Modularity-Maximizing Graph Communities via Mathematical Programming

Gaurav Agarwal and David Kempe

大綱

介紹兩個Modularity最大化的演算法

1. 轉換成Linear Programming

2. 轉換成Vector Programming

先備知識 Modularity

$$Q(\mathcal{C}) := \frac{1}{2m} \sum_{u,v} (a_{u,v} - \frac{d_u d_v}{2m}) \cdot \delta(\gamma(u), \gamma(v))$$

演算法 Intro

our approach is to aim for the best division at each level individually, requiring a partition into two clusters at each level. Clusters are recursively subdivided as long as an improvement is possible.

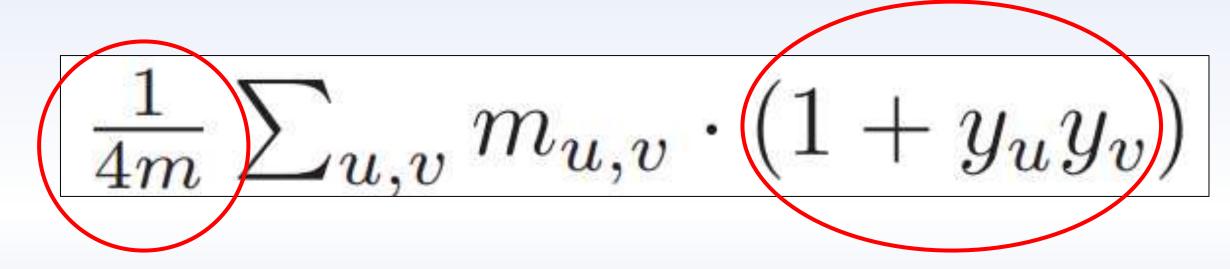
演算法 原理

Quadratic Programming is NP-complete. Hence, we use the standard technique of relaxing the QP to a corresponding Vector Program (VP), which in turn can be solved in polynomial time using semi-definite programming (SDP).

演算法 Step 1. Replacing

$$\frac{1}{2m} \sum_{u,v} (a_{u,v} - \frac{d_u d_v}{2m}) \cdot \delta(\gamma(u), \gamma(v))$$

替換成 $m_{u,v}$



For every vertex v, we have a variable y_v which is 1 or -1 depending on whether the vertex is in one or the other partition.

Maximize $\frac{1}{4m} \sum_{u,v} m_{u,v} \cdot (1 + y_u y_v)$ subject to $y_v^2 = 1$ for all v.

|演算法 QP→VP

To turn a quadratic program into a vector program, one replaces each variable y_v with a (n-dimensional) vector-valued variable y_v , and each product y_uy_v with the inner product $y_u\cdot y_v$

Maximize
$$\frac{1}{4m} \sum_{u,v} m_{u,v} \cdot (1 + y_u y_v)$$
 subject to $y_v^2 = 1$ for all v .

| 演算法 QP→VP

 y_v : (n-dimensional)

To turn a quadratic program into a vector program, one replaces each variable y_v with a (n-dimensional) vector-valued variable y_v , and each product y_uy_v with the inner product $y_u\cdot y_v$

Maximize
$$\frac{1}{4m} \sum_{u,v} m_{u,v} \cdot (1 + y_u \cdot y_v)$$
 subject to $y_v^2 = 1$ for all v .

演算法 solving SDP

We use the standard process

- 1. for transforming the VP formulation to the SDP formulation
- 2. for obtaining back the solution to the VP from the solution to SDP.

The result of solution of VP will be vectors y_V for all vertices v

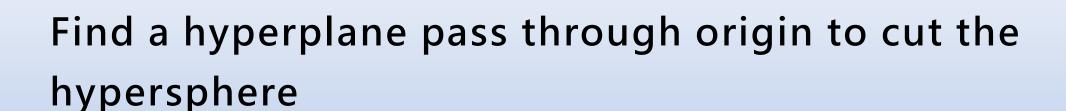
For solving the SDP problems in our experiments, we use a standard offthe-shelf solver CSDP

To obtain a partition from the node locations y_V , we use a rounding procedure for the Max-Cut problem.

Maximize
$$\frac{1}{2} \sum_{(u,v) \in E} (1 - y_u y_v)$$
 subject to $y_v^2 = 1$ for all v .

Maximize
$$\frac{1}{4m} \sum_{u,v} m_{u,v} \cdot (1 + y_u \cdot y_v)$$
 subject to $y_v^2 = 1$ for all v .

All y_V on n-dimensional hypersphere



1. All y_V on n-dimensional hypersphere

The constraint for all v ensures that all nodes are embedded at distance 1 from the origin.

e.g.
$$(1, 0, 0) \cdot (3/5, 4/5, 0) \cdot (\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3})$$

演算法 Step 2. Rounding Maximize
$$\frac{1}{2}\sum_{(u,v)\in E}(1-y_uy_v)$$
 subject to $y_v^2=1$ for all v .

2. Find a hyperplane pass through origin to cut the hypersphere

Find vector S, which is an n-dimensional vector, each of whose components is an independent N (0, 1) Gaussian.

→ choose random hyperplanes and retain the best resulting partition.

$$S := \{ v \mid \mathbf{y}_v \cdot \mathbf{s} \ge 0 \} \text{ and } \overline{S} := \{ v \mid \mathbf{y}_v \cdot \mathbf{s} < 0 \}$$

After this 2 steps, we obtain two clusters C' and C", then we calculate the modularity. The modularity Q increases by:

$$\Delta Q(C) = \frac{1}{m} \left(\frac{\left(\sum_{v \in C'} d_v\right)\left(\sum_{u \in C''} d_u\right)}{2m} - |e(C', C'')| \right)$$

Algorithm 2 Hierarchical Clustering

- 1: Let M be an empty Max-Heap.
- 2: Let C be a cluster containing all the vertices.
- 3: Use VP rounding to calculate (approximately) the maximum increase in modularity possible, $\Delta Q(C)$, achievable by dividing C into two partitions.
- 4: Add $(C, \Delta Q(C))$ to M.
- 5: while the head element in M has $\Delta Q(C) > 0$ do
- 6: Let C be the head of M.
- 7: Use VP rounding to split C into two partitions C', C'', and calculate $\Delta Q(C'), \Delta Q(C'')$.
- 8: Remove C from M.
- 9: Add $(C', \Delta Q(C')), (C'', \Delta Q(C''))$ to M.
- 10: end while
- 11: Output as the final partitioning all the partitions remaining in the heap M, as well as the hierarchy produced.