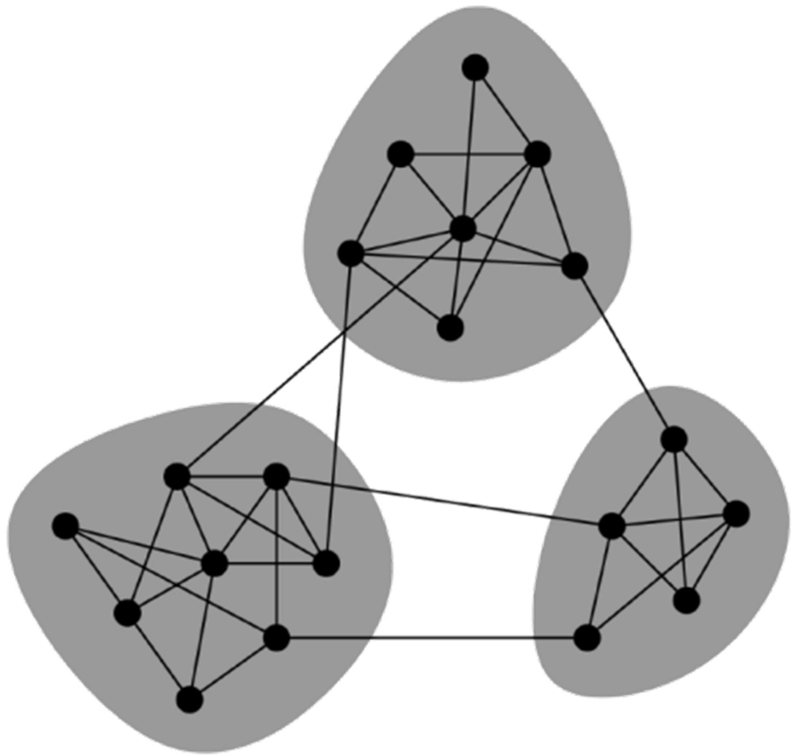
第8-3章： Network Module

- Definition
- Module detection
- Bayesian approach
- Markov clustering algorithm

Network Modular



Modularity

- Suppose we are given a candidate division of the vertices into some number of groups. The modularity of this division is defined to be the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random.

http://en.wikipedia.org/wiki/Modularity_(networks)

Modularity

- A_{ij}: adjacency matrix
- k_i: degree
- m: total number of edges

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \delta(c_i, c_j)$$

Modularity

- For two class problem, let s_i=1 of node I belongs to group1 and s_i=-1 if it belongs to group 2,

$$\delta(c_i, c_j) = \frac{1}{2}(s_i s_j + 1)$$

$$Q = \frac{1}{4m} \sum_{ij} S^T B S$$

$$B = (B_{ij}), B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

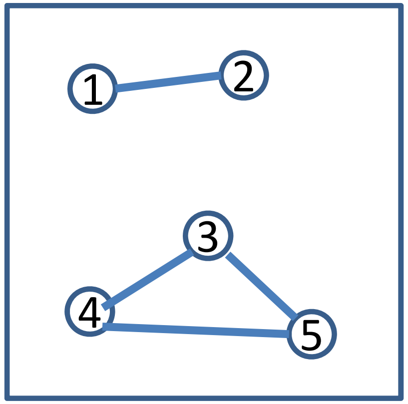
$$S = (s_1, \cdots, s_n)^T$$

Example

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$m = 4, k_1 = k_2 = 1$$

$$k_3 = k_4 = k_5 = 1$$



$$B = \frac{1}{8} \begin{pmatrix} -1 & 7 & -2 & -2 & -2 \\ 7 & -1 & -2 & -2 & -2 \\ -2 & -2 & -4 & 4 & 4 \\ -2 & -2 & 4 & -4 & 4 \\ -2 & -2 & 4 & 4 & -4 \end{pmatrix}$$

Spectrum Method

- The largest eigenvectors will gives the best grouping, positive entries corresponding to one class, and negative ones corresponding to another class.

- This can be achieved by power method

$$\lim_{k \to +\infty} = \frac{A^k e}{e^T A^k e} = w$$

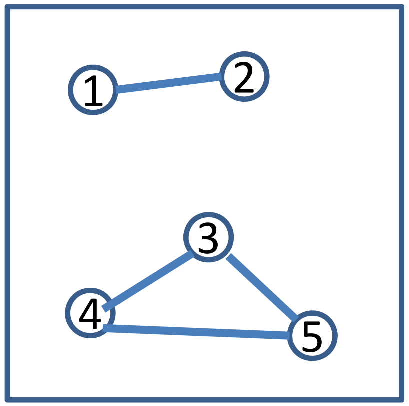
where $e=(1,1,\ldots,1)^T$

Example

- 对于上述矩阵B, 可以计算出最大特征值为10, 对应的特征向量

$$v = (-0.55, -0.55, 0.37, 0.37, 0.37)$$

- 于是我们对节点的划分为{1,2}; {3,4,5}



优化方法

- 既然现在有一个衡量划分“好坏”的量Q, 那么一般的优化方法都可以使用;
  - 1. 给定初始划分
  - 2. 对于划分的某种修正, 计算Q的改变量
  - 3. 依据一定的原则考虑是否接受这种修正，重复步骤2, 直到某种收敛条件满足。
- Greedy方法
- 模拟退火方法

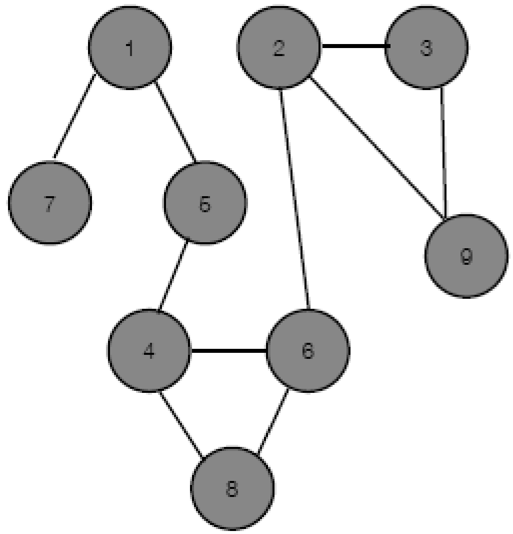
A Bayesian Approach to Network Modularity

Slides for this part are mainly from Hofman's talk

*www.jakehofman.com/talks/**apam_20071019**.pdf*

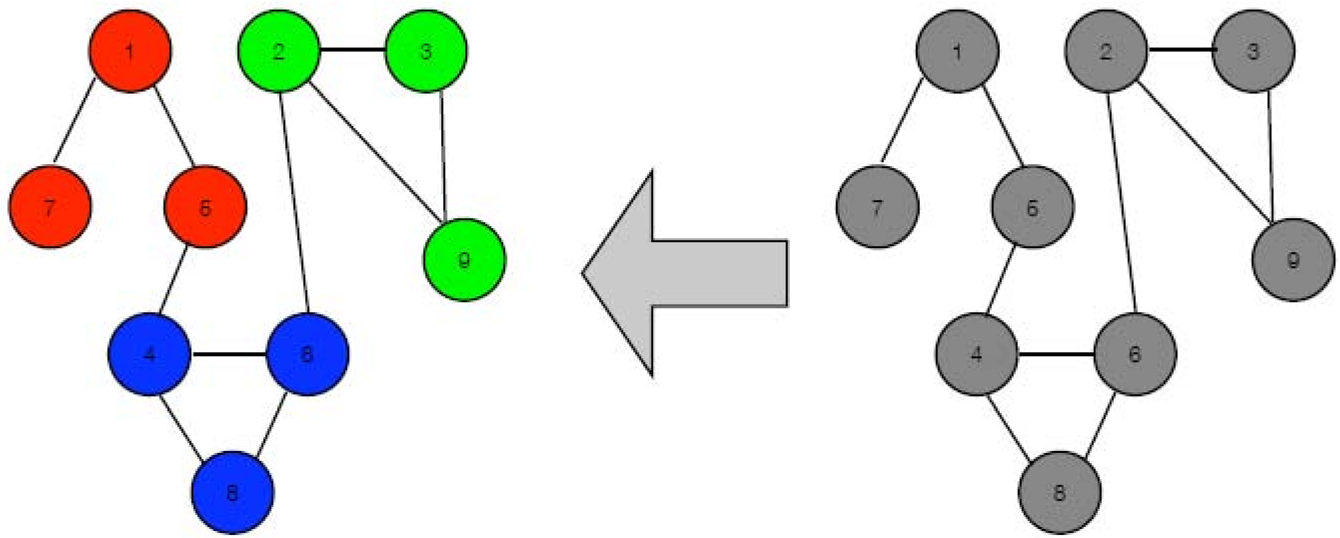
Overview: Modular Networks

- Given a network
  - Assign nodes to modules?
  - Determine number of modules(scale/complexity)?



Overview: Modular Networks

- With a generative model of modular networks, rules of probability tell us how to calculate model parameters (e.g. number of modules & assignments)



Generative Models

Known model

(parameters, assignment variables, complexity)

→

Generate Synthetic data

Inferred Model

(parameters, latent variables, complexity)

←

Observed real data

Generating Modular Networks

- For each node:
  - **Roll K-sided die to determine** $z_i=1,\ldots,K$, the (unobserved) module assignment for ith node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip "in community" coin with bias $\boldsymbol{\theta_c}$ to determine** edge
  - If $z_i \neq z_j$, **flip "between communities" coin with bias $\boldsymbol{\theta_d}$** to determine edge

Generating Modular Networks

- Die rolling, coin flipping, and priors:

$$P(\vec{z}|\vec{\pi}) = \prod_{\mu=1}^{K} \pi_{\mu}^{n_\mu}$$

$$P(A|\vec{z},\vec{\pi},\vec{\theta}) = \theta_c^{c+}(1-\theta_c)^{c-}\theta_d^{d+}(1-\theta_d)^{d-}$$

$$P(\vec{\theta}) = Beta(\theta_c;\tilde{c}_{+0},\tilde{c}_{-0})Beta(\theta_d;\tilde{d}_{+0},\tilde{d}_{-0})$$

$$P(\vec{\pi}) = Dir(\vec{\pi},\tilde{\vec{n}})$$

Generating Modular Networks

- Edges within modules $$c_+ = \sum_{i,j} A_{ij}\delta_{z_i,z_j}$$
- Non-edges within modules $$c_- = \sum_{i,j} (1 - A_{ij})\delta_{z_i,z_j}$$
- Edges between modules $$d_+ = \sum_{i,j} A_{ij}(1 - \delta_{z_i,z_j})$$
- Non-edges between modules $$d_- = \sum_{i,j} (1 - A_{ij})(1 - \delta_{z_i,z_j})$$
- Nodes in each modules $$n_\mu = \sum_{i=1}^{N} \delta_{z_i,\mu}$$

Inferring Modular Networks

- From observed graph structure, infer distributions over module assignments, model parameters, and model complexity

$$P(\vec{\pi}, \vec{\theta} | A, K) = \frac{P(A | \vec{\pi}, \vec{\theta}, K) P(\vec{\pi}, \vec{\theta} | K)}{P(A | K)}$$

$$P(\vec{z} | A, K) = \frac{P(A | \vec{z}, K) P(\vec{z} | K)}{P(A | K)}$$

$$P(A|K) = \sum_{\vec{z}} \int d\vec{\theta} \int d\vec{\pi} P(A, \vec{z}, \vec{\pi}, \vec{\theta} | K) \longleftarrow$$

Can do integrals, but sum is intractable, $O(K^N)$

Approximate Inference for Modular Networks

- Jensen's inequality (log of expected value bounds expected value of log) for *any* distribution q

$$
\begin{aligned}
-\ln P(A|K) &= -\ln \sum_{\vec{z}} \int d\vec{\theta} \int d\vec{\pi} P(A, \vec{Z}, \vec{\pi}, \vec{\theta}|K) \\
&= -\ln \sum_{\vec{Z}} \int d\vec{\theta} \int d\vec{\pi} q(\vec{Z}, \vec{\pi}, \vec{\theta}) \frac{P(A, \vec{z}, \vec{\pi}, \vec{\theta}|K)}{q(\vec{z}, \vec{\pi}, \vec{\theta})} \\
&\leq \underbrace{- \sum_{\vec{z}} \int d\vec{\theta} \int d\vec{\pi} q(\vec{Z}, \vec{\pi}, \vec{\theta}) \ln \frac{P(A, \vec{z}, \vec{\pi}, \vec{\theta}|K)}{q(\vec{z}, \vec{\pi}, \vec{\theta})}}_{F\{q;A\}}
\end{aligned}
$$

Approximate Inference for Modular Networks

- F is a functional of q; find approximation to posterior by optimizing approximation to evidence
- Take $q(z, \pi, \theta)=q(z)q(\pi)q(\theta)$; $Q_{i\mu}$ is probability of node i in module μ

Variational Bayesian

$$
\begin{aligned}
F\{q; A\} = &- \ln \frac{Z_{\vec{\pi}} Z_c Z_d}{\tilde{Z}_{\vec{\pi}} \tilde{Z}_c \tilde{Z}_d} + \sum_{\mu=1}^{K} \sum_{i=1}^{N} Q_{i\mu} \ln Q_{i\mu} \\
&- (\tilde{c}_+ - (\langle c_+ \rangle + \tilde{c}_{+0})) \langle \ln \theta_c \rangle \\
&- (\tilde{c}_- - (\langle c_- \rangle + \tilde{c}_{-0})) \langle \ln(1 - \theta_c) \rangle \\
&- (\tilde{d}_+ - (\langle d_+ \rangle + \tilde{d}_{+0})) \langle \ln \theta_c \rangle \\
&- (\tilde{d}_- - (\langle d_- \rangle + \tilde{d}_{-0})) \langle \ln(1 - \theta_d) \rangle \\
&- \sum_{\mu=1}^{K} (\tilde{n}_\mu - (\langle n_\mu \rangle + \tilde{n}_{\mu 0})) \langle \ln \pi_\mu \rangle
\end{aligned}
$$

Variational Bayesian

- Where the expected counts

$$\langle c_+ \rangle = \frac{1}{2} Tr(Q^T A Q)$$
$$\langle c_- \rangle = \frac{1}{2} Tr(Q^T \overline{A} Q)$$
$$\langle d_+ \rangle = M - \langle c_+ \rangle$$
$$\langle d_- \rangle = C - M - \langle c_- \rangle$$
$$\langle n_\mu \rangle = \sum_{j=1}^{N} Q_{j\mu}$$

Variational Bayesian Method

- 问题：

  给定数据D, 缺失数据Z={Z_1, Z_2,…,Z_k}, 极大化后验概率分布 $P(Z|D)$,目标函数复杂，很难得到Close form.

- 策略

  寻找一个合适的近似分布Q(Z)

如何近似

- 解决两个问题
  - 如何度量Q(Z)和P(Z | D)的差异

    Kullback-Leibler Divergence

  - 如何得到简单的分布Q(Z)

    可分解分布(Factorized Approximation)

Kullback-Leibler Divergence

$$D_{KL}(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

- 具有如下三个性质
  - 不对称性：$D_{KL}(p||q) \neq D_{KL}(p||q)$
  - 非负性：$D_{KL}(p||q) \geq 0$，当且仅当p=q时为0’
  - 不满足三角不等式

优化基础

$$D_{kL}(Q||P) = \sum_{Z} Q(Z) \log \frac{Q(Z)}{P(Z|D)}$$

$$= \sum_{Z} Q(Z) \frac{Q(Z)}{P(Z,D)} + \log P(D)$$

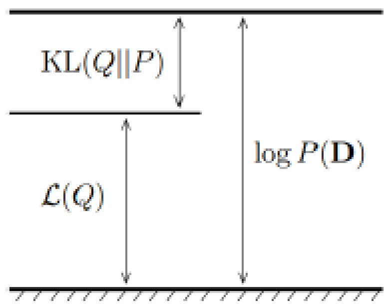
- 故令

$$L(Q) = \sum_{Z} Q(Z) \frac{P(Z,D)}{Q(Z)}$$

- 则

$$\log P(D) = D_{KL}(Q||P) + L(Q)$$

$$\max L(Q) \Leftrightarrow \min D_{KL}(Q||P)$$



优化基础

- 另一方面

$$
\begin{aligned}
L(Q) &= \sum_Z Q(Z) \log P(Z,D) - \sum_Z Q(Z) \log Q(Z) \\
&= E_Q[\log P(Z,D)] + H(Q)
\end{aligned}
$$

- 这里 log P(Z,D)在物理上称之为能量，而后者是Q的熵

平均场(Mean Field)近似

- 采用可分解分布进行近似

$$Q(Z) = \prod_{k=1}^{K} Q_i(z_i)$$

$$\int Q_i(z_i) dz_i = 1$$

平均场近似下的优化算法

- 优化问题

$$\begin{aligned}\max L(Q) &= \max\{\sum_Z Q(Z)\log P(Z,D) - \sum_Z Q(Z)\log Q(Z)\}\\ &= \max\{E_Q[\log P(Z,D)] + H(Q)\}\end{aligned}$$

- 约束条件

$$Q(Z) = \prod_{k=1}^{K} Q_i(z_i)$$

$$\int Q_i(z_i)dz_i = 1$$

优化问题求解

- 可以证明，上述优化问题的解：

$$Q_i(Z_i) \propto \frac{1}{C} \exp\left\{ E_{Q_{[-i]}(Z_{[-i]})}[\ln P(Z_i, Z_{[-i]}, D)] \right\}$$

- 其中 C为归一化因子. $Z_{[-i]}$ 表示除了Z_i之外的其它隐变量。

$$Q_{[-i]}(Z_{[-i]}) = \prod_{j \neq i} Q_j(Z_j)$$

优化公式推导

$$
\begin{aligned}
L(Q) &= E_Q[\ln P(Z,D)] + H(Q) \\
&= \int (\prod Q_i(Z_i)) \ln P(Z,D) dZ - \int (\prod Q_i(Z_i)) \sum_i \ln Q_i(Z_i)
\end{aligned}
$$

- 考虑Z=(Z_i, $Z_{[-i]}$)

$$
\begin{aligned}
& E_Q[\ln P(Z,D)] \\
&= \int (\prod Q_i(Z_i) dZ_i) \ln P(Z,D) dZ \\
&= \int Q_i(Z_i) dZ_i \int Q_{[-i]}(Z_{[-i]}) \ln P(Z,D) dZ_{[-i]} \\
&= \int Q_i(Z_i) \langle \ln P(Z,D) \rangle_{Q_{[-i]}} dZ_i \\
&= \int Q_i(Z_i) \ln \exp \langle \ln P(Z,D) \rangle_{Q_{[-i]}} dZ_i
\end{aligned}
$$

优化公式推导

- 定义 $Q_i^*(Z_i) = \frac{1}{C}\exp\langle \ln P(Z,D)\rangle_{Q_{[-i]}}$, 其中C为Q*的归一化因子. Q*定义了一个$Z_i$上的新的概率分布

$$
\begin{aligned}
&E_Q[\ln P(Z,D)] \\
&= \int Q_i(Z_i)\ln Q_i^*(Z_i)dZ_i + \ln C
\end{aligned}
$$

- 另一方面

$$
\begin{aligned}
H(Q) &= -\sum_i \int (\prod Q_i(Z_i)) \sum_i \ln Q_i(Z_i)dZ \\
&= -\sum_i \int Q_i(Z_i)\ln Q_i(Z_i)dZ_i \int Q_{[-i]}(Z_{[-i]})dZ_{[-i]} \\
&= -\sum_i \int Q_i(Z_i)\ln Q_i(Z_i)dZ_i
\end{aligned}
$$

优化公式推导

- 于是

$$
\begin{aligned}
L(Q) &= \int Q_i(Z_i)\ln Q_i^*(Z_i)dZ_i + \ln C - \sum_i \int Q_i(Z_i)\ln Q_i(Z_i)dZ_i \\
&= \int Q_i(Z_i)\ln Q_i^*(Z_i)dZ_i - \int Q_i(Z_i)\ln Q_i(Z_i)dZ_i + \sum_{j\neq i} H(Q_j) + \ln C \\
&= -D_{KL}(Q_i||Q_i^*) + \sum_{j\neq i} H(Q_j) + \ln C
\end{aligned}
$$

- 上式作为Q_i的泛函，由变分原理得

$$
\frac{\partial}{\partial Q_i(Z_i)}[-D_{KL}(Q_i^*(Z_i)||Q_i(Z_i))] - \lambda_i(\int Q_i(Z_i)dZ_i - 1) = 0
$$

优化公式推导

- 实际上，直接由KL散度的非负性，只有当散度为0时，L(Q)取最大值，即

$$Q_i(Z_i) = Q_i^*(Z_i) = \frac{1}{C}\exp\left\{\langle \ln P(Z,D)\rangle_{Q_{[-i]}(Z_{[-i]})}\right\}$$

变分原理

- 泛函：以函数为自变量的函数

$$J(y)=\int_{x_0}^{x_1} F(x,y,y')dx$$

- 泛函J(y)在y=y(x)极大值：对于y的任何一个扰动δy

$$J(y+\delta y)\leq J(y)$$

- 从形式上看，和y是普通变量的含义类似。

变分原理

- 极大值的必要条件：一级变分为0

$$\frac{\partial F}{\partial y} - \frac{d}{dx}\frac{\partial F}{\partial y'} = 0$$

- 带约束$J_0(y)$极大值的必要条件(Lagrange乘子)

$$J_0(y) = \int_{x_0}^{x_1} G(x, y, y') = C$$

$$\tilde{J} = J - \lambda J_0$$

$$(\frac{\partial}{\partial y} - \frac{d}{dx}\frac{\partial}{\partial y'})(F - \lambda G) = 0$$

- 从形式上看，和y是普通变量的含义类似。

迭代算法

$$Q(Z)=\prod_{i=1}^{M}q(Z_i\mid D)$$

（1）　初始化 $Q^{(1)}(Z_i)$，可随机取；

（2）　在第 k 步，计算 Z_{-i} 的边缘密度 $Q^{[k]}(Z_{-i}\mid D)\propto\exp\int_{Z_i^*}Q^{[k-1]}(Z_i\mid D)\log P(Z_i,Z_{-i},D)dZ_i$

（3）　计算 Z_i 的边缘密度 $Q^{[k]}(Z_i\mid D)\propto\exp\int_{Z_{-i}^*}Q^{[k]}(Z_{-i}\mid D)\log P(Z_i,Z_{-i},D)dZ_{-i}$

（4）　理论上 $Q^{[\infty]}(Z_i\mid D)$ 将会收敛，则反复执行(2), (3)直到 $Q(Z_i)$, $Q(Z_{-i})$稳定，或稳定在某个小范围内。

（5）　最后，得 $Q(Z)=Q(Z_i\mid D)Q(Z_{-i}\mid D)$

Markov Clustering Algorithm

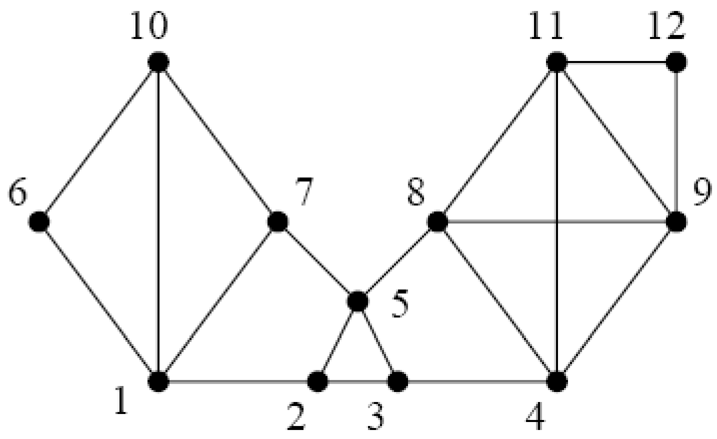
van Dongen. A cluster algorithm for graphs. Information Systems, 2000

K-length Path

- Basic idea: dense regions in sparse graphs corresponding with regions in which the number of k-length path is relatively large.

- Random walks can also be used to detect clusters in graphs, the idea is that the more closed is a subgraph, the largest the time a random walker need to escape from it.

K-path Clustering



$$\begin{pmatrix}
5 & 2 & 1 & 0 & 2 & \mathbf{3} & \mathbf{3} & 0 & 0 & \mathbf{4} & 0 & 0 \\
2 & 4 & \mathbf{3} & 1 & \mathbf{3} & 1 & 2 & 1 & 0 & 1 & 0 & 0 \\
1 & \mathbf{3} & 4 & 2 & \mathbf{3} & 0 & 1 & 2 & 1 & 0 & 1 & 0 \\
0 & 1 & 2 & 5 & 2 & 0 & 0 & \mathbf{4} & \mathbf{4} & 0 & \mathbf{4} & 2 \\
2 & \mathbf{3} & \mathbf{3} & 2 & 5 & 0 & 2 & 2 & 1 & 1 & 1 & 0 \\
\mathbf{3} & 1 & 0 & 0 & 0 & 3 & 2 & 0 & 0 & \mathbf{3} & 0 & 0 \\
\mathbf{3} & 2 & 1 & 0 & 2 & 2 & 4 & 1 & 0 & \mathbf{3} & 0 & 0 \\
0 & 1 & 2 & \mathbf{4} & 2 & 0 & 1 & 5 & \mathbf{4} & 0 & \mathbf{4} & 2 \\
0 & 0 & 1 & \mathbf{4} & 1 & 0 & 0 & \mathbf{4} & 5 & 0 & \mathbf{5} & \mathbf{3} \\
\mathbf{4} & 1 & 0 & 0 & 1 & \mathbf{3} & \mathbf{3} & 0 & 0 & 4 & 0 & 0 \\
0 & 0 & 1 & \mathbf{4} & 1 & 0 & 0 & \mathbf{4} & \mathbf{5} & 0 & 5 & \mathbf{3} \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & \mathbf{3} & 0 & \mathbf{3} & 3
\end{pmatrix}$$

Matrix manipulation: $(N+I)^2$

Markov Clustering

- Expansion: Through matrix manipulation (power), one obtains a matrix for a n-steps connection.

- Inflation: Enhance intercluster passages by raising the elements to a certain power and then normalize

Markov Clustering Algorithm

- Iteratively running two operators
  - Inflation:

$$(T_r M)_{ij} = \frac{M_{ij}^r}{\sum_i M_{ij}^r}$$

Column normalization

  - Expansion:

$$\mathrm{Expand}(M) = M^k$$

MCL Running

$$\begin{pmatrix}
0.380 & 0.087 & 0.027 & -- & 0.077 & 0.295 & 0.201 & -- & -- & 0.320 & -- & -- \\
0.047 & 0.347 & 0.210 & 0.017 & 0.150 & 0.019 & 0.066 & 0.012 & -- & 0.012 & -- & -- \\
0.014 & 0.210 & 0.347 & 0.056 & 0.150 & -- & 0.016 & 0.046 & 0.009 & -- & 0.009 & -- \\
-- & 0.027 & 0.087 & 0.302 & 0.062 & -- & -- & 0.184 & 0.143 & -- & 0.143 & 0.083 \\
0.058 & 0.210 & 0.210 & 0.056 & 0.406 & -- & 0.083 & 0.046 & 0.009 & 0.019 & 0.009 & -- \\
0.142 & 0.017 & -- & -- & -- & 0.295 & 0.083 & -- & -- & 0.184 & -- & -- \\
0.113 & 0.069 & 0.017 & -- & 0.062 & 0.097 & 0.333 & 0.012 & -- & 0.147 & -- & -- \\
-- & 0.017 & 0.069 & 0.175 & 0.049 & -- & 0.016 & 0.287 & 0.143 & -- & 0.143 & 0.083 \\
-- & -- & 0.017 & 0.175 & 0.012 & -- & -- & 0.184 & 0.288 & -- & 0.288 & 0.278 \\
0.246 & 0.017 & -- & -- & 0.019 & 0.295 & 0.201 & -- & -- & 0.320 & -- & -- \\
-- & -- & 0.017 & 0.175 & 0.012 & -- & -- & 0.184 & 0.288 & -- & 0.288 & 0.278 \\
-- & -- & -- & 0.044 & -- & -- & -- & 0.046 & 0.120 & -- & 0.120 & 0.278
\end{pmatrix}$$

$\Gamma_2 M^2$, M defined in Figure 8

MCL Running

$$\begin{pmatrix}
0.448 & 0.080 & 0.023 & -- & 0.068 & 0.426 & 0.359 & -- & -- & 0.432 & -- & -- \\
0.018 & 0.285 & 0.228 & 0.007 & 0.176 & 0.006 & 0.033 & 0.005 & -- & 0.007 & -- & -- \\
0.005 & 0.223 & 0.290 & 0.022 & 0.173 & -- & 0.010 & 0.017 & 0.003 & 0.001 & 0.003 & 0.001 \\
-- & 0.018 & 0.059 & 0.222 & 0.040 & -- & 0.001 & 0.187 & 0.139 & -- & 0.139 & 0.099 \\
0.027 & 0.312 & 0.314 & 0.028 & 0.439 & 0.005 & 0.054 & 0.022 & 0.003 & 0.010 & 0.003 & 0.001 \\
0.116 & 0.007 & 0.001 & -- & 0.004 & 0.157 & 0.085 & -- & -- & 0.131 & -- & -- \\
0.096 & 0.040 & 0.013 & -- & 0.037 & 0.083 & 0.197 & 0.001 & -- & 0.104 & -- & -- \\
-- & 0.012 & 0.042 & 0.172 & 0.029 & -- & 0.002 & 0.198 & 0.133 & -- & 0.133 & 0.096 \\
-- & 0.001 & 0.015 & 0.256 & 0.009 & -- & -- & 0.266 & 0.326 & -- & 0.326 & 0.346 \\
0.290 & 0.021 & 0.002 & -- & 0.017 & 0.323 & 0.260 & -- & -- & 0.316 & -- & -- \\
-- & 0.001 & 0.015 & 0.256 & 0.009 & -- & -- & 0.266 & 0.326 & -- & 0.326 & 0.346 \\
-- & -- & 0.001 & 0.037 & 0.001 & -- & -- & 0.039 & 0.069 & -- & 0.069 & 0.112
\end{pmatrix}$$

$$\Gamma_2(\Gamma_2 M^2 \cdot \Gamma_2 M^2)$$

MCL Running

$$\begin{pmatrix}
0.807 & 0.040 & 0.015 & -- & 0.034 & 0.807 & 0.807 & -- & -- & 0.807 & -- & -- \\
-- & 0.090 & 0.092 & -- & 0.088 & -- & -- & -- & -- & -- & -- & -- \\
-- & 0.085 & 0.088 & -- & 0.084 & -- & -- & -- & -- & -- & -- & -- \\
-- & 0.001 & 0.001 & 0.032 & 0.001 & -- & -- & 0.032 & 0.031 & -- & 0.031 & 0.031 \\
-- & 0.777 & 0.798 & -- & 0.786 & -- & 0.001 & -- & -- & -- & -- & -- \\
0.005 & -- & -- & -- & -- & 0.005 & 0.005 & -- & -- & 0.005 & -- & -- \\
0.003 & 0.001 & -- & -- & 0.001 & 0.003 & 0.003 & -- & -- & 0.003 & -- & -- \\
-- & -- & 0.001 & 0.024 & -- & -- & -- & 0.024 & 0.024 & -- & 0.024 & 0.024 \\
-- & -- & 0.002 & 0.472 & 0.001 & -- & -- & 0.472 & 0.472 & -- & 0.472 & 0.472 \\
0.185 & 0.005 & 0.001 & -- & 0.004 & 0.185 & 0.184 & -- & -- & 0.185 & -- & -- \\
-- & -- & 0.002 & 0.472 & 0.001 & -- & -- & 0.472 & 0.472 & -- & 0.472 & 0.472 \\
-- & -- & -- & 0.001 & -- & -- & -- & 0.001 & 0.001 & -- & 0.001 & --
\end{pmatrix}$$

$(\Gamma_2 \circ Squaring)$ iterated four times on M

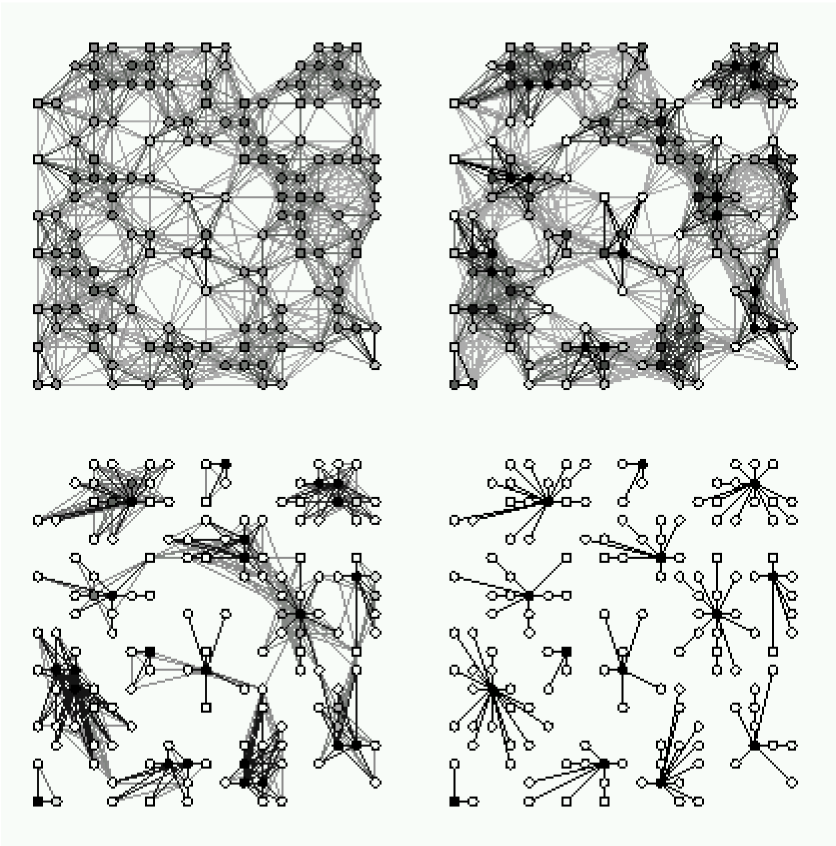
MCL Running

$$
\begin{pmatrix}
1.000 & -- & -- & -- & -- & 1.000 & 1.000 & -- & -- & 1.000 & -- & -- \\
-- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
-- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
-- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
-- & 1.000 & 1.000 & -- & 1.000 & -- & -- & -- & -- & -- & -- & -- \\
-- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
-- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
-- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
-- & -- & -- & 0.500 & -- & -- & -- & 0.500 & 0.500 & -- & 0.500 & 0.500 \\
-- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
-- & -- & -- & 0.500 & -- & -- & -- & 0.500 & 0.500 & -- & 0.500 & 0.500 \\
-- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & --
\end{pmatrix}
$$

M_{mcl}^{∞}

A Heuristic for MCL

We take a random walk on the graph described by the similarity matrix



After each step we weaken the links between distant nodes and strengthen the links between nearby nodes

Graphic from van Dongen, 2000

我们的一个扩展

- Diffusion kernel可以定义网络上的节点之间的Global影响。
- 而MCL算法将在同一个模块内的关联加强，而将不同模块之间的联系减弱。
- 如何将两者融合起来？

Naifang Su, Lin Wang, Yufu Wang, Minping Qian and Minghua Deng. Prediction of Protein Functions from Protein-Protein Interaction Data Based on a New Measure of Network Betweenness. Proceeding of iCBBE 2010.

我们的一个扩展

- 定义矩阵

$$K = I + \frac{\Gamma_{f(1)}(\tau H)}{1!} + \cdots + \frac{\Gamma_{f(n)}(\tau H)^n}{n!} + \cdots$$

其中

$$(\Gamma_{f(n)} A)_{ij} = \frac{A_{ij}^{f(n)}}{\sum_{i=1}^{N} A_{ij}^{f(n)}}$$

- 最后的相似矩阵B: 对角上为0, 对K做列归一化矩阵

$$B_{ij} = \frac{K_{ij}}{\sum_{j=1}^{N} K_{ij}}$$

参数的选取

- 实际应用中，我们尝试选取

$$f(n) = n^{\alpha}, \alpha = 0.3, 0.4, 0.5, 0.6$$
$$f(n) = log_k n, k = 2, 3, 4, 5, 6.$$

- 在我们的交叉验证中，$f(n) = \log_5 n$ 得到了最大的AUC.

References

- Newman MEJ, Modularity and community structure in network. PNAS 103: 8577-8582, 2006.
- Hofman, JM and Wiggins, CH. Bayesian approach to network modularity. Physical Review Letters 100:258701, 2008.
- S. van Dongen. A cluster algorithm for graphs. CWI Report INS-R0010, 2000.
- 侯琳, 邓明华. 网络聚类算法及其生物信息学应用. 数学建模及其生物信息学应用. Vol.2(2): 9-14, 2013.
- CharlesW. Fox, Stephen J. Roberts. A tutorial on variational Bayesian inference. Artif Intell Rev (2012) 38:85–95.DOI 10.1007/s10462-011-9236-8
- C. M. Bishop. Pattern Recognition and Machine Learning. Springer. 2006. (第10章：变分近似)