# Course project «Optimization approaches to community detection»

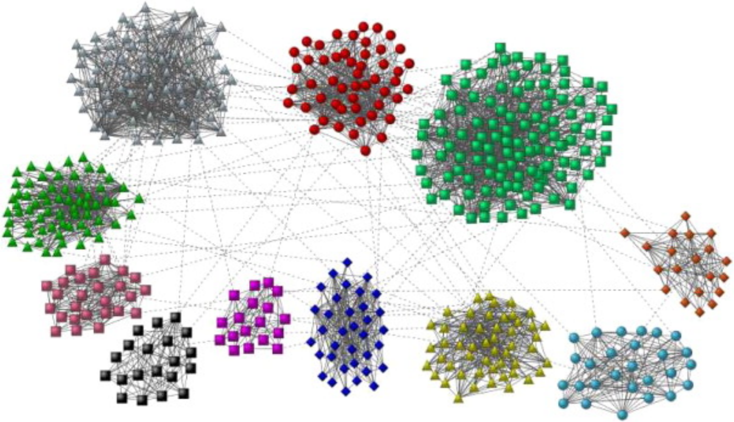
Marina Danilova, Alexander Podkopaev, Nikita Puchkin, Igor Silin

Skolkovo institute of science and technology

December 16, 2016

# Plan

# Example



The goal of community detection is to find **partition** of nodes into **non-overlapping** clusters.

# Notations

## Graph

- **Undirected unweighted** graphs **without loops** with $n$ nodes and $m$ edges.
- The nodes are enumerated as $\{1, ..., n\}$.
- Graph is given by its $n \times n$ adjacency matrix $A$.
- Degree of the node $i$ is $d_i$.

## Clusters

- The number of clusters is $k$.
- The clusters are denoted as $\mathcal{C}_1, ..., \mathcal{C}_k$.
- Cluster sizes are $|\mathcal{C}_1|, ..., |\mathcal{C}_k|$.
- Labeling $z$: $z(i)$ is the cluster containing node $i$, i.e. $i \in \mathcal{C}_{z(i)}$.

# Modularity-based method

## Modularity

- Fraction of edges which lie within communities:

$$\frac{1}{2m} \sum_{i,j=1}^{n} a_{ij} \cdot \mathbb{1}\{z(i) = z(j)\}.$$

- Expected fraction of edges which lie within communities (for some probabilistic model):

$$\sum_{i,j=1}^{n} \frac{d_i}{2m} \frac{d_j}{2m} \cdot \mathbb{1}\{z(i) = z(j)\}.$$

- **Modularity** is the difference between two previous fractions:

$$Q(z) = \frac{1}{2m} \sum_{i,j=1}^{n} \left( a_{ij} - \frac{d_i \cdot d_j}{2m} \right) \cdot \mathbb{1}\{z(i) = z(j)\}.$$

# Modularity-based method

## Formulating an optimization problem

$$\textit{maximize} \quad Q$$
$$s.t. \text{ all possible labelings } z$$

## Greedy algorithm

1. Initially every node forms its own cluster: $\mathcal{C}_i = \{i\}$ and $Q = 0$.
2. Iteratively:
   - choose two current clusters, joining of which gives maximal gain $\Delta Q$ of modularity.
   - unite these two current clusters into one and recalculate $Q := Q + \Delta Q$.
3. Pick the clustering with maximal $Q$ from partitions that we had during the iterations.

# Spectral method

## Measuring sizes of clusters

Consider the following two ways of measuring size of clusters:

- $|\mathcal{C}_i| = \{\text{number of vertices in } \mathcal{C}_i\}$
- $vol(\mathcal{C}_i) = \sum_{i \in \mathcal{C}_i} d_i$

## MinCut

Define $W(\mathcal{C}_p, \mathcal{C}_q) := \sum_{i \in \mathcal{C}_p, j \in C_q} a_{ij}$. Then MinCut problem is:

$$cut(\mathcal{C}_1, \ldots, \mathcal{C}_k) = \frac{1}{2} \sum_{i=1}^{k} W(\mathcal{C}_i, \overline{\mathcal{C}_i}) \to \min_{\mathcal{C}_1, \ldots, \mathcal{C}_k}$$

# Spectral method

## RatioCut and Normalized Cut

The MinCut solution separates one individual vertex from the rest. The following objectives are considered:

$$RatioCut(\mathcal{C}_1, \dots, \mathcal{C}_k) = \sum_{i=1}^{k} \frac{cut(\mathcal{C}_i, \overline{\mathcal{C}_i})}{|\mathcal{C}_i|} \to \min_{\mathcal{C}_1, \dots, \mathcal{C}_k}$$

$$NormCut(\mathcal{C}_1, \dots, \mathcal{C}_k) = \sum_{i=1}^{k} \frac{cut(\mathcal{C}_i, \overline{\mathcal{C}_i})}{vol(\mathcal{C}_i)} \to \min_{\mathcal{C}_1, \dots, \mathcal{C}_k}$$

Balancing conditions lead to NP-hard problem. Spectral clustering is a way to solve relaxed versions of those problems

# Spectral method

## Types of Laplacians

- Unnormilized Laplacian: $L = D - A$
- Symmetric Laplacian: $L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
- Random walk Laplacian: $L_{rw} = D^{-1} L = I - D^{-1} A$

## Idea

- Solving relaxed problem is equivalent to considering eigenvectors corresponding to $k$ smallest eigenvalues of Laplacian that describe cluster properties of given graph

# Semidefinite relaxation method

## Clustering matrix

- Clustering matrix $X$ of size $n \times n$ with $x_{ij} = \mathbb{1}\{z(i) = z(j)\}$.
- Space of all matices with such structure is $\mathcal{X}$.

## Formulating an optimization problem

Likelihood for special case of stochstic block model is

$$\mathcal{L}(X) = trace(AX).$$

Maximum likelihood method:

$$\begin{aligned} &maximize \;\; \mathcal{L}(X) \\ &s.t. \;\; X \in \mathcal{X} \end{aligned}$$

NP-hard combinatorial optimization problem $\Rightarrow$ relaxations!

# Semidefinite relaxation method

## Semidefinite relaxation

We just relax $X \in \mathcal{X}$ and get semidefinite program:

$$\begin{aligned} & \textit{maximize} \quad \textit{trace}(AX) \\ & s.t. \quad X \text{ is positive semi-definite}, \\ & \qquad\ \ X \geq 0, \\ & \qquad\ \ \textit{diagonal}(X) = e, \\ & \qquad\ \ Xe = \frac{n}{k}e, \end{aligned}$$

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

This problem can be solved with SDP solvers implemented in cvx.

# Natural conjugate gradients method

- Model parametrized by

  $z(i) \sim \text{Poly}(\pi), \quad i = \overline{1, n}$

  $P = \|p_{ij}\|_{i,j=\overline{1,k}}$ - probabilities of inter-cluster edges occurrence
- Bayesian approach:

$$\pi \sim \text{Dirichlet}(\alpha)$$
$$p_{ii} \sim \text{Beta}(\beta), \quad i = \overline{1, k}$$
$$p_{ij} \ll 1, \quad \forall i \neq j$$

- $p(z, \pi, P|A)$ - true posterior with observed adjacency matrix $A$
- $\mathcal{Q}$ - family of feasible distributions

# Natural conjugate gradients method

## Formulating an optimization problem

$$\mathcal{L}(q) \equiv -\mathrm{KL}\left(q \| p(Z, \pi, P | A)\right) \longrightarrow \max_{q \in \mathcal{Q}}$$

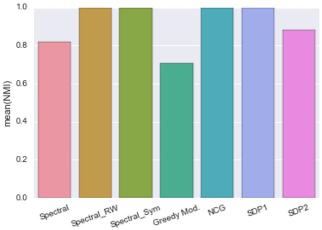
- The problem can be reduced to unconstrained optimization:

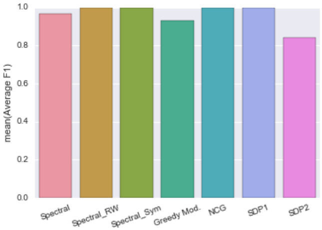
$$\mathcal{L} = \mathcal{L}(\theta) \longrightarrow \max, \quad \theta \text{ - } n \times (k-1) \text{ matrix}$$

- Use conjugate gradients method in a statistical manifold
- Metrics is defined by a matrix

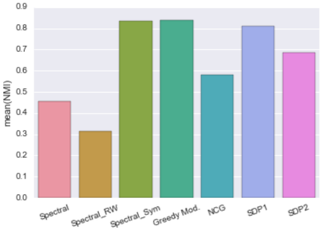
$$\mathcal{I}(\theta) - \text{Fischer information}$$

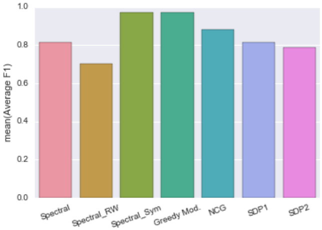
# Artificial graph:




# Real-world graph:



Thanks for your attention!