markov Blanket, Wikipedia, `http://en.wikipedia.org/wiki/Markov_blanket`