优化的层次

—意见领袖算法详解

alexcheng

优化的层次

- **算法层次**
  - 时间和空间的优化
  - 并发和锁的优化
  - 数据结构的设计
- **系统层次**
  - 系统负载均衡
  - 充分利用硬件性能
  - 减少额外开销
- **代码层次**
  - Cache
  - 执行顺序
  - 语言优化

意见领袖算法详解

- **算法原理**

  sum-product

  max-product

  Affinity Propagation

  Topical Affinity Propagation

- **工程实现**

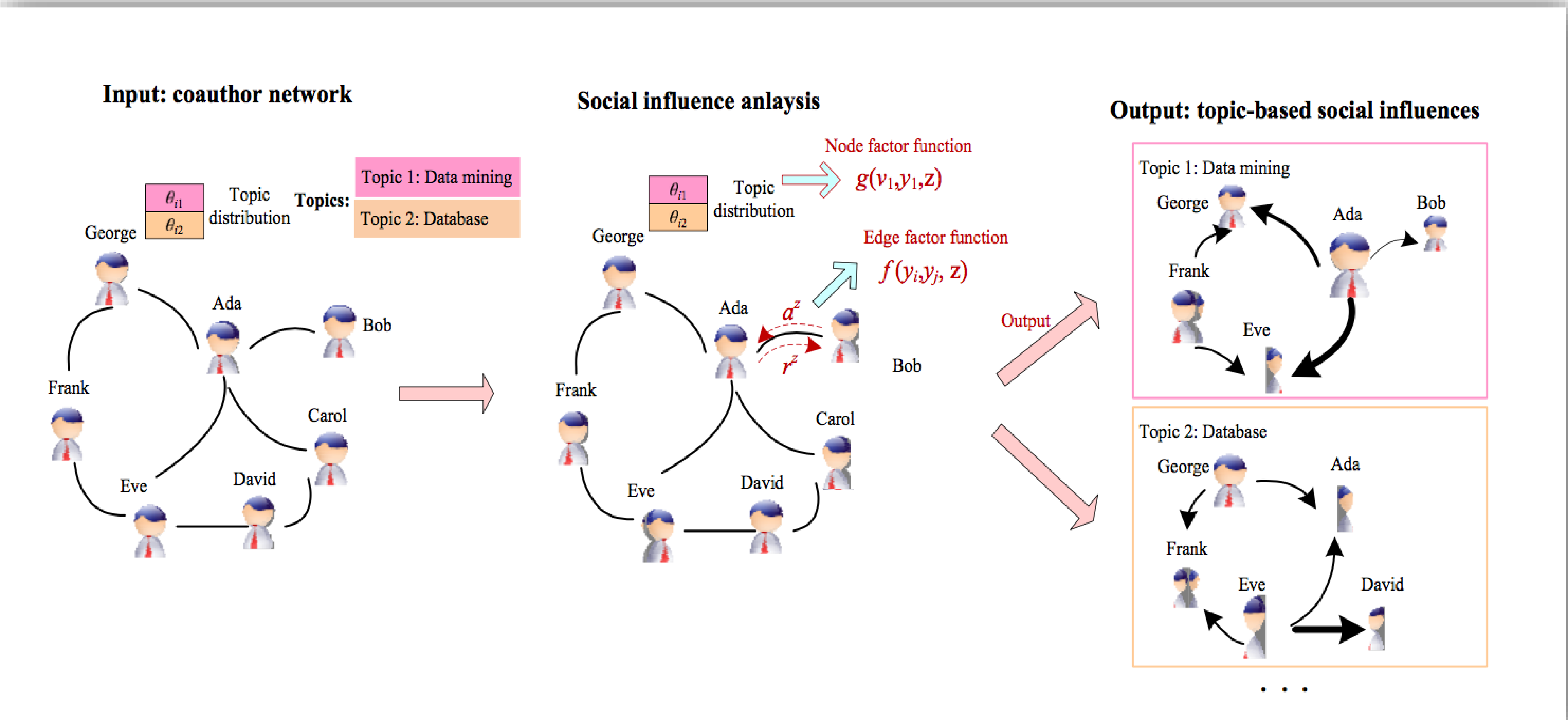
  Graphx介绍

  调参

  负载均衡

  Cache

算法原理-定义问题

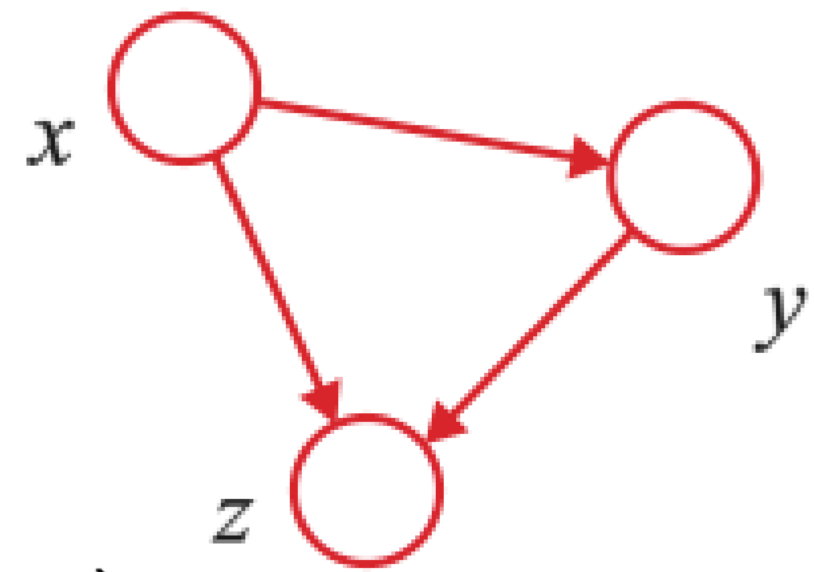


Input: coauthor network
θ_{i1}
θ_{i2}
Topic distribution
Topics:
Topic 1: Data mining
Topic 2: Database
George
Ada
Bob
Frank
Carol
Eve
David
Social influence anlaysis
Node factor function
$g(v_1,y_1,z)$
Topic distribution
Edge factor function
$f(y_i,y_j,z)$
a^z
r^z
Output
Output: topic-based social influences
Topic 1: Data mining
Topic 2: Database
. . .

算法原理-概率图

- 算出每种组合下的概率，取最大的作为解

$$p(x, y, z)$$

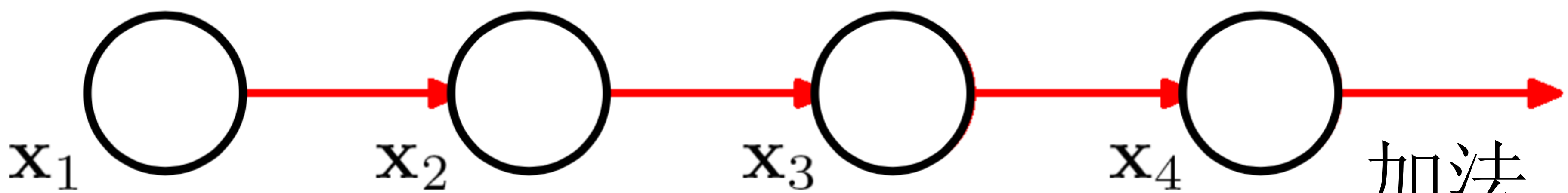


- 利用条件概率降低计算量

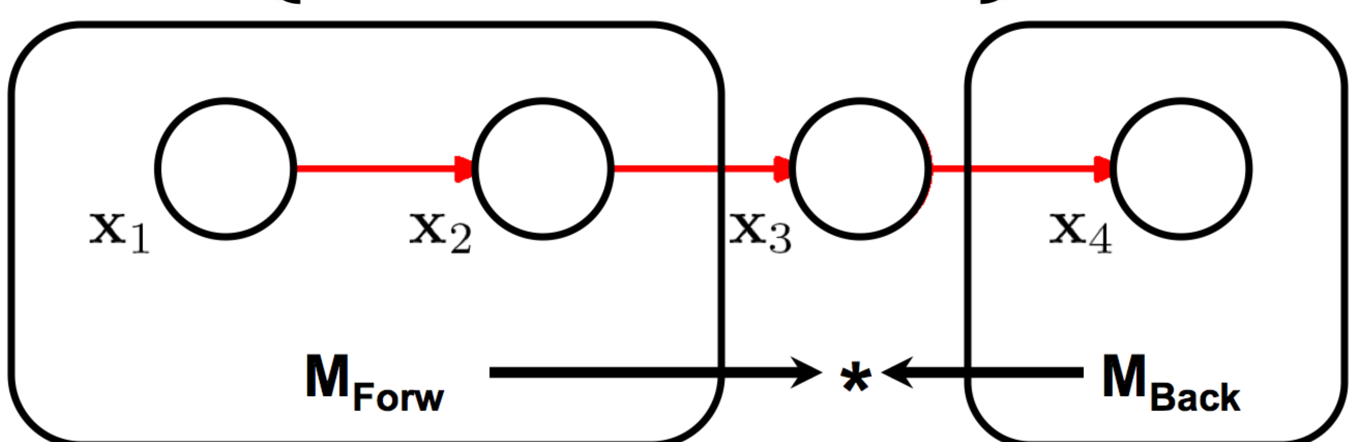
$$\begin{aligned} p(x, y, z) &= p(x)p(y, z|x) \\ &= p(x)p(y|x)p(z|x, y) \end{aligned}$$

$$p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i|\mathrm{pa}_i)$$

算法原理-消息传播



	加法	乘法
$P(x3) = \sum_{x1}\sum_{x2}\sum_{x4} P(x1)\, P(x2\|x1)\, P(x3\|x2)\, P(x4\|x3)$	7	7*3
$= \sum_{x1}\sum_{x2} P(x1)\, P(x2\|x1)\, P(x3\|x2) \left(\sum_{x4} P(x4\|x3)\right)$		
$= \left(\sum_{x2}\left(\sum_{x1} P(x1)\, P(x2\|x1)\right) P(x3\|x2)\right)\left(\sum_{x4} P(x4\|x3)\right)$	3+1	4+2



$= [.458 \;\; .542]$

$= \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$p(x_i) = \frac{1}{Z} m_\alpha(x_i) m_\beta(x_i)$$

$= [\,(.458)(1) \quad (.542)(1)\,]$

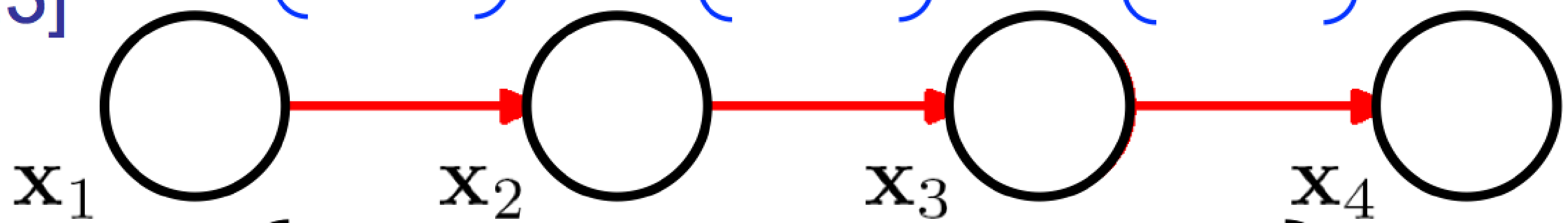
$= [.458 \;\; .542]$

$= [.458 \;\; .542]$ (after normalizing, but note that it was already normalized. Again, not a coincidence)

算法原理-sum product算法

$P(x1)$ $[.7 \quad .3]$

$P(x2|x1)$ $\begin{bmatrix} .4 & .6 \\ .5 & .5 \end{bmatrix}$ $P(x3|x2)$ $\begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix}$ $P(x4|x3)$ $\begin{bmatrix} .5 & .5 \\ .7 & .3 \end{bmatrix}$



$$P(x3) = \left(\sum_{x2} \left[\sum_{x1} P(x1)\ P(x2|x1) \right] P(x3|x2) \right) \left[\sum_{x4} P(x4|x3) \right]$$

$$\sum_{x1} \quad \overset{.7}{P(x1=1)}\overset{.4}{P(x2=1|x1=1)} \qquad \overset{.7}{P(x1=1)}\overset{.6}{P(x2=2|x1=1)}$$

$$\overset{.3}{P(x1=2)}\overset{.5}{P(x2=1|x1=2)} \qquad \overset{.3}{P(x1=2)}\overset{.5}{P(x2=2|x1=2)}$$

$$(.7)(.4) + (.3)(.5) \qquad (.7)(.6) + (.3)(.5)$$

$$0.43 \qquad 0.57$$

$$\begin{bmatrix} .5 & .5 \\ .7 & .3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

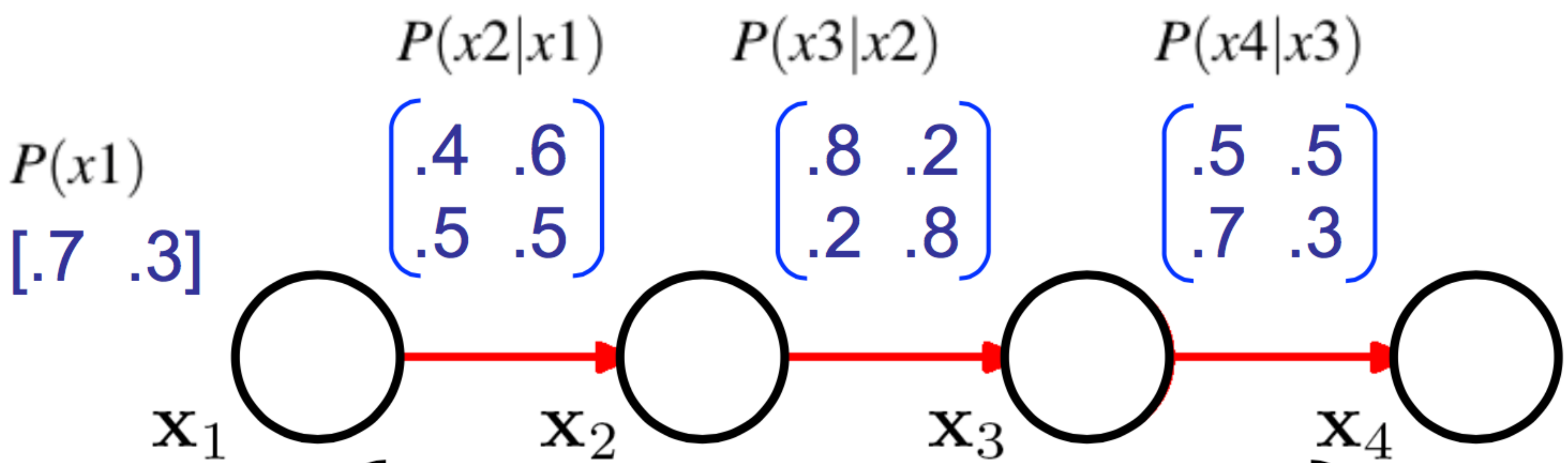
note: this is not a coincidence

$$[.7 \quad .3] \begin{bmatrix} .4 & .6 \\ .5 & .5 \end{bmatrix} \begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix}$$

$$= [.43 \quad .57] \begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix}$$

$$= [.458 \quad .542]$$

算法原理-max product算法



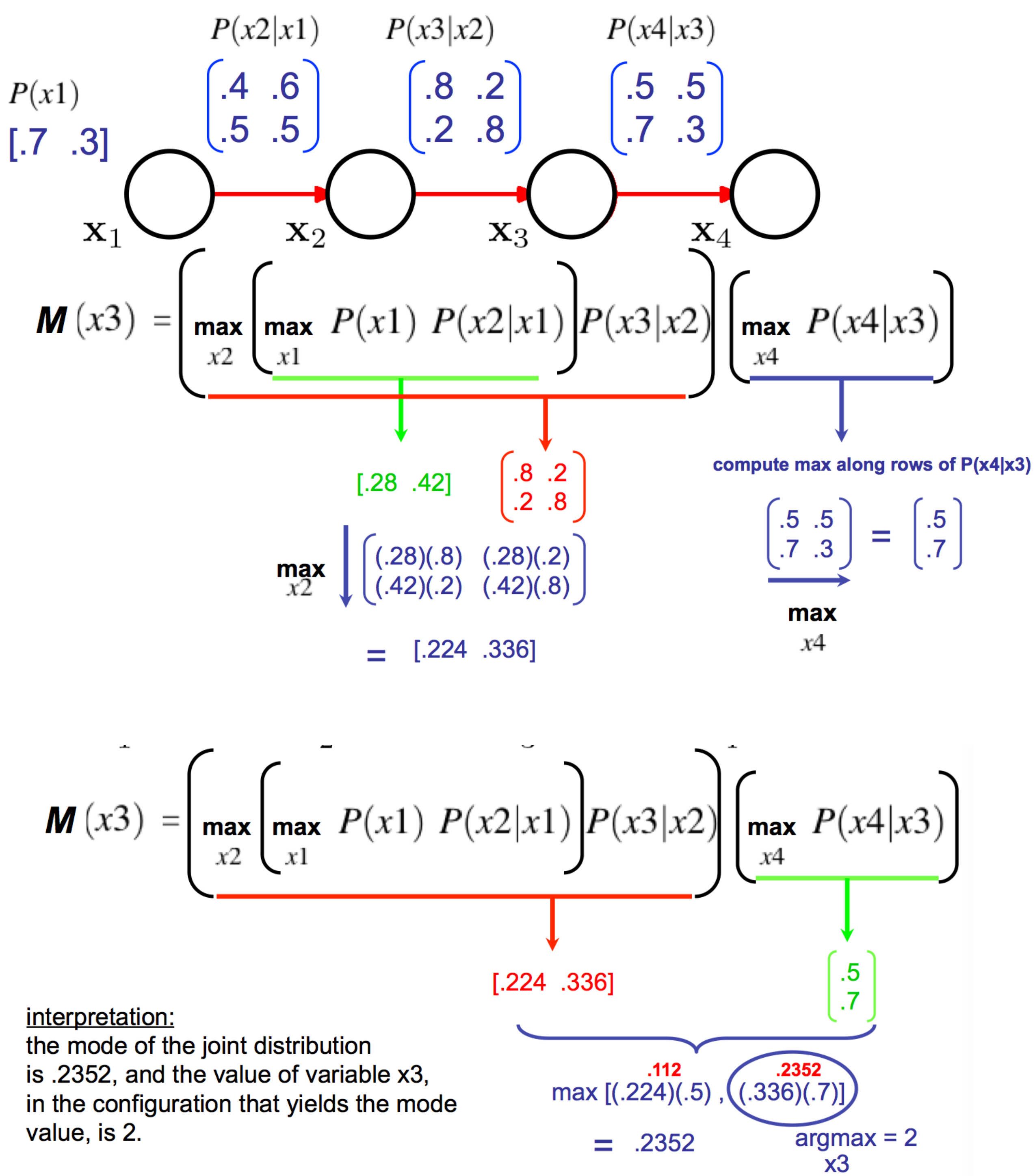
$$\boldsymbol{M}(x3) = \left[\max_{x2}\left[\max_{\underline{x1}} \underbrace{P(x1)\, P(x2|x1)}\right] P(x3|x2)\right]\left[\max_{x4} P(x4|x3)\right]$$

max marginal

$\max_{x1}$

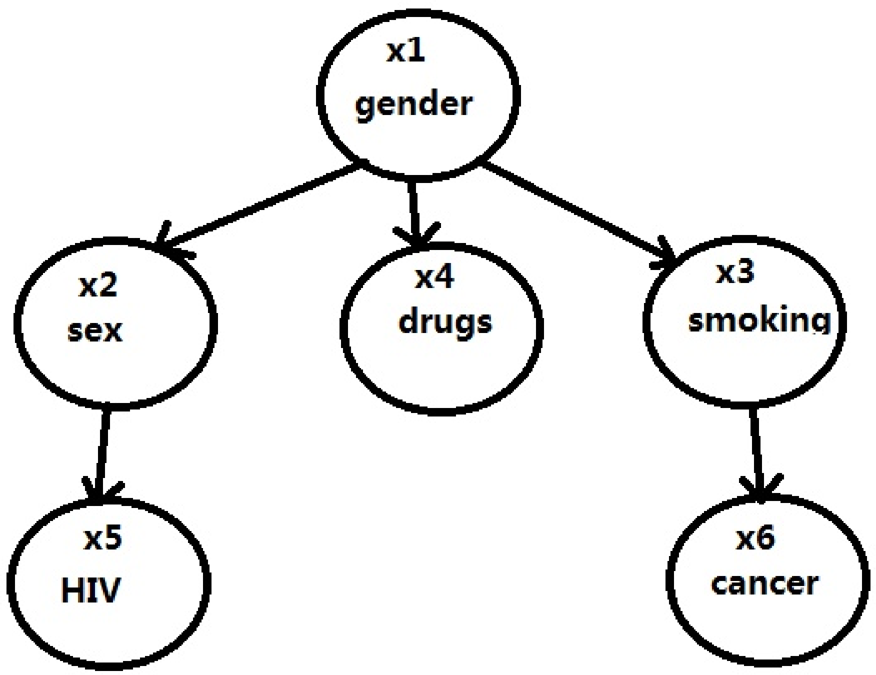
$\overset{.7}{P(x1=1)}\overset{.4}{P(x2=1\|x1=1)}$	$\overset{.7}{P(x1=1)}\overset{.6}{P(x2=2\|x1=1)}$
$\overset{.3}{P(x1=2)}\overset{.5}{P(x2=1\|x1=2)}$	$\overset{.3}{P(x1=2)}\overset{.5}{P(x2=2\|x1=2)}$
max[(.7)(.4) , (.3)(.5)]	max[(.7)(.6) , (.3)(.5)]
0.28	0.42

算法原理-max product算法



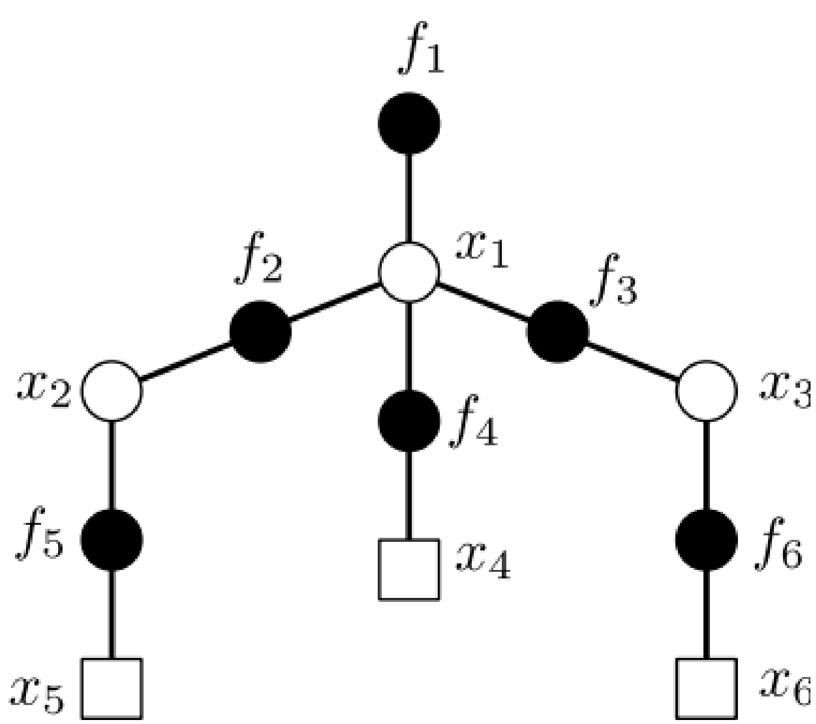
x1	x2	x3	x4	P(x1,x2,x3,
1.0000	1.0000	1.0000	1.0000	0.1120
1.0000	1.0000	1.0000	2.0000	0.1120
1.0000	1.0000	2.0000	1.0000	0.0392
1.0000	1.0000	2.0000	2.0000	0.0168
1.0000	2.0000	1.0000	1.0000	0.0420
1.0000	2.0000	1.0000	2.0000	0.0420
1.0000	2.0000	2.0000	1.0000	0.2352
1.0000	2.0000	2.0000	2.0000	0.1008
2.0000	1.0000	1.0000	1.0000	0.0600
2.0000	1.0000	1.0000	2.0000	0.0600
2.0000	1.0000	2.0000	1.0000	0.0210
2.0000	1.0000	2.0000	2.0000	0.0090
2.0000	2.0000	1.0000	1.0000	0.0150
2.0000	2.0000	1.0000	2.0000	0.0150
2.0000	2.0000	2.0000	1.0000	0.0840
2.0000	2.0000	2.0000	2.0000	0.0360

算法原理-因子图



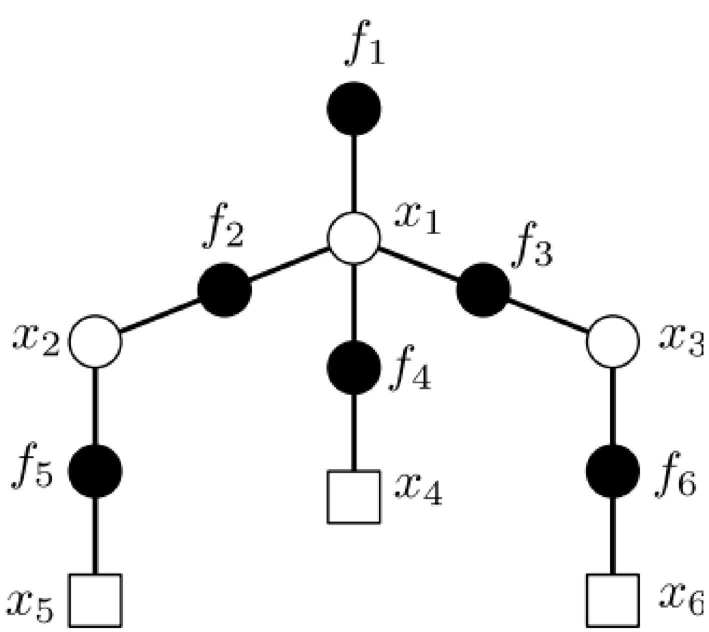
$$p(\boldsymbol{x}) = p(x_5|x_2)p(x_2|x_1)p(x_4|x_1)p(x_3|x_1)p(x_6|x_3)p(x_1)$$

$$p(x_1 = male) = 0.6 \quad p(x_1 = female) = 0.4$$
$$p(x_4 = addicted|x_1 = male) = 0.7 \quad p(x_4 = non\_addicted|x_1 = male) = 0.3$$
$$p(x_4 = addicted|x_1 = female) = 0.4 \quad p(x_4 = non\_addicted|x_1 = female) = 0.6$$
$$p(x_3 = smoking|x_1 = male) = 0.8 \quad p(x_3 = non\_smoking|x_1 = male) = 0.2$$
$$p(x_3 = smoking|x_1 = female) = 0.6 \quad p(x_3 = non\_smoking|x_1 = female) = 0.4$$
$$p(x_6 = cancer|x_3 = smoking) = 0.2 \quad p(x_6 = healthy|x_3 = smoking) = 0.8$$
$$p(x_6 = cancer|x_3 = non\_smoking) = 0.1 \quad p(x_6 = healthy|x_3 = non\_smoking) = 0.9$$
$$p(x_5 = HIV|x_2 = have\_sex) = 0.2 \quad p(x_5 = healthy|x_2 = have\_sex) = 0.8$$
$$p(x_5 = HIV|x_2 = non\_sex) = 0.1 \quad p(x_5 = healthy|x_2 = non\_sex) = 0.9$$
$$p(x_2 = have\_sex|x_1 = male) = 0.8 \quad p(x_2 = non\_sex|x_1 = male) = 0.2$$
$$p(x_2 = have\_sex|x_1 = female) = 0.7 \quad p(x_2 = non\_sex|x_1 = female) = 0.3$$



$$f_5(x_2, x_5) = p(x_5|x_2) \quad f_2(x_1, x_2) = p(x_2|x_1) \quad f_4(x_1, x_4) = p(x_4|x_1)$$
$$f_3(x_1, x_3) = p(x_3|x_1) \quad f_6(x_3.x_6) = p(x_6|x_3) \quad f_1(x_1) = p(x_1)$$

算法原理-因子图



$$u_{x_5\to f_5}(x_5)=1$$

$$u_{f_5\to x_2}(x_2)=\sum_{x_5} f_5(x_2,x_5)u_{x_5\to f_5}(x_5)=\sum_{x_5} f_5(x_2,x_5)$$

$$u_{x_2\to f_2}(x_2)=u_{f_5\to x_2}(x_2)=\sum_{x_5} f_5(x_2,x_5)$$

$$u_{f_2\to x_1}(x_1)=\sum_{x_2} f_2(x_1,x_2)u_{x_2\to f_2}(x_2)=\sum_{x_2}\sum_{x_5} f_2(x_1,x_2)f_5(x_2,x_5)$$

$$u_{f_1\to x_1}(x_1)=f_1(x_1)$$

$$u_{x_4\to f_4}(x_4)=1$$

$$u_{f_4\to x_1}(x_1)=\sum_{x_4} f_4(x_1,x_4)u_{x_4\to f_4}(x_4)=\sum_{x_4} f_4(x_1,x_4)$$

$$u_{x_1\to f_3}(x_1)=u_{f_1\to x_1}(x_1)\cdot u_{f_2\to x_1}(x_1)\cdot u_{f_4\to x_1}(x_1)$$

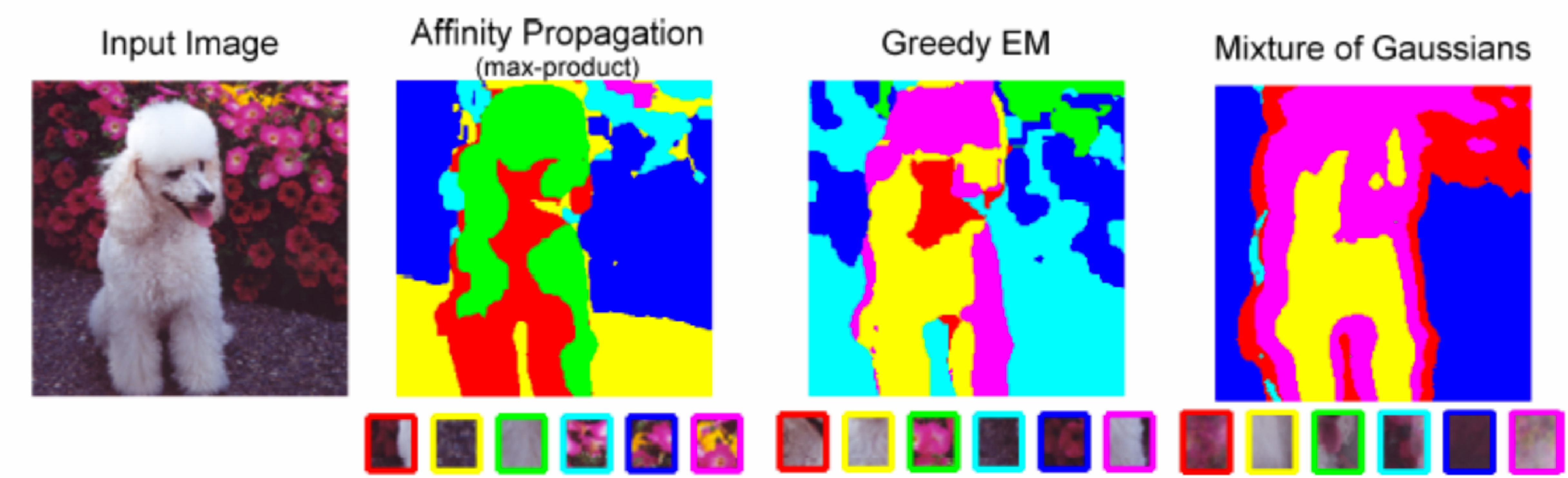
$$\begin{aligned}u_{x_1\to f_3}(x_1)=u_{f_3\to x_3}(x_3)&\\=u_{f_1\to x_1}(x_1)\cdot u_{f_2\to x_1}(x_1)\cdot u_{f_4\to x_1}(x_1)&\\=f_1(x_1)\cdot\sum_{x_2}\sum_{x_5} f_2(x_1,x_2)f_5(x_2,x_5)\cdot\sum_{x_4} f_4(x_1,x_4)&\\=\sum_{x_2}\sum_{x_4}\sum_{x_5} f_1(x_1)f_2(x_1,x_2)f_4(x_1,x_4)f_5(x_2,x_5)&\end{aligned}$$

$$u_{x_6\to f_6}(x_6)=1$$

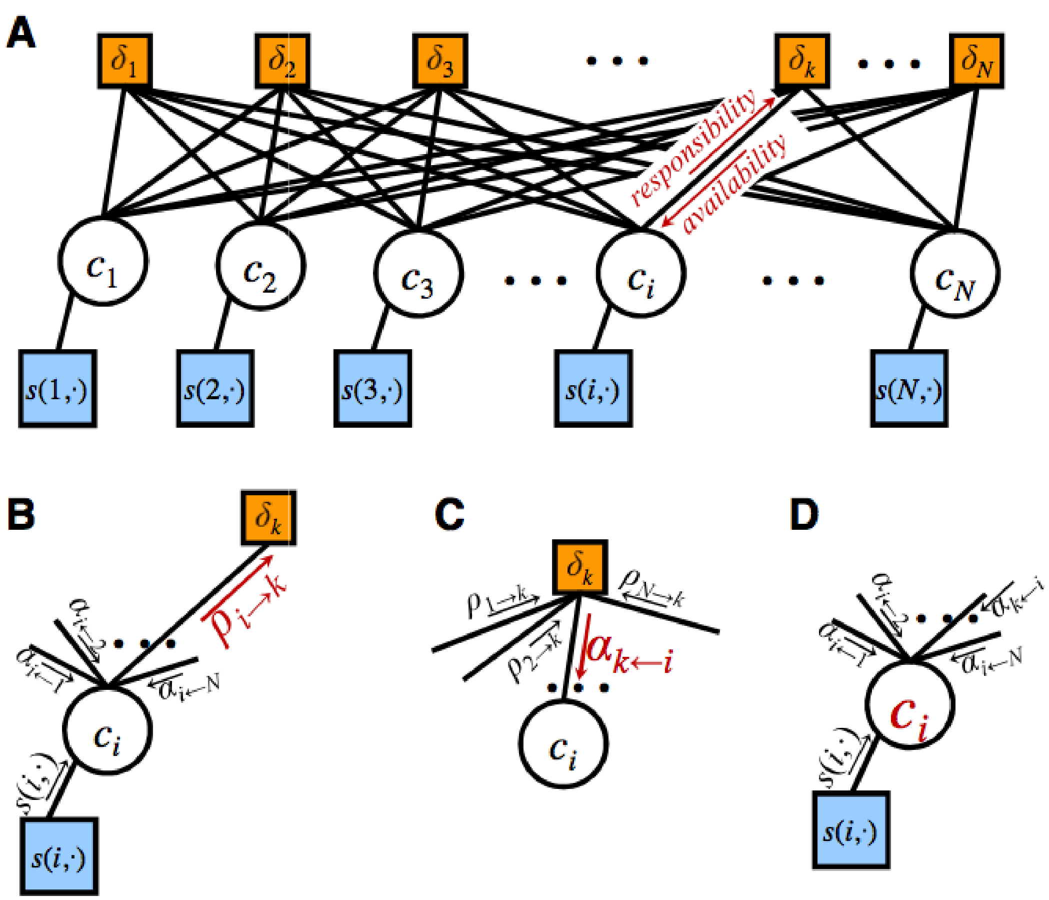
$$u_{f_6\to x_3}(x_3)=\sum_{x_6} f_6(x_3,x_6)u_{x_6\to f_6}(x_6)=\sum_{x_6} f_6(x_3,x_6)$$

$$\begin{aligned}p(x_3)&=u_{f_6\to x_3}(x_3)\cdot u_{f_3\to x_3}(x_3)\\&=\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5}\sum_{x_6} f_1(x_1)f_2(x_1,x_2)f_3(x_1,x_3)f_4(x_1,x_4)f_5(x_2,x_5)f_6(x_3,x_6)\\&=\sum\sum\sum\sum\sum p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_1)p(x_5|x_2)p(x_6|x_3)\end{aligned}$$

算法原理-Affinity Propagation



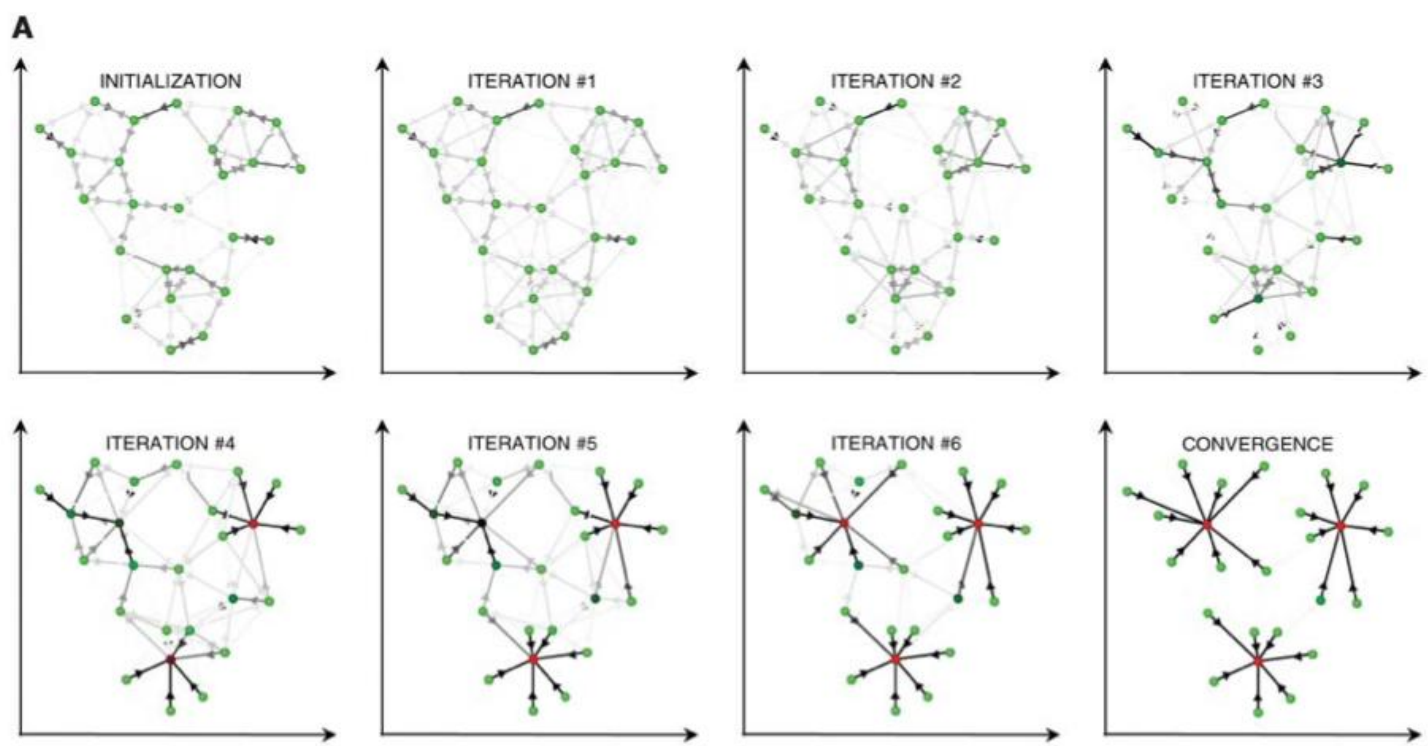
算法原理-Affinity Propagation



$$\mathcal{S}(\mathbf{c}) = -E(\mathbf{c}) + \textstyle\sum_{k=1}^{N} \delta_k(\mathbf{c}) = \sum_{i=1}^{N} s(i,c_i) + \sum_{k=1}^{N} \delta_k(\mathbf{c}) \qquad \text{where } \delta_k(\mathbf{c}) = \begin{cases} -\infty, & \text{if } c_k \neq k \text{ but } \exists i\colon c_i = k \\ 0, & \text{otherwise} \end{cases} \tag{S1}$$

$$\rho_{i\to k}(c_i) = s(i,c_i) + \sum\nolimits_{k':k'\neq k} \alpha_{i\leftarrow k'}(c_i)$$

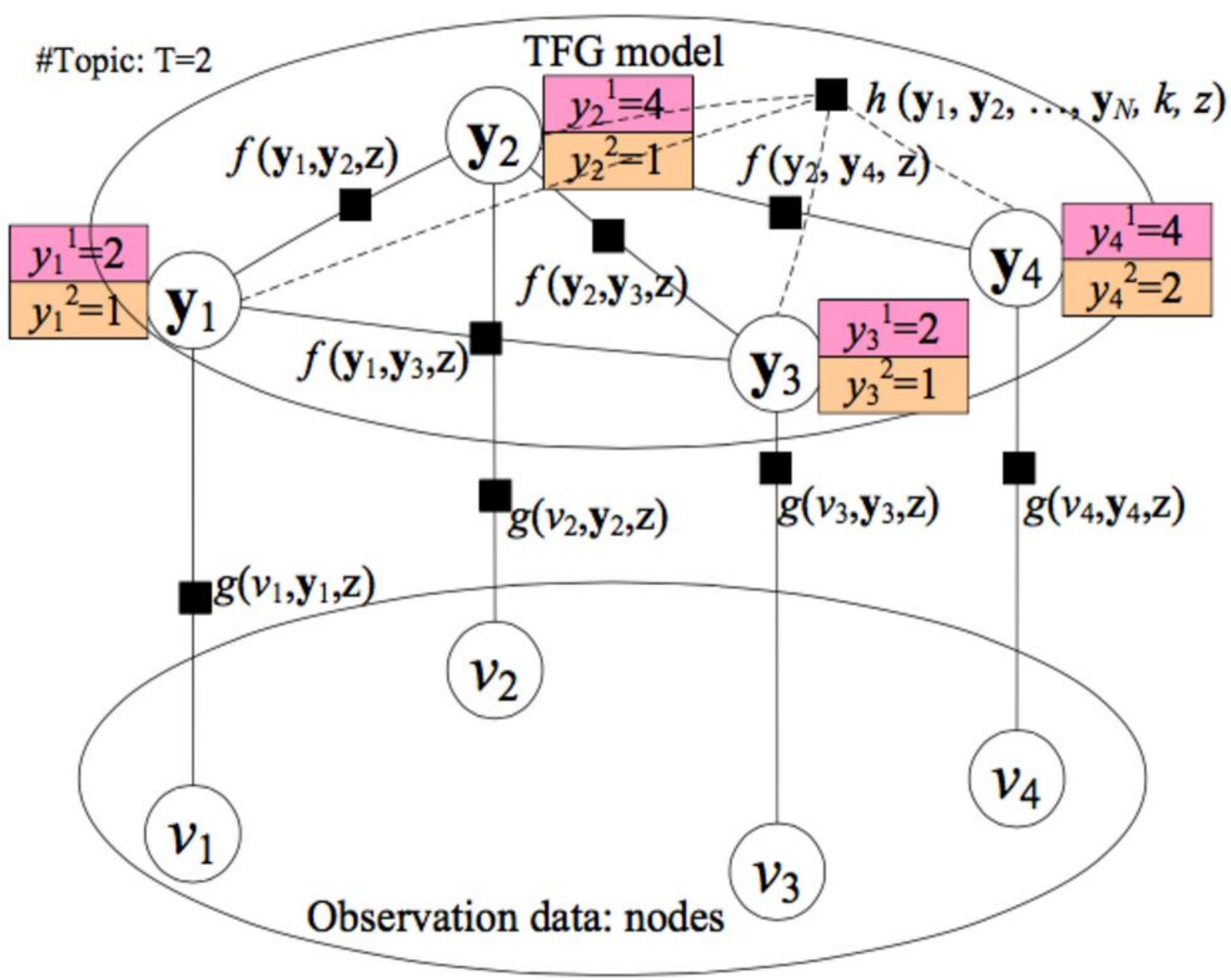
$$\alpha_{i\leftarrow k}(c_i) = \overbrace{\max_{j_1,\ldots,j_{i-1},j_{i+1},\ldots,j_N} \left[\delta_k(j_1,\ldots,j_{i-1},c_i,j_{i+1},\ldots,j_N) + \sum\nolimits_{i':i'\neq i} \rho_{i'\to k}(j_{i'})\right]}^{\text{best possible configuration satisfying } \delta_k \text{ given } c_i}$$



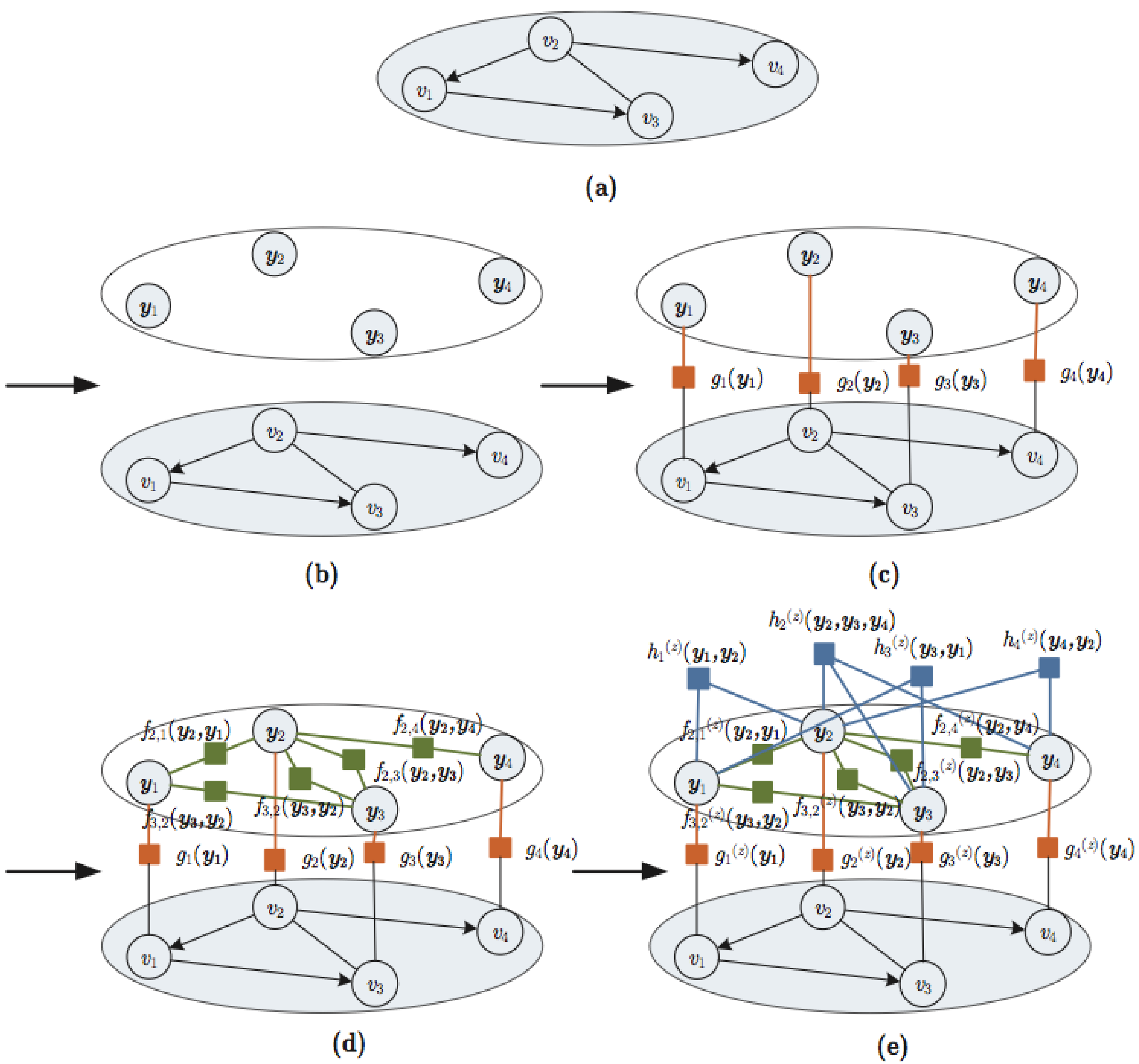
算法原理-TAP构建

$$P(\mathbf{y_{1:N}}) = \frac{1}{Z}\prod_{z=1}^{T}\prod_{i=1}^{N} g_i^z(\mathbf{y_i}) \prod_{z=1}^{T}\prod_{e_{i,j}\in E} f_{i,j}^z(\mathbf{y_i}, \mathbf{y_j}) \prod_{z=1}^{T}\prod_{k=1}^{N} h_k^z(\mathbf{y_{I(k)\cup\{k\}}})$$

$$= \frac{1}{Z}\prod_{z=1}^{T}\left(\prod_{i=1}^{N} g_i^z(y_i^z) \prod_{e_{i,j}\in E} f_{i,j}^z(y_i^z, y_j^z) \prod_{k=1}^{N} h_k^z(y_{I(k)\cup\{k\}}^z)\right)$$



算法原理-TAP构建

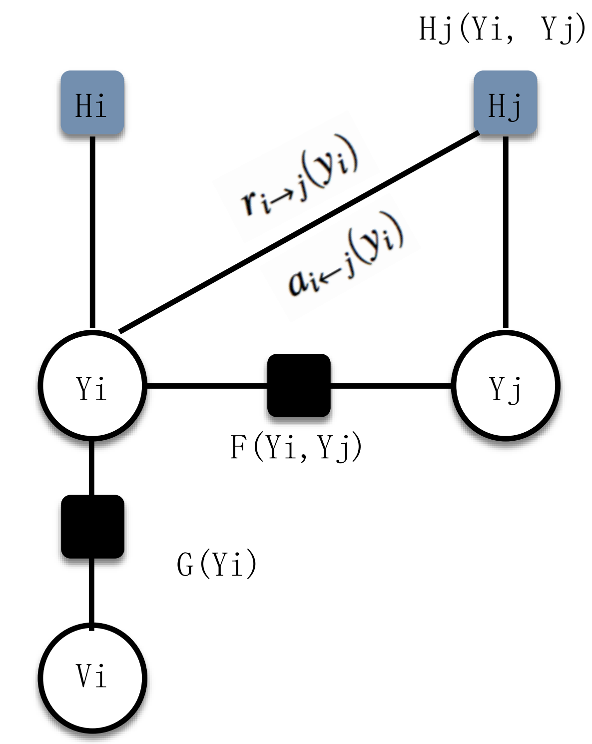


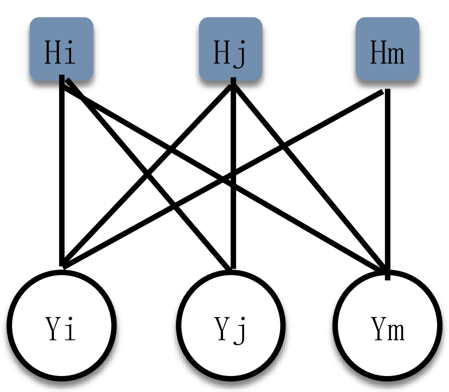
$$g_i^z(\mathbf{y_i}) = g_i^z(y_i^z) = \begin{cases} \kappa^z w_{i,y_i^z}^z & \text{如果} y_i^z \in O(i) \\ \kappa^z \sum_{j \in I(i)} w_{j,i}^z & \text{如果} y_i^z = i \\ 0 & \text{否则} \end{cases}$$

$$f_{i,j}^z(\mathbf{y_i}, \mathbf{y_j}) = f_{i,j}^z(y_i^z, y_j^z) = \begin{cases} \lambda & \text{如果} y_i^z = y_j^z \\ 1 - \lambda & \text{如果} y_i^z \neq y_j^z \end{cases}$$

$$h_k^z(\mathbf{y_{I(k)\cup\{k\}}}) = h_k^z(y_{I(k)\cup\{k\}}^z) = \begin{cases} 0 & \text{如果} y_k^z = k \text{ 且} \forall i \in I(k) : y_i^z \neq k \\ 1 & \text{否则} \end{cases}$$

算法原理-TAP推导

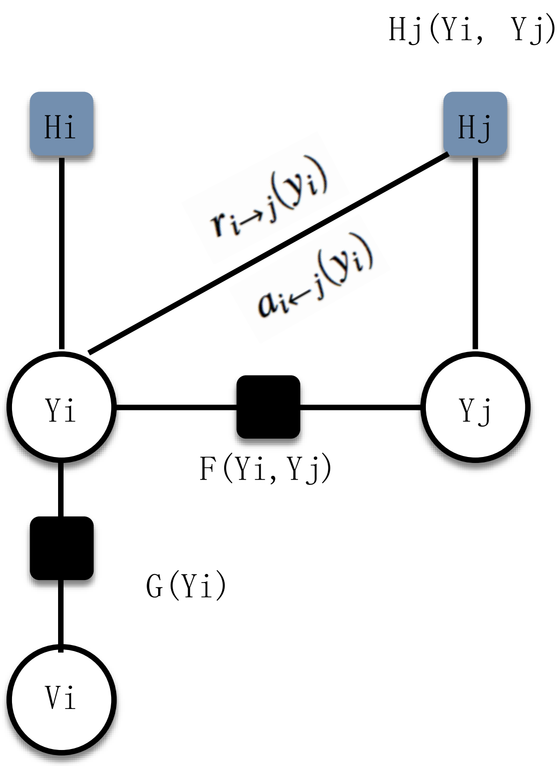




$$r_{i\to j}(y_i) = \log \kappa g_i(y_i) + \sum_{k\in O(i)\setminus\{j\}} a_{i\leftarrow k}(y_i)$$

$$\begin{aligned}
a_{i\leftarrow j}(y_i) &= \max_{y_{O(i)}} \left(\log h_j(y_{1:N}) + \sum_{k\in I(j)\cup\{j\}\setminus\{i\}} r_{k\to j}(y_k) \right) \\
&= \max_{y_{O(i)}} \sum_{k\in I(j)\cup\{j\}\setminus\{i\}} r_{k\to j}(y_k) \qquad \text{如果}(v_j \neq j) \vee (\exists k \in I(j), y_k = j) \\
&= \begin{cases}
\displaystyle\sum_{k\in I(j)} \max_{y_k} r_{k\to j}(y_k) + \max_{k\in I(j)} \left(r_{k\to j}(j) - \max_{y_k} r_{k\to j}(y_k) \right) & \text{如果} i = j = y_i \\
\displaystyle\sum_{k\in I(j)} \max_{y_k} r_{k\to j}(y_k) & \text{如果} i = j \neq y_i \\
\displaystyle\sum_{k\in I(j)\cup\{j\}\setminus\{i\}} \max_{y_k} r_{k\to j}(y_k) & \text{如果} i \neq j = y_i \\
\displaystyle\sum_{k\in I(j)\cup\{j\}\setminus\{i\}} \max_{y_k} r_{k\to j}(y_k) & \\
\quad + \max \left\{ \begin{array}{l} r_{j\to j}(j) - \max\limits_{y_j} r_{j\to j}(y_j) \\ \quad + \max\limits_{k\in I(j)\setminus\{i\}} \left(r_{k\to j}(j) - \max\limits_{y_k} r_{k\to j}(y_k) \right), \\ \max\limits_{y_j\neq j} \left(r_{j\to j}(y_j) - \max\limits_{y_i} r_{j\to j}(y_j) \right) \end{array} \right\} & \text{如果} i \neq j \neq y_i
\end{cases}
\end{aligned}$$

算法原理-TAP推导



$$r_{i\to j}(y_i) = r^1_{i\to j}(y_i) + r^2_{i\to j}, \text{其中} r^2_{i\to j} = \max_{y_i\neq j} r_{i\to j}(y_i)$$

$$a_{i\leftarrow j}(y_i) = a^1_{i\leftarrow j}(y_i) + a^2_{i\leftarrow j}, \text{其中} a^2_{i\leftarrow j} = a_{i\leftarrow j}(y_i)|_{y_i\neq j}$$

可以如下得到一些性质:

$$\max_{y_i\neq j} r^1_{i\to j}(y_i) = 0$$

$$\max_{y_i} r^1_{i\to j}(y_i) = \max\left\{0, r^1_{i\to j}(j)\right\}$$

$$a^1_{i\leftarrow j}(y_i) = 0 \quad \text{if } y_i \neq j$$

$$\sum_{k:k\neq j} a^1_{i\leftarrow k}(y_i) = \begin{cases} a^1_{i\leftarrow y_i}(y_i) & \text{if } y_i \neq j \\ 0 & \text{if } y_i = j \end{cases}$$

$$\sum_k a^1_{i\leftarrow k}(y_i) = a^1_{i\leftarrow y_i}(y_i)$$

$$a^z_{ii} = \max_{k\in I(j)} \min\left\{r^z_{kj}, 0\right\}$$

$$a^z_{ij} = \min\left\{-\min\left\{r^z_{jj}, 0\right\} - \max_{k\in I(j)\setminus\{i\}} \min\left\{r^z_{kj}, 0\right\}, \max\left\{r^z_{jj}, 0\right\}\right\}$$

$$r^z_{ij} = g^z_{ij} + \sum_{k\in I(i)\cup O(i)} c^z_{ikj} - \max_{j'\in O(i)\cup\{i\}\setminus\{j\}} \left(g^z_{ij'} + a^z_{ij'} + \sum_{k\in I(i)\cup O(i)} c^z_{ikj'}\right)$$

算法原理-TAP流程

Input: $G = (V, E)$ and topic distributions $\{\theta_v\}_{v \in V}$

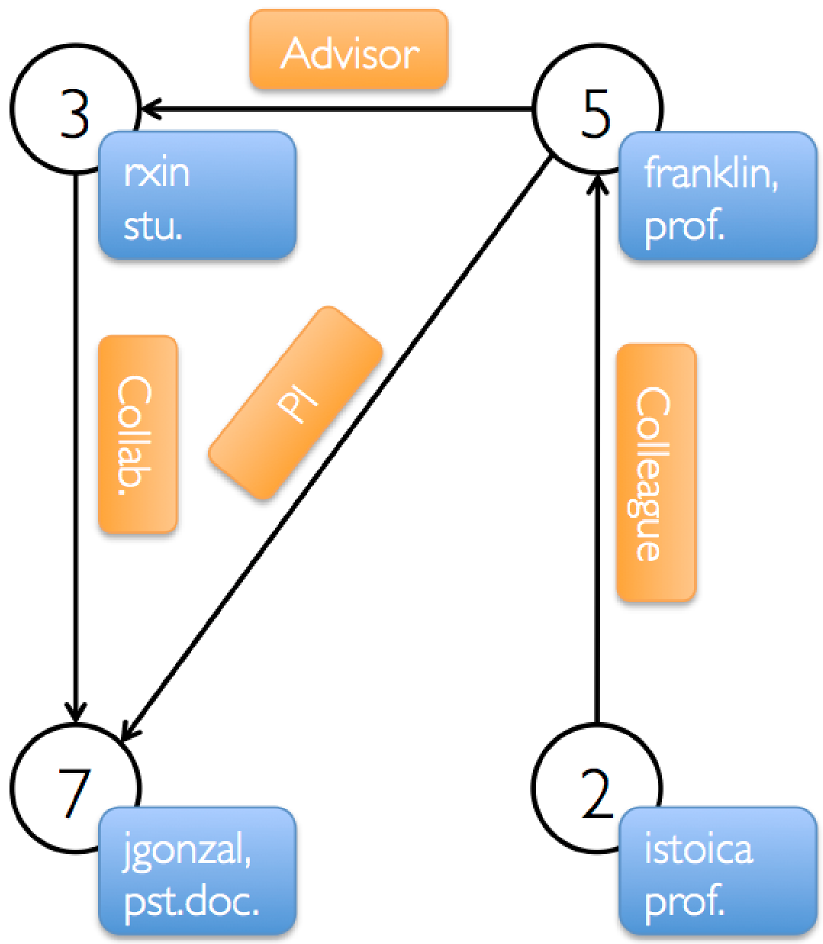
Output: topic-level social influence graphs $\{G_z = (V_z, E_z)\}_{z=1}^{T}$

Calculate the node feature function $g(v_i, \mathbf{y}_i, z)$;
Calculate b_{ij}^{z} according to Eq. 8;
Initialize all $\{r_{ij}^{z}\} \leftarrow 0$;
repeat
    foreach *edge-topic pair* (e_{ij}, z) **do**
        Update r_{ij}^{z} according to Eq. 5;
    end
    foreach *node-topic pair* (v_j, z) **do**
        Update a_{jj}^{z} according to Eq. 6;
    end
    foreach *edge-topic pair* (e_{ij}, z) **do**
        Update a_{ij}^{z} according to Eq. 7;
    end
until *convergence*;
foreach *node* v_t **do**
    foreach *neighboring node* $s \in NB(t) \cup \{t\}$ **do**
        Compute μ_{st}^{z} according to Eq. 9;
    end
end
Generate $G_z = (V_z, E_z)$ for every topic z according to $\{\mu_{st}^{z}\}$;

Algorithm 1: The new TAP learning algorithm.

工程实现-Graphx

Property Graph

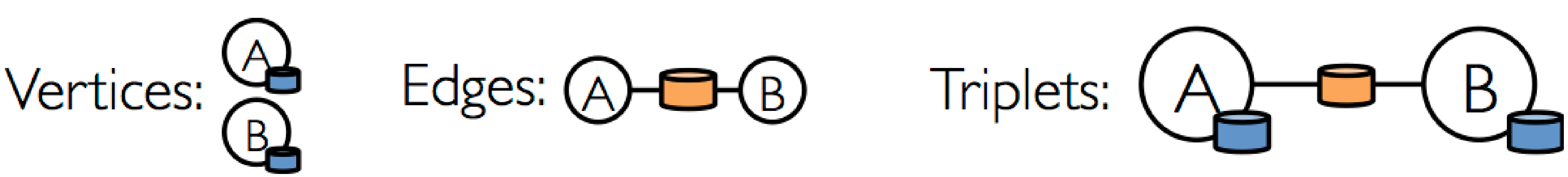


Vertex Table

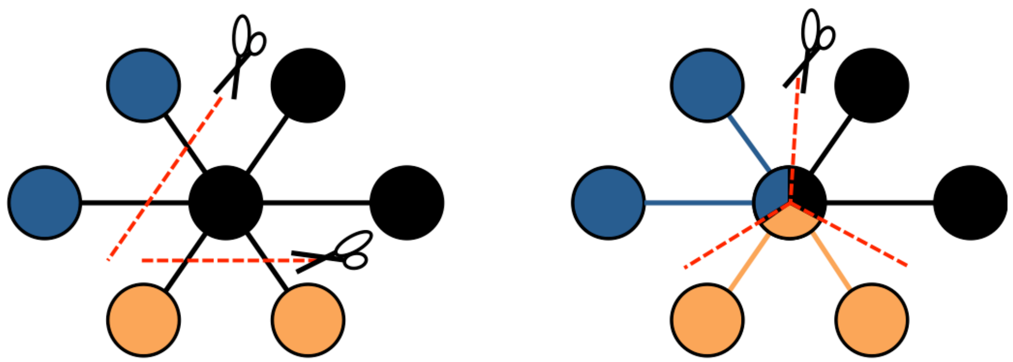
Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI



工程实现-Graphx



Edge Cut　　　　　　Vertex Cut

```
       __________________________________

v0     | P0 *     | P1       | P2    *  |
v1     |  ****    |  *       |          |
v2     |  ******* |      **  |  ****    |
v3     |  *****   |  *  *    |       *  |
       ----------------------------------
v4     | P3 *     | P4 ***   | P5 **  * |
v5     |  *  *    |  *       |          |
v6     |       *  |      **  |  ****    |
v7     |  * * *   |  *  *    |       *  |
       ----------------------------------
v8     | P6   *   | P7    *  | P8  *   *|
v9     |     *    |  *    *  |          |
v10    |       *  |      **  |  *  *    |
v11    | * <-E    |  ***     |       ** |
       ----------------------------------
```

1. RandomVertexCut

通过取源顶点和目标顶点id的哈希值来将边分配到不同的分区。

2. CanonicalRandomVertexCut

哈希值的产生带有确定的方向（即两个顶点中较小id的顶点在前）

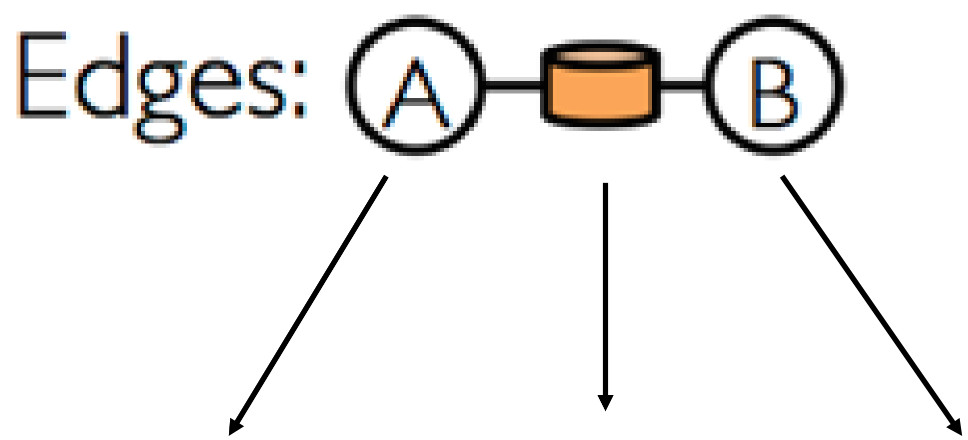
3. EdgePartition1D

仅仅根据源顶点id来将边分配到不同的分区。有相同源顶点的边会分配到同一分区。

4. EdgePartition2D

使用稀疏边连接矩阵的2维区分来将边分配到不同的分区

工程实现-资源估计



数据	**8**	**+**	**8**	**+**	**8**
指针	**8**		**8**		**8**

下限 **8*6*数量=48*100亿=447G**

上限 **447G * 10 = 4.4T**

1 ~ 20

工程实现-参数配置

目标是用尽当前集群资源，一个集群有6个节点有NodeManager在上面运行，每个节点有16个core以及64GB的内存

1. NodeManager容量计算

   yarn.nodemanager.resource.memory-mb=63*1024=64512(MB)
   yarn.nodemanager.resource.cpu-vcores = 16 - 1 = 15

2. 最大配置计算

   - num-executors 6
   - executor-cores 15
   - executor-memory 63G

   最大core量　90=6*15
   单executor最大5个core　2-3个比较合适
   最大memory 小于63GB

3. 根据约束调整参数

executor大小

90 cores / 5 = 18 executors
18 executors - 1 = 17。这个配置会在每个节点上生成3个 executor，除了应用的master运行的机器，这台机器上只会运行2个 executor。

memory大小

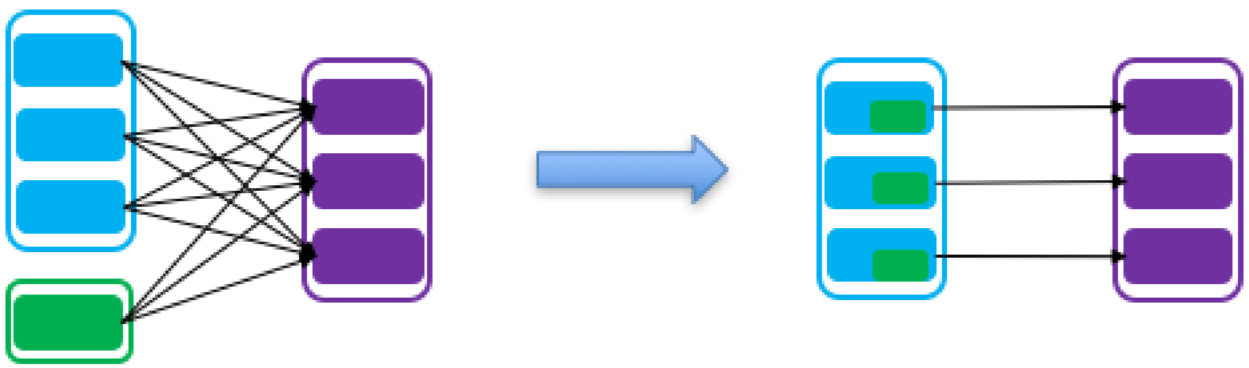
63G / 3 executor = 21 G。
21 G * (1 - 0.07) ~ 19 G。

工程实现-减少shuffle

- Broadcast+map - 大表join小表
  - 避免shuffle

```
// 传统的join操作会导致shuffle操作
val joinedRdd = rdd1.join(rdd2)

// 使用Broadcast将一个数据量较小的RDD作为广播变量
val rdd2Data = rdd2.collect()
val rdd2DataBroadcast = sc.broadcast(rdd2Data)
val joinedRdd = rdd1.map{ x =>
  val rdd2Data = rdd2DataBroadcast.value
  …
}
```

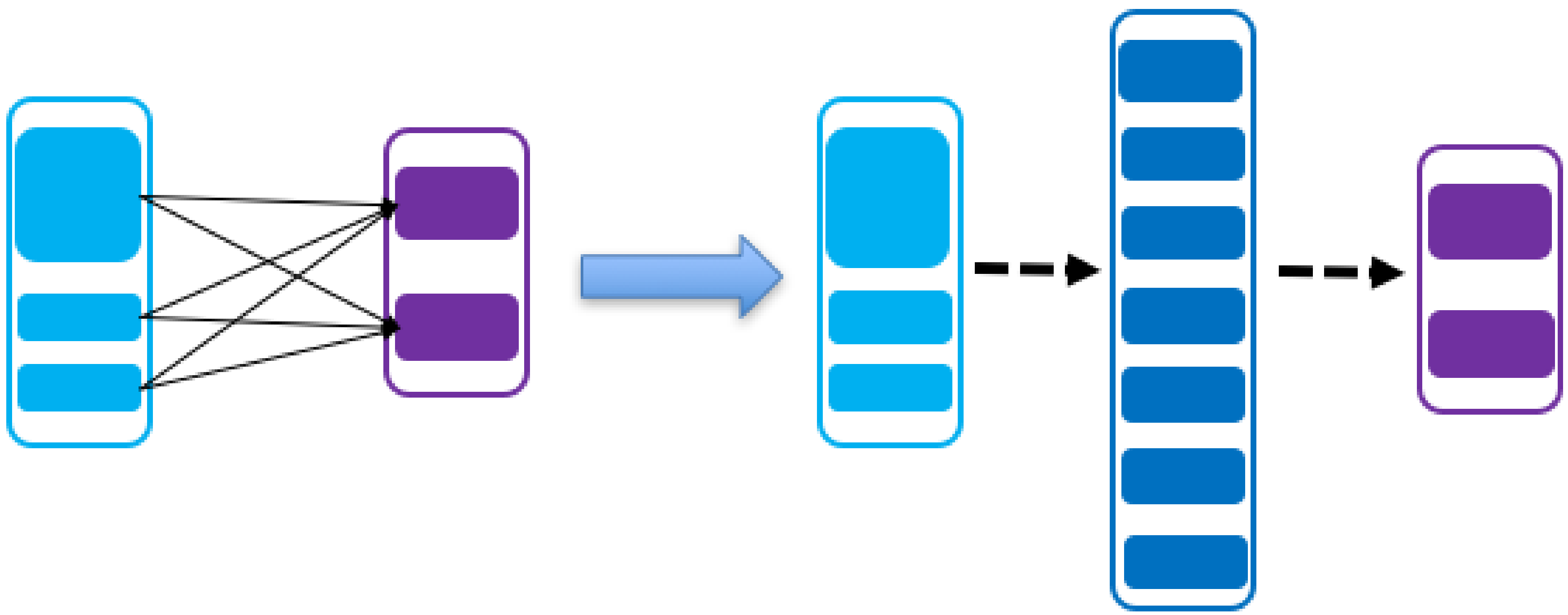


工程实现-数据倾斜

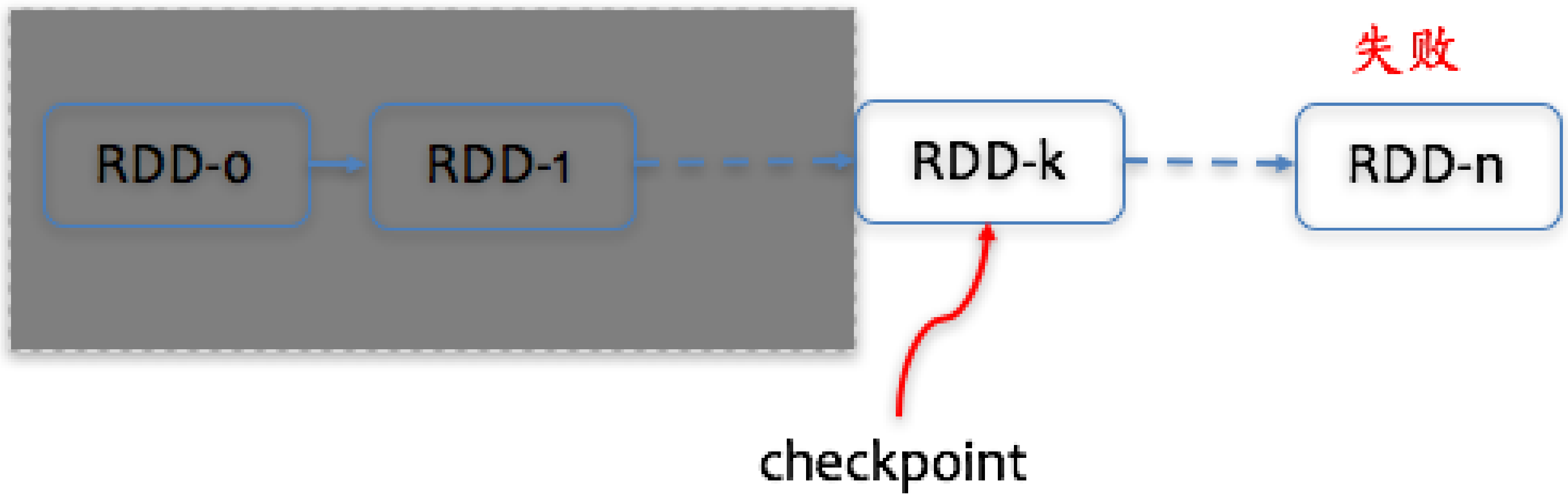
- 数据倾斜处理 - reduceByKey
  - 将key随机分到100个桶中，先对每个桶进行汇总，再汇总每个key

```
rdd.map( x => ((x._1, Random.nextInt(100)),
    x._2)).reduceByKey(_ + _)
  .map( x => (x._1._1, x._2).reduceByKey(_ + _)
```

  - 自定义partitioner



工程实现-Checkpoint



工程实现-Cache

下面的代码正确的是哪个

```
rdd.cache
rdd.filter(rule)
```

```
rdd.cache
rdd.take
rdd.filter(rule)
```

```
rdd.cache
rdd.count
rdd.filter(rule)
```

小结

- **算法层次**

  时间和空间的优化 ： sum-product

  并发和锁的优化 : max-product

  数据结构的设计 : graphx框架

- **系统层次**

  系统负载均衡 ： 资源估计，减少shuffle, 数据倾斜

  充分利用硬件性能 ： 参数设置

  减少额外开销 ：多个小JVM，checkpoint

- **代码层次**

  Cache ： 保障完整Cache

  执行顺序 ： inline,checkpoint

  语言优化 ： 内存小心使用