10-708: Probabilistic Graphical Models 10-708, Spring 2016

12 : Variational Inference I

Lecturer: Eric P. Xing *Scribes: Jing Chen, Yulan Huang, Yu-Fang Chang*

1 Kalman Filtering

From last lecture, we have introduced Kalman Filtering as a recursive procedure to update the belief state. In each iteration, there are two steps: **Predict Step** and **Update Step**. In Predict Step, we compute latent state distribution $P(\mathbf{X}_{t+1}|\mathbf{y}_{1:t})$ from prior belief $P(\mathbf{X}_t|\mathbf{y}_{1:t})$ and dynamic model $p(\mathbf{X}_{t+1}|\mathbf{X}_t)$. This step is also called time update. In Update Step, we compute new belief of the latent state distribution $p(\mathbf{X}_{t+1}|\mathbf{y}_{1:t+1})$ from prediction $p(\mathbf{X}_{t+1}|\mathbf{y}_{1:t})$ and observation $\mathbf{y}_{t+1}$ by using the observation model $p(\mathbf{y}_{t+1}|\mathbf{X}_{t+1})$. The step is also called measurement update since its using the measured information y_{t+1}. The reason for doing so is that under a joint multivariant gaussian distribution, we can compute the conditional influences, marginal influences easily. Since all distributions are gaussian, their linear combinations are also gaussian. Hence we just need the mean and covariance, which can be computed easily in this case, to describe the influence.

1.1 Derivation

Our goal is to compute $p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t+1})$ We first utilize the dynamic model for finding the parameters of the distribution of $p(\mathbf{X}_{t+1}|\mathbf{y}_{1:t})$. With the dynamic model, we define $\mathbf{x}_{t+1}$:

$$\mathbf{x}_{t+1} = Ax_t + Gw_t$$

where W is the noise model with zero mean and covariance matrix Q. We can then predict the expectation $\hat{x}_{t+1|t}$ and covariance $P_{t+1|t}$ of the distribution $p(\mathbf{X}_{t+1}|\mathbf{y}_{1:t})$ as following.

$$\begin{aligned}
\hat{x}_{t+1|t} &= \mathbb{E}[x_{t+1}|y_{1:t}] = \mathbb{E}[Ax_t + Gw_{t+1}|y_{1:t}] = A\hat{x}_{t|t} + \mathbf{0} = A\hat{x}_{t|t} \\
P_{t+1|t} &= \mathbb{E}[(x_{t+1} - \hat{x}_{t+1|t})(x_{t+1} - \hat{x}_{t+1|t})^T|y_{1:t}] \\
&= \mathbb{E}[(Ax_t + Gw_t - \hat{x}_{t+1|t})(Ax_t + Gw_t - \hat{x}_{t+1|t})^T|y_{1:t}] \\
&= \mathbb{E}[(Ax_t + Gw_t - A\hat{x}_{t|t})(Ax_t + Gw_t - A\hat{x}_{t|t})^T|y_{1:t}] \\
&= \mathbb{E}[(Ax_t - A\hat{x}_{t|t})(Ax_t - A\hat{x}_{t|t})^T + (Gw_t(Ax_t - A\hat{x}_{t|t})^T + Ax_t - A\hat{x}_{t|t})Gw_t^T + Gw_tGw_t^T] \\
&= AP_{t|t}A^T + 0 + 0 + GQG^T \\
&= AP_{t|t}A^T + GQG^T
\end{aligned}$$

For observation model, we have

$$y_t = Cx_t + v_t$$

$$v_t \sim \mathcal{N}(0; R)$$

We can then derive the mean and variance of observation and state variables.

$$\begin{aligned}
\mathbb{E}[Y_{t+1}|y_{1:t}] &= \mathbb{E}[Cx_t + w_t] = C\hat{x}_{t+1|t} \\
Var(y_{t+1}|y_{1:t}) &= \mathbb{E}[(Y_{t+1} - \hat{y}_{t+1|t})(Y_{t+1} - \hat{y}_{t+1|t})^T|y_{1:t}] \\
&= \mathbb{E}[(CX_{t+1} + v_t - C\hat{x}_{t+1|t})(CX_{t+1} + v_t - C\hat{x}_{t+1|t})^T|y_{1:t}] \\
&= CP_{t+1|t}C^T + R \\
Cov(y_{t+1}, x_{t+1}|y_{1:t}) &= \mathbb{E}[(Y_{t+1} - \hat{y}_{t+1|t})(X_{t+1} - \hat{x}_{t+1|t})^T|y_{1:t}] \\
&= \mathbb{E}[(CX_{t+1} + v_t - C\hat{x}_{t+1|t})(X_{t+1} + \hat{x}_{t+1|t})^T|y_{1:t}] \\
&= CP_{t+1|T} \\
Cov(x_{t+1}, y_{t+1}|y_{1:t}) &= Cov(y_{t+1}, x_{t+1}|y_{1:t})^T = P_{t+1|t}C^T
\end{aligned}$$

Next, we can combine the above result and get the joint distribution $p(X_{t+1}, Y_{t+1}|y_{1:t}) \sim \mathcal{N}(m_{t+1}, V_{t+1})$ with

$$m_{t+1} = \begin{bmatrix} \hat{x}_{t+1|t} \\ C\hat{x}_{t+1|t} \end{bmatrix}, \quad V_{t+1} = \begin{bmatrix} P_{t+1|t} & P_{t+1|t}C^T \\ CP_{t+1|t} & CP_{t+1|t}C^T + R \end{bmatrix}$$

We can see that $Var(y_{t+1}|y_{1:t})$ is similar to $P_{t+1|t}$, so we introduce Kalman gain matrix $\mathbf{K}$ to replace some repeated step as following:

$$\begin{aligned}
K &= Cov(x_{t+1}, y_{t+1}|y_{1:t})Var(y_{t+1}|y_{1:t})^{-1} \\
&= P_{t+1|t}C^T(CP_{t+1|t}C^T + R)^{-1}
\end{aligned}$$

Since K doesn't require a new observation, in other words, independent of the data, it can be precomputed. For the measurements update, by the formula for conditional Faussian distribution, we have

$$\begin{aligned}
\hat{x}_{t+1|t+1} &= \hat{x}_{t+1|t} + Cov(x_{t+1}, y_{t+1}|y_{1:t})Var(y_{t+1}|y_{1:t})^{-1}(y_{t+1} - \hat{y}_{t+1|t}) \\
&= \hat{x}_{t+1|t} + K(y_{t+1} - C\hat{x}_{t+1|t}) \\
P_{t+1|t+1} &= P_{t+1|t} - KCP_{t+1|t}
\end{aligned}$$

and we finally done with the derivation.

1.2 Example

Now we look at a simple example of the Kaulman Filter. We consider noisy observations of a 1D particle moving randomly.

$$\begin{aligned}
x_{t|t-1} &= x_{t-1} + w, \quad w \sim \mathcal{N}(0, \sigma_x) \\
z_t &= x_t + v, \quad v \sim \mathcal{N}(0, \sigma_z)
\end{aligned}$$

and then we can plugged into the Kalman Filter equation that gives us the new mean and variance.

$$\begin{aligned}
P_{t+1|t} &= AP_{t|t}A + GQG^T = \sigma_t + \sigma_x \\
\hat{x}_{t+1|t} &= A\hat{x}_{t|t} = \hat{x}_{t|t} \\
K &= P_{t+1|t}C^T(CP_{t+1|t}C^T + R)^{-1} = (\sigma_t + \sigma_x)(\sigma_t + \sigma_x + \sigma_z) \\
\hat{x}_{t+1|t+1} &= \hat{x}_{t+1|t} + K_{t+1}(z_{t+1} - C\hat{x}_{t+1|t}) = \frac{(\sigma_t + \sigma_x)z_t + \sigma_z\hat{x}_{t|t}}{\sigma_t + \sigma_x + \sigma_z} \\
P_{t+1|t+1} &= P_{t+1|t} - KCP_{t+1|t} = \frac{(\sigma_t + \sigma_x)\sigma_z}{\sigma_t + \sigma_x + \sigma_z}
\end{aligned}$$

We can see that, with the initial setting, the noise is not going to generate a non-zero shift on the mean since the noise is centered. But the variance is changing. Let's begin with the $P(x_0)$ and base on the transition model, we can now predict the distribution of the next timestamp. It turns out a distribution with wider gaussian variance. This is because that the point move due to the noise, which increase the uncertainty of the point. Once we have a new point z_{t+1}, the mean is shifted as the previous one add the transformation one. The noise is reduced due to our choice which reduce the uncertainty. And now it induce to a new belief of the distribution of the point. Then we come to the intuition that once we have an observation, we can update the mean as following:

$$\hat{x}_{t+1|t+1} = \hat{x}_{t+1|t} + K_{t+1}(z_{t+1} - C\hat{x}_{t+1|t}) = \frac{(\sigma_t + \sigma_x)z_t + \sigma_z \hat{x}_{t|t}}{\sigma_t + \sigma_x + \sigma_z}$$

which the term $(z_{t+1} - C\hat{x}_{t+1|t})$ is called the innovation.

2 Background Review

2.1 Inference Problem

In this lecture, we look deeply into variational inference in graphical models. Given a graphical model G and corresponding distribution P defined over a set of variables (nodes) V, the general inference problem involves the computation of:

- the likelihood of observed data
- the marginal distribution $p(x_A)$ over a particular subset of nodes $A \subset V$
- the conditional distribution $p(x_A|x_B)$ for disjoint subsets A and B
- a mode of the density $\hat{x} = arg\max_{x \in \chi^m} p(x)$

Previous lectures have covered several exact inference algorithms like brute force enumeration, variable elimination, sum-product and junction tree algorithms. More specifically, while brute force enumeration sum over all variable excluding the query nodes, variable elimination fully utilize the graphical structure by taking ordered summation over variables. Both algorithm treat individual computations independently, which is to some degree wasteful. On the contrary, message passing approaches such as sum-product and belief propagation are considered more efficient as a result of sharing intermediate terms.

However, the above approaches typically apply to trees and can hardly converge on non-tree structures, in other words, loopy graphs. One exception is the Junction tree algorithm, which provides a way to convert any arbitrary loopy graph to a clique trees and then perform exact inference on the clique trees by passing messages. Nevertheless few people actually use this approach because though junction is locally and globally consistent, it is to expensive with computational cost exponential to the maximum number of nodes in the each clique.

In order to solve the inference problem on loopy graphs, we introduce approximate inference approaches, and focus on loopy belief propagation in this lecture.

2.2 Review of Belief Propagation

We first give a brief review for the Belief Propagation (BP), one classical message passing algorithm. Figure 1a demonstrates the procedure of massage update rules in BP, where node i passes a message to each of its

neighboring nodes j once receiving messages from its neighbors excluding j. The message update rule can be formulated as below:

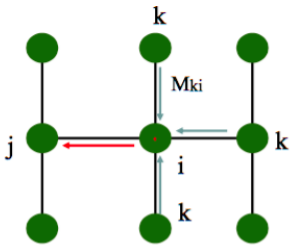
$$m_{i\to j}(x_j) \propto \sum_{x_i} \psi_{ij}(x_i, x_j)\psi_i(x_i) \prod_{k\in N(i)\setminus j} m_{j\to i}(x_i)$$

where $\psi_{ij}(x_i, x_j)$ is called *Compatibilities (interactions)* and $\psi_i(x_i)$ is called *external evidence*.

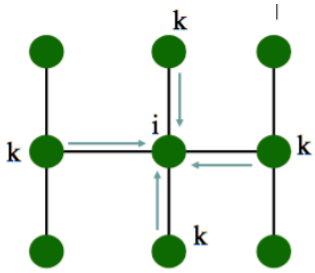
Further, the marginal probability of each node in graph can then be computed in a procedure as Figure **??** shows, which can be formulated as:

$$b_i(x_i) \propto \psi_i(x_i) \prod_{k\in N(i)} m_k(x_k)$$

Particularly, it is worth noticing that BP on trees always converges to exact marginals as Junction Tree algorithm reveals.



(a) BP message passing procedure



(b) BP node marginal

Figure 1: BP Message-update Rules

Moreover, we can generalize BP model on factor graph, where the square nodes denote factors and circle nodes denote variables as Figure 2b. Similarly, we can compute the marginal probabilities for variable i and factor a as Figure **??** shows:

$$b_i(x_i) \propto f_i(x_i) \prod_{a\in N(i)} m_{a\to i}(x_i)$$

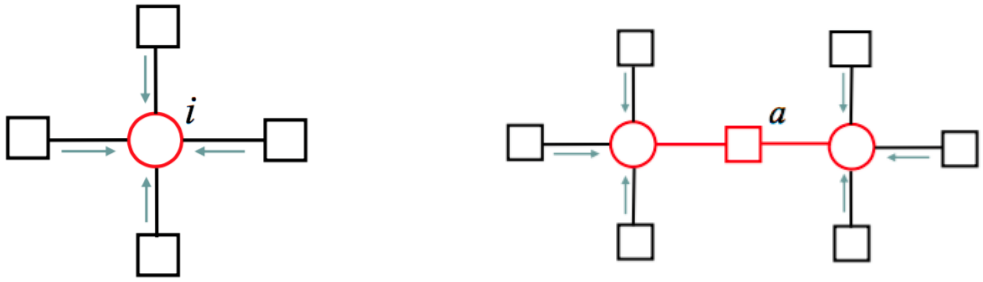
$$b_a(X_a) \propto f_a(x_a) \prod_{i\in N(a)} m_{i\to a}(x_i)$$

where we call $b_i(x_i)$ "beliefs" and $m_{a\to i}$ "messages".

Therefore, messages are passed in two ways: (1) from variable i to factor a; (2) from factor a to variable i, which is written as:

$$m_{i\to a}(x_i) = \prod_{c\in N(i)\setminus a} m_{c\to i}(x_i)$$

$$m_{a\to i}(x_i) = \sum_{X_a\setminus x_i} f_a(X_a) \prod_{j\in N(a)\setminus i} m_{j\to a}(x_j)$$



(a) (b)

Figure 2: BP on Factor Graph

3 Loopy Belief Propagation

Now we consider the inference on an arbitrary graph. As is mentioned, most previously discussed algorithms apply on tree-structured graphs and even some can converge on arbitrary graphs, say, loopy graphs, they suffer from the problem of expensive computational cost. Also, we can hardly prove the correctness of these algorithms, e.g. Junction Tree algorithm.

Therefore, the Loopy Belief Propagation (LBP) algorithm is proposed. LBP can be viewed as a fixed-point iterative procedure that tries to minimize F_{bethe}. More specifically, LBP starts with random initialization of messages and belief, and iterate the following steps until convergence.

$$b_i(x_i) \propto \prod_{a \in N(i)} m_{a \to i}(x_i)$$

$$b_a(X_a) \propto f_a(X_a) \prod_{i \in N(a)} m_{i \to a}(x_i)$$

$$m^{new}_{i \to a}(x_i) = \prod_{a \in N(i) \setminus a} m_{c \to i}(x_i)$$

$$m^{new}_{a \to i}(x_i) = \sum_{X_a \setminus x_i} f_a(X_a) \prod_{j \in N(a) \setminus i} m_{j \to a}(x_j)$$

However, it is not clear whether such procedure will converge to a correct solution. In late 90's, there was much research devotion trying to investigate the theory lying behind this algorithm. Murphy et. al (1999) has revealed empirically that a good approximation is still achievable if:

- stop after fixed number of iterations
- stop when no significant change in beliefs
- of solution is not oscillatory but converges, it usually is a good approximation

However, whether the good performance of LBP is a dirty hack ? In the following sections, we try to understand the theoretical characteristics of this algorithm and show that LBP can lead to an almost optimal approximation to the actual distribution.

3.1 Approximating a Probability Distribution

Let us denote the actual distribution of an arbitrary graph G as P:

$$P(X) = \frac{1}{Z} \prod_{f_a \in F} f_a(X_a)$$

Then we wish to find a distribution Q such that Q is a "good" approximation to P. To achieve this, we first recall the definition of KL-divergence:

$$KL(Q_1||Q_2) = \sum_X Q_1(X) \log(\frac{Q_1(X)}{Q_2(X)})$$

satisfying:

$$KL(Q_1||Q_2) \geq 0$$

$$KL(Q_1||Q_2) = 0 \iff Q_1 = Q_2$$

$$KL(Q_1||Q_2) \neq KL(Q_2||Q_1)$$

Therefore, our goal of finding an optimal approximation can be converted to minimizing the KL-divergence between P and Q. However, the computation of $KL(P||Q)$ requires inference steps on P while $KL(Q||P)$ not. Thus we adopt $KL(Q||P)$:

$$\begin{aligned} KL(Q||P) &= \sum_X Q(X) \log(\frac{Q(X)}{P(X)}) \\ &= \sum_X Q(X) \log Q(X) - \sum_X Q(X) \log P(X) \\ &= -H_Q(X) - E_Q \log(\frac{1}{Z} \prod_{f_a \in F} f_a(X_a)) \\ &= -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a) + \log Z \end{aligned}$$

Therefore we can formulate the optimization function as:

$$KL(Q||P) = -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a) + \log Z$$

where H_Q denotes the entropy of distribution Q, and $F(P, Q)$ is called *"free energy"*:

$$F(P, Q) = -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a)$$

Particularly, we have $F(P, P) = -\log Z$ and $F(P, Q) \geq F(P, P)$. More specifically, we see that while $\sum_{f_a \in F} E_Q \log f_a(X_a)$ can be computed based on marginals over each f_a, $H_Q = -\sum_X Q(X) \log Q(X)$ is hard since it requires summation over all possible values of X. This makes the computation of F hard. One possible solution is to approximate $F(P, Q)$ with some $\hat{F}(P, Q)$ that is easy to compute.

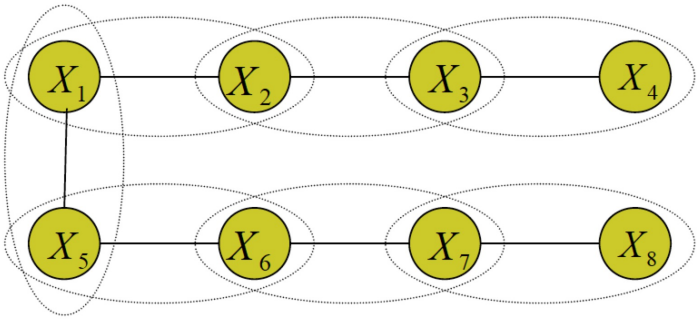


Figure 3: A tree graph

3.2 Deriving the Bethe Approximation

As shown in Figure 3, for a tree-structured graph, the joint probability can be written as $b(x) = \prod_a b_a(x_a) \prod_i b_i(x_i)^{1-d_i}$, where a enumerates all edges in the graph, i enumerates all nodes in the graph and d_i are the degree of the vertex i. We can then calculate their entropy as well as the free energy:

$$H_{tree} = -\sum_a \sum_{x_a} b_a(x_a) ln b_a(x_a) + \sum_i (d_i - 1) \sum_{x_i} b_i(x_i) ln b_i(x_i)$$

$$F_{Tree} = \sum_a \sum_{x_a} b_a(x_a) ln \frac{b_a(x_a)}{f_a(x_a)} + \sum_i (1 - d_i) \sum_{x_i} b_i(x_i) ln b_i(x_i)$$

Since the entropy and free energy only involves summation over edges and vertices, it is easy to compute.

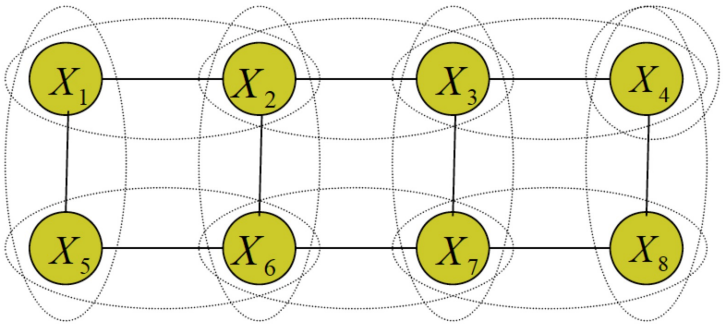


Figure 4: An arbitrary graph

However, for an arbitrary graph as Figure 4, the entropy and free energy is hard to write down. Therefore, we use the Bethe approximation which has the exact same formula as free energy for a tree-structured graph:

$$H_{Bethe} = -\sum_a \sum_{x_a} b_a(x_a) ln b_a(x_a) + \sum_i (d_i - 1) \sum_{x_i} b_i(x_i) ln b_i(x_i)$$

$$F_{Bethe} = \sum_a \sum_{x_a} b_a(x_a) ln \frac{b_a(x_a)}{f_a(x_a)} + \sum_i (1 - d_i) \sum_{x_i} b_i(x_i) ln b_i(x_i)$$

As we can see, the advantage of Bethe approximation is that it is easy to compute, since the entropy term only involves sum over pairwise and single variables. However, the approximation $\hat{F}(P, Q)$ may or may not

be well connect to the true $F(P,Q)$. There is no guarantee that it will be greater, equal or less than the true $F(P,Q)$.

To find the belief $b(x_a)$ and $b(x_i)$ that minimize the KL divergence, while still satisfying the local consistency. For the discrete case, the local consistency is:

$$\forall i, x_i \sum_{x_i} b_i(x_i) = 1$$
$$\forall a, i \in N(a), x_i \sum_{x_a | x_a = x_i, x_j} b_a(x_a) = b_i(x_i)$$

where N(a) is all neighbors of x_i. Thus, using the Lagrangian multiplier, the objective function becomes:

$$L = F_{Bethe} + \sum_i \gamma_i [1 - \sum_{x_i} b_i(x_i)] + \sum_a \sum_{i \in N(a)} \sum_{x_i} \lambda_{ai}(x_i)[b_i(x_i) - \sum_{X_a \setminus x_i} b_a(X_a)]$$

We can find the stationary points by setting the derivative to zero:

$$\frac{\partial L}{\partial b_i(x_i)} = 0$$
$$\Rightarrow b_i(x_i) \propto exp(\frac{1}{d_i - 1} \sum_{a \in N(i)} \lambda_{ai}(x_i))$$
$$\frac{\partial L}{\partial b_a(x_a)} = 0$$
$$\Rightarrow b_a(x_a) \propto exp(-log f_a(X_a) + \sum_{i \in N(a)} \lambda_{ai}(x_i))$$

3.3 Bethe Energy Minimization using Belief Propagation

By setting the derivative of the objective function, in the previous section we obtain the update formula for $b_i(x_i)$ and $b_a(X_a)$ in a factor graph. For the variables in the factor graph, with $\lambda_{ai}(x_i) = log(m_{i \to a}(x_i)) = log \prod_{b \in N(i) \neq a} m_{b \to i}(x_i)$, we have:

$$b_i(x_i) \propto f_i(x_i) \prod_{a \in N(i)} m_{a \to i}(x_i)$$

For the factors, we have:

$$b_a(X_a) \propto f_a(X_a) \prod_{i \in N(a)} \prod_{c \in N(i) \setminus a} m_{c \to i}(x_i)$$

Using $b_{a \to i}(x_i) = \sum_{X_a \setminus x_i} b_a(X_a)$, we get

$$m_{a \to i}(x_i) = \sum_{X_a \setminus x_i} f_a(X_a) \prod_{j \in N(a) \setminus i} \prod_{b \in N(j) \setminus a} m_{b \to j}(x_j)$$

As we can see, the Belief Propagation-update is in a sum product form and is easy to compute.

4 Theory Behind Loopy Belief Propagation

For a distribution $p(X|\theta)$ associated with a complex graph, computing the marginal (or conditional) probability of arbitrary random variable(s) is intractable. Thus, instead the variational methods optimize over an

easier (tractable) distribution q. It want to find

$$q^* = argmin_{q \in S}\{F_{Betha}(p, q)\}$$

Now, we can just optimize H_q. However, optimizing H_q is still difficult, so instead we do not optimize the q(X) explicitly, but relax the optimization problem to the approximate objective, the Bethe free energy of the beliefs $F(b)$, and a loose feasible set with local constrains. The Loopy belief propagation can be viewed as a fixed point iteration procedure that want to optimize F(b). Loopy belief propagation often not converge to the correct solution, although empirically it often performs well.

5 Generalized Belief Propagation

Instead of considering single node in normal loopy belief propagation, the Generalized Belief Propagation considers regions of graph. This enables us to use more accurate H_q for approximation and achieve better results. In the generalized belief propagation, instead of using Bethe free energy, it uses Gibbs free energy which is more generalized and achieves better results in the cost of an increased computational complexity.

More precisely, the Generalized Belief Propagation defines the belief in a region as the product of the local information (factors in region), messages from parent regions and messages into descendant regions from parents who are not descendants. Moreover, the message-update roles are obtained by enforcing marginalization constrains.

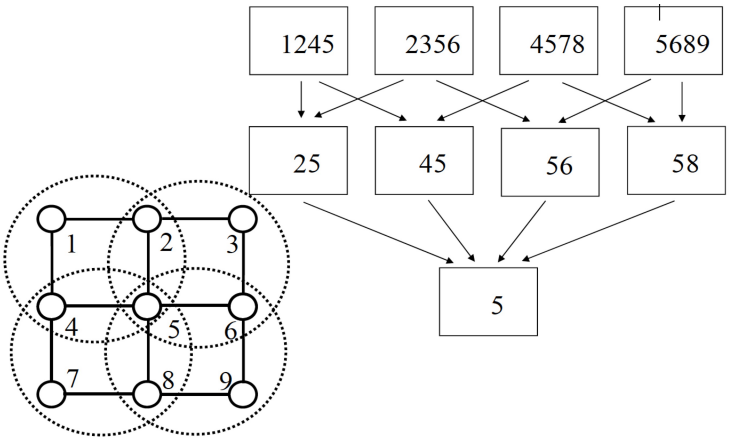


Figure 5: Example: Generalized Belief Propagation

There is an example for this region integration: As shown in Figure 5, the graph contains four regions, each of which contains four nodes.

When we compute beliefs, we integrate them hierarchically, which increases the computational cost. As shown in Figure 6 and Figure **??**, when we want to compute belief for a particular region, we consider nodes that share regions and calculate their energies from their parent regions. As we can see, more regions can make the approximation more accurate, but also increases the computational cost.

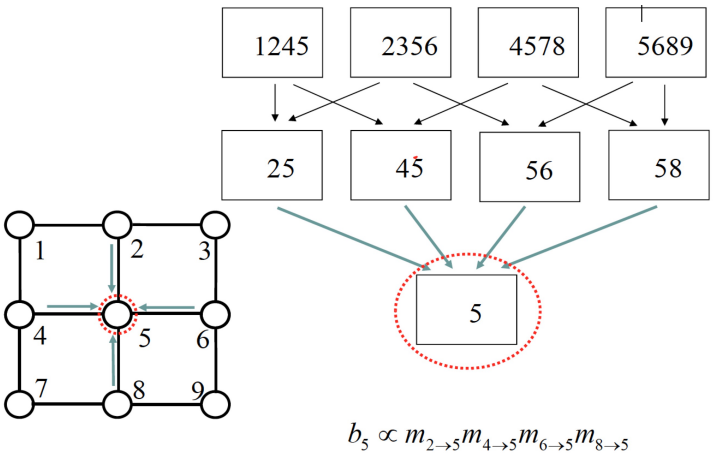


Figure 6: Example: Hierarchically compute belief in Generalized Belief Propagation