Check list

- [ ] Microphone turned on
- [ ] Zoom room open
- [ ] Recording on
- [ ] Mouse cursor visible
- [ ] Sound Volume on

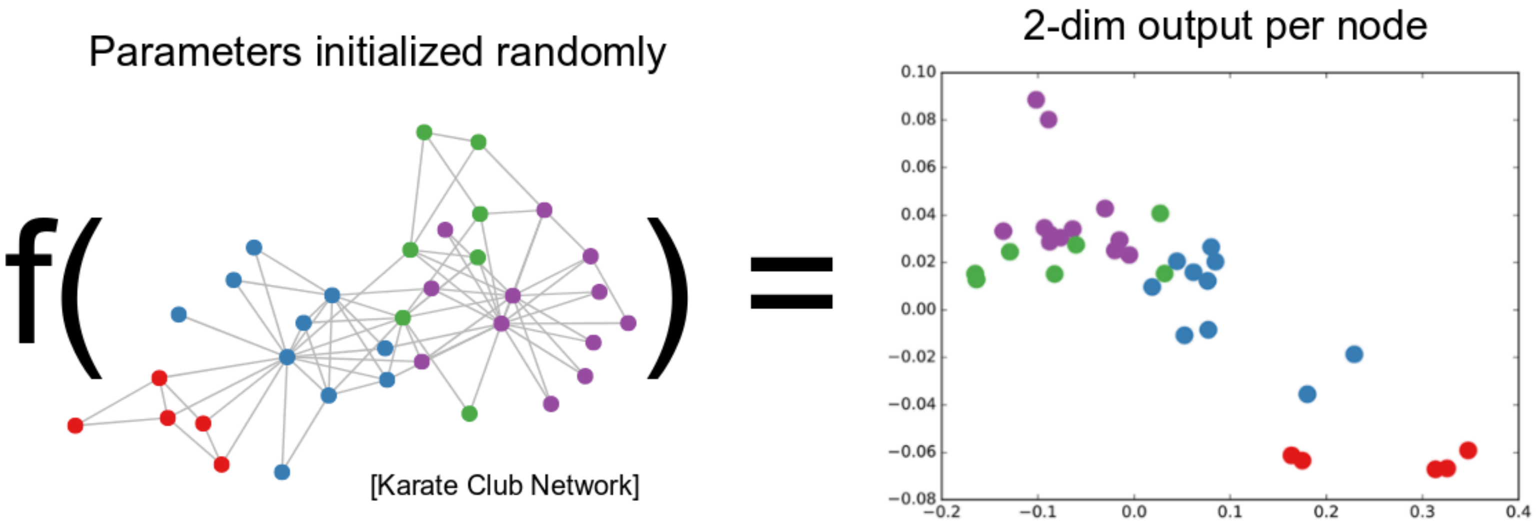
Advanced Topics in Network Science

Lecture 08: Network Embedding

Sadamori Kojaku

Network Embedding 🌐

- **Task**: Embed a network into a low-dimensional vector space
- **What to expect for a good embedding**:
    - Should preserve the network structure
    - Can recover the network structure from the embedding with high accuracy
    - Embedding should be low-dimensional
- **Question**: Think as many strategies as possible to produce such embeddings!



Pen and paper time!

Idea 1: Network reconstruction

Compress adjacency matrix A into low-dim representation U that minimizes **the reconstruction error**:

$$\min \frac{1}{2}||\mathbf{A} - \lambda \mathbf{U}\mathbf{U}^\top||_F^2$$

where

$$U = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_d \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times d}, \quad \mathbf{u}_i^\top \mathbf{u}_j = 0 \quad \forall i \neq j, \text{and} \quad \sum_i u_i^2 = 1$$

- **Q**: Solve for the 1d case where $U = [\mathbf{u}] \in \mathbb{R}^{n \times 1}$.
    - Hint: $||\mathbf{A} - \mathbf{u}\mathbf{u}^T||_F^2 = \sum_{i,j}(A_{ij} - u_i u_j)^2 = \mathbf{Tr}((\mathbf{A} - \mathbf{u}\mathbf{u}^T)^\top(\mathbf{A} - \mathbf{u}\mathbf{u}^T))$.

$$\frac{1}{2}||\mathbf{A} - \lambda \mathbf{U}\mathbf{U}^\top||_F^2 = \frac{1}{2}\sum_i \sum_j (A_{ij} - \lambda u_i u_j)^2$$

This is a convex function of u_i. Meaning that the gradient is 0 at the minimum. Thus, by taking derivative with respect to u_i, we can find the optimal u_i by solving the following equation:

$$\frac{\partial}{\partial u_i}\frac{1}{2}||\mathbf{A} - \lambda \mathbf{U}\mathbf{U}^\top||_F^2 = 0$$

Q: Take derivative with respect to u_i and solve for u_i for the 1d case:

$$\frac{1}{2}||\mathbf{A} - \lambda\mathbf{U}\mathbf{U}^\top||_F^2 = \frac{1}{2}\sum_i\sum_j(A_{ij} - \lambda u_i u_j)^2$$

This is a convex function of u_i. Meaning that the gradient is 0 at the minimum. Thus, by taking derivative with respect to u_i, we can find the optimal u_i by solving the following equation:

$$\frac{\partial}{\partial u_i}\frac{1}{2}||\mathbf{A} - \lambda\mathbf{U}\mathbf{U}^\top||_F^2 = 0$$

Q: Take derivative with respect to u_i and solve for u_i for the 1d case:

$$\frac{1}{2}\sum_i\sum_j(A_{ij} - \lambda u_i u_j)^2 = -\frac{1}{2}\cdot \underbrace{2}_{\substack{\text{Because } A_{ij}-\lambda u_i u_j \\ \text{appears twice} \\ \text{in the sum}}} \times \sum_j(A_{ij} - \lambda u_i u_j)\cdot\lambda u_j$$

$$= -\lambda\sum_j\left(A_{ij}u_j\right) + \lambda^2 u_i$$

By setting the derivative to 0, we get:

$$\frac{\partial}{\partial u_i}\frac{1}{2}||\mathbf{A}-\lambda\mathbf{U}\mathbf{U}^\top||_F^2 = 0 \iff \lambda\sum_j (A_{ij}u_j) - \lambda^2 u_i = 0$$
$$\iff \sum_j (A_{ij}u_j) = \lambda u_i$$

or equivalently in matrix form,

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$$

This is an eigenvalue problem of $\mathbf{A}$!

Q: But we have many eigenvectors u_i that yields the gradient of 0. Which one should we choose?

Let us rewrite the reconstruction error in terms of the eigenvalues and eigenvectors of $\mathbf{A}$:

$$\frac{1}{2}||\mathbf{A} - \lambda \mathbf{u}\mathbf{u}^\top||_F^2 = \mathrm{Tr}((\mathbf{A} - \lambda \mathbf{u}\mathbf{u}^\top)^\top (\mathbf{A} - \lambda \mathbf{u}\mathbf{u}^\top))$$

where we have used the fact that $||\mathbf{M}||_F^2 = \mathrm{Tr}(\mathbf{M}^\top \mathbf{M})$. Expanding it gives

$$\mathrm{Tr}((\mathbf{A} - \lambda \mathbf{u}\mathbf{u}^\top)^\top (\mathbf{A} - \lambda \mathbf{u}\mathbf{u}^\top)) = \mathrm{Tr}(\mathbf{A}^\top \mathbf{A}) - 2\lambda \mathrm{Tr}(\mathbf{u}^\top \mathbf{A}\mathbf{u}) + \lambda^2 \mathrm{Tr}(\mathbf{u}^\top \mathbf{u}\mathbf{u}^\top \mathbf{u}))$$

Note that

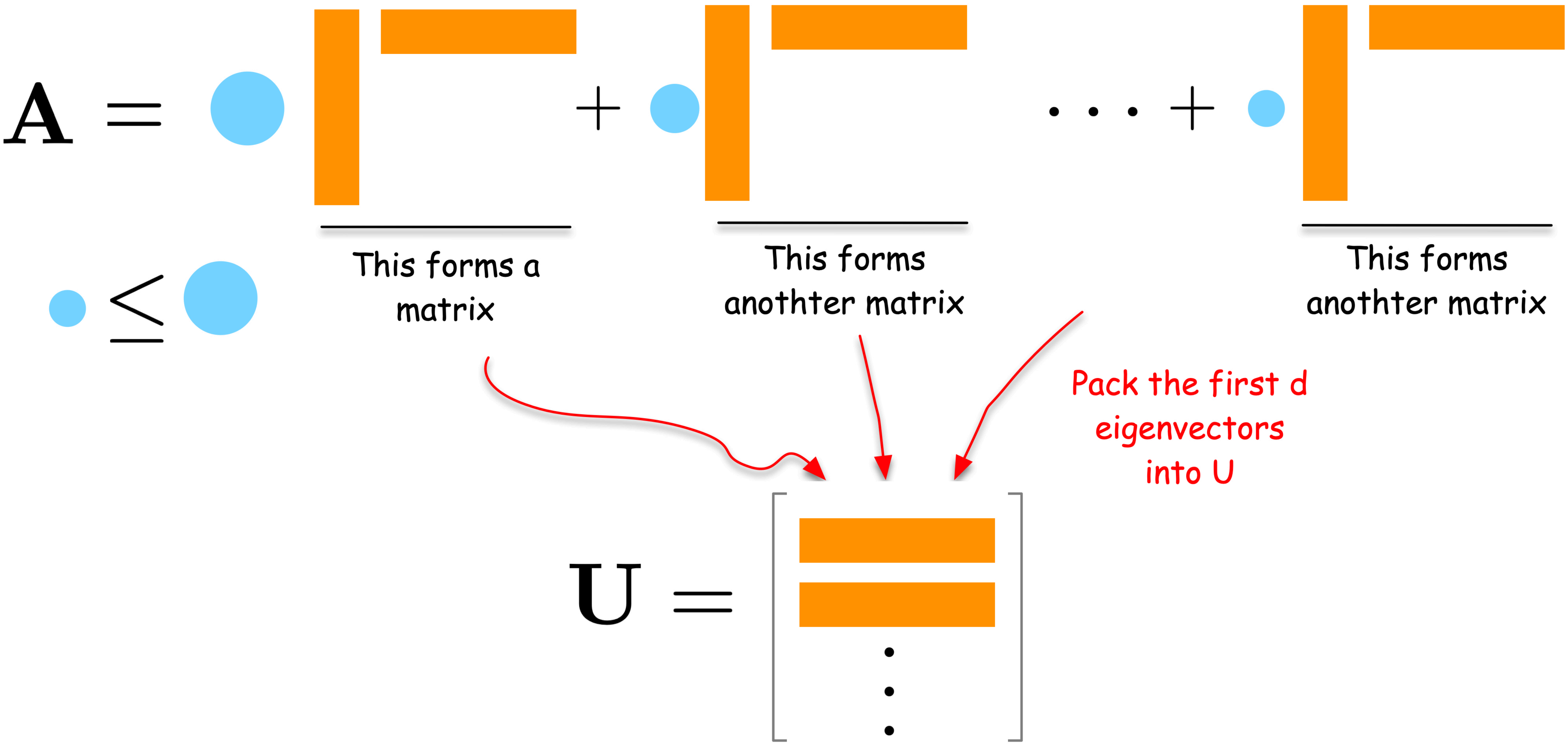
1. $\mathrm{Tr}(\mathbf{A}^\top \mathbf{A})$ is constant with respect to $\mathbf{U}$ and thus can be ignored in the optimization.
2. $\mathrm{Tr}(\mathbf{u}^\top \mathbf{u}\mathbf{u}^\top \mathbf{u})$ is also constant since $\mathbf{u}$ is a unit vector.
3. $\mathrm{Tr}(\mathbf{u}^\top \mathbf{A}\mathbf{u}) = \lambda$ because $\mathbf{u}$ is an eigenvector of $\mathbf{A}$ with eigenvalue λ.

Altogether, the reconstruction error is

$$\frac{1}{2}||\mathbf{A} - \lambda \mathbf{u}\mathbf{u}^\top||_F^2 = -\lambda^2 + \mathrm{const.}$$

It is minimized when λ is the largest eigenvalue. So the best u is the eigenvector corresponding to the largest eigenvalue.

What are the eigenvectors of a network?



This forms a matrix

This forms anothter matrix

This forms anothter matrix

Pack the first d eigenvectors into U

Solution for general d dimensional case

$$\frac{1}{2}||\mathbf{A} - \lambda\mathbf{U}\mathbf{U}^\top||_F^2 = -\mathrm{Tr}(\mathbf{U}^\top\mathbf{A}\mathbf{U}) + \text{const.}$$

The best $\mathbf{U}$ is given by the top d eigenvectors of $\mathbf{A}$ with the largest eigenvalues.

Fed up with the math? Let's see some examples!

- 🔗 Exercise notebook
- Complete only the first section.

Idea 2: Graph Cut:

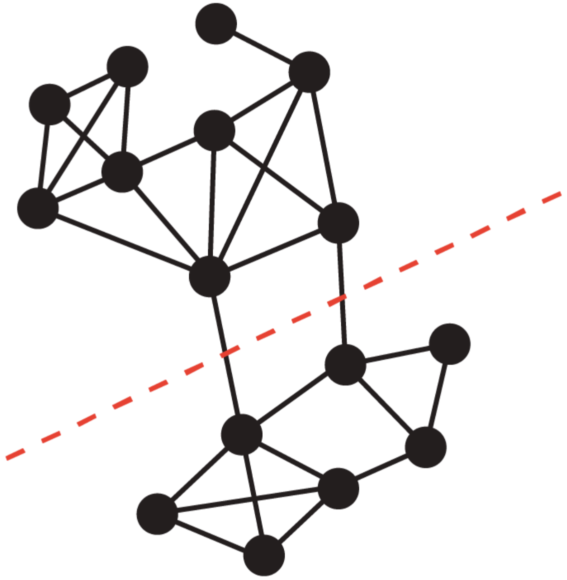
Let us consider a community detection task

What is a graph cut? 🤔

- 🎯 **Task**: Disconnect a graph into two components by cuttting the minimum number of edges

$$\min_{Q,S} \operatorname{cut}(Q,S) = \sum_{i \in Q} \sum_{j \in S} A_{ij}$$

- ❌ **Problem**: Can create unbalanced clusters of nodes



Ratio Cut

$$\mathrm{RatioCut}(Q, S) = \frac{\mathrm{cut}(Q, S)}{|Q|} + \frac{\mathrm{cut}(Q, S)}{|S|}$$

- ✨ **Benefits**:

    i. Balances cluster sizes

    ii. Avoids singleton clusters

    iii. More meaningful partitions

Optimization

Let us formulate the ratio cut problem as an optimization problem.

$$\mathrm{RatioCut}(Q,S) = \frac{\mathrm{cut}(Q,S)}{|Q|} + \frac{\mathrm{cut}(Q,S)}{|S|} = \sum_{i\in Q}\sum_{j\in S} A_{ij}\left(\frac{1}{|Q|}+\frac{1}{|S|}\right)$$

Let x_i be the indicator variable for $i \in Q$ or $i \in S$:

$$x_i = \begin{cases} \sqrt{\frac{|S|}{|Q|}} & \text{if } i \in Q \\ -\sqrt{\frac{|Q|}{|S|}} & \text{if } i \in S \end{cases}$$

Zero mean: $\sum_i x_i = \sqrt{\frac{|S|}{|Q|}}|Q| - \sqrt{\frac{|Q|}{|S|}}|S| = 0$

Normalization: $\sum_i x_i^2 = \frac{|S|}{|Q|}|Q| + \frac{|Q|}{|S|}|S| = N$

where $N = |Q| + |S|$ is the number of nodes.

Our goal: Express the ratio cut objective in terms of x_i and A_{ij}.

Consider distance between points x_i and x_j:

$$(x_i - x_j)^2 = \left(\sqrt{\frac{|S|}{|Q|}} + \sqrt{\frac{|Q|}{|S|}}\right)^2 = \frac{|S|^2 + |Q|^2 + 2|S||Q|}{|Q||S|} = \frac{(|S| + |Q|)^2}{|Q||S|} = \frac{N^2}{|Q||S|}$$

$$\begin{aligned}\mathrm{RatioCut}(Q, S) &= \sum_{i \in Q} \sum_{j \in S} A_{ij} \left(\frac{1}{|Q|} + \frac{1}{|S|}\right) = \sum_{i \in Q} \sum_{j \in S} A_{ij} \left(\frac{|Q| + |S|}{|Q||S|}\right) \\ &= \frac{1}{N} \sum_{i \in Q} \sum_{j \in S} A_{ij}(x_i - x_j)^2 \\ &= \frac{1}{2N} \underbrace{\sum_{i=1}^{N} \sum_{j=1}^{N}}_{\text{Sum over all node pairs.}} A_{ij}(x_i - x_j)^2\end{aligned}$$

(The last equality holds because $(x_i - x_j)^2 = 0$ for $i, j \in Q$ or $i, j \in S$.)

By expanding the square term, we get

$$\begin{aligned}
\mathrm{RatioCut}(Q,S) &= \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{N}A_{ij}(x_i-x_j)^2 = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{N}A_{ij}\left(x_i^2-2x_ix_j+x_j^2\right)\\
&= \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{N}A_{ij}\left(x_i^2+x_j^2\right)-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}A_{ij}x_ix_j\\
&= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}A_{ij}x_i^2-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}A_{ij}x_ix_j\\
&= \frac{1}{N}\sum_{i=1}^{N}x_i^2\underbrace{\sum_{j=1}^{N}A_{ij}}_{\text{degree } k_i}-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}A_{ij}x_ix_j\\
&= \frac{1}{N}\sum_{i=1}^{N}k_ix_i^2-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}A_{ij}x_ix_j
\end{aligned}$$

Cont.

$$\begin{aligned}\mathrm{RatioCut}(Q,S) &= \frac{1}{N}\sum_{i=1}^{N} k_i x_i^2 - \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij} x_i x_j \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} [k_i \mathbf{1}(i=j) - A_{ij}] x_i x_j \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} L_{ij} x_i x_j\end{aligned}$$

where L is the (combinatorial) Laplacian matrix given by

$$\mathbf{L} = \begin{bmatrix} k_1 & -A_{12} & \cdots & -A_{1n} \\ -A_{21} & k_2 & \cdots & -A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -A_{n1} & -A_{n2} & \cdots & k_n \end{bmatrix}$$

So the optimization problem is

$$\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{L} \mathbf{x}$$

subject to

$$\mathbf{x}\mathbf{1} = 0, \quad \mathbf{x}^\top \mathbf{x} = N \quad x_i \in \left\{ \sqrt{|S|/|Q|}, -\sqrt{|Q|/|S|} \right\}$$

This is an NP-hard problem 😿. But, there is a way to get a good suboptimal solution by using spectral embedding!

💫 **Core idea** 💫: Relax the discrete constraint by allowing x_i to be any real number. But keep the normalization constraint:

$$\mathbf{x}^\top \mathbf{1} = 0, \quad \mathbf{x}^\top \mathbf{x} = 1 \quad x_i \in [-1, 1]$$

($\mathbf{x}^\top \mathbf{x} = 1$ is effectively the same as $\mathbf{x}^\top \mathbf{x} = N$.)

Q: What is the solution to this optimization problem?

Consider the eigenvalue problem:

$$\mathbf{L}\mathbf{x} = \lambda\mathbf{x}$$

By applying $\mathbf{x}^\top$ to both sides, we get

$$\mathbf{x}^\top\mathbf{L}\mathbf{x} = \lambda\mathbf{x}^\top\mathbf{x} = \lambda$$

where we have used $\mathbf{x}^\top\mathbf{x} = 1$. The left-hand side is exactly the objective function we want to minimize!

Thus the solution is the eigenvector corresponding to the smallest eigenvalue 😉....

Consider the eigenvalue problem:

$$\mathbf{L}\mathbf{x} = \lambda \mathbf{x}$$

By applying $\mathbf{x}^\top$ to both sides, we get

$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \lambda \mathbf{x}^\top \mathbf{x} = \lambda$$

where we have used $\mathbf{x}^\top \mathbf{x} = 1$. The left-hand side is exactly the objective function we want to minimize!

~~Thus the solution is the eigenvector corresponding to the smallest eigenvalue~~ 😉.

The eigenvector corresponding to the smallest eigenvalue is parallel to the all-ones vector.

$$\mathbf{x}_1 = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^\top$$

which violates a constraint $\mathbf{x}^\top \mathbf{1} = 0$.

But the second smallest eigenvector is orthogonal to x_1.

- This is because the eigenvectors are orthogonal to each other.
- Thus, the solution is the eigenvector corresponding to **the second smallest eigenvalue**!
- The second, and third, and … smallest eigenvectors can be used to get a k-way partition of the network.

Code exercise

- 🔗 Exercise notebook
- Complete only the second section.

Group work 01

Let us derive another spectral embedding method based on the normalized cut.

$$\mathrm{Ncut}(Q, S) = \frac{\mathrm{cut}(Q, S)}{D(Q)} + \frac{\mathrm{cut}(Q, S)}{D(S)}$$

where $D(Q) = \sum_{i \in Q} k_i$ and $D(S) = \sum_{i \in S} k_i$ are the sum of the degrees of the nodes in Q and S, respectively.

Plot the 2D embedding of the karate club network based on the normalized cut. (Section 3 in the exercise notebook)

Group work 02

Let us derive another spectral embedding method based on the modularity.

$$J = \sum_{i=1}^{N} \sum_{j} \left[A_{ij} - \frac{k_i k_j}{2M} \right] \delta(c_i, c_j)$$

where k_i is the degree of node i and M is the total number of edges.

$$\delta(c_i, c_j) = \begin{cases} 1 & \text{if } c_i = c_j \\ 0 & \text{otherwise} \end{cases}$$

Express J in forms of $z^\top M z$ where z is a vector and M is a matrix. Derive the spectral embedding method based on the modularity, and plot the 2D embedding of the karate club network. (Section 4 in the exercise notebook)

Laplacian Eigenmap 🔄

- 📝 **Objective**: Position connected nodes close together
- 🧮 **Math formulation**:

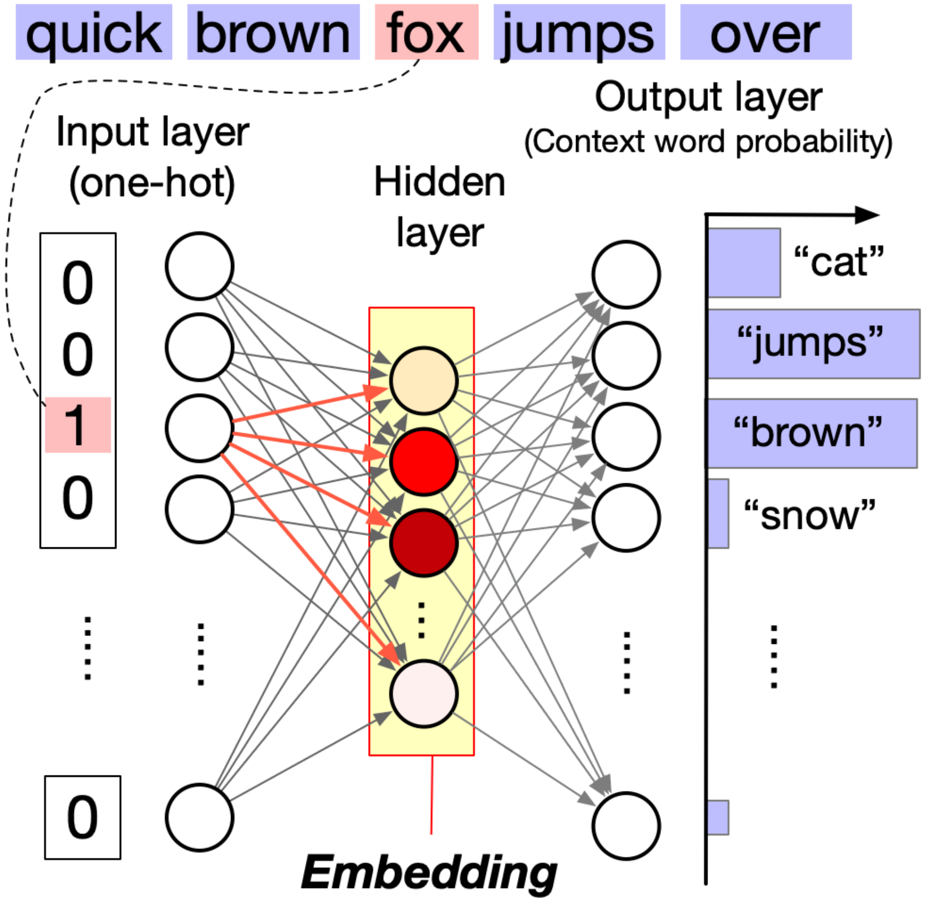
$$\min_{\mathbf{u}} \sum_{i,j} A_{ij} (\mathbf{u}_i - \mathbf{u}_j)^2$$

Q: What is the solution to this optimization problem?

Neural Embedding Methods

Word2Vec Architecture 🏗️

- 📚 **Input**: One-hot encoded word vector (vocab size N)
- 🧠 **Hidden Layer**: Dense representation (size d << N)
- 📤 **Output**: Probability distribution over vocab
- 🎯 **Training Objective**:
    - Predict context words from target (Skip-gram)
    - Or predict target from context (CBOW)
- 🔑 **Key Parameter**: Window size for context

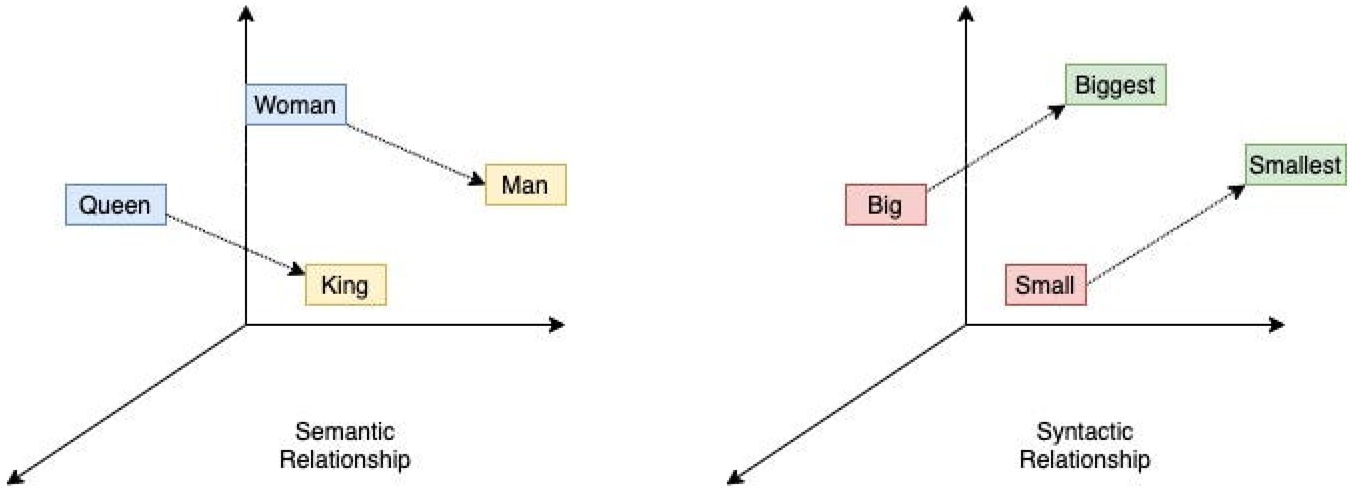


Pen and Paper exercise

Pen and Paper Exercise

What's special about Word2Vec?

With word2vec, words are represented as dense vectors, enabling us to explore their relationships using simple linear algebra. This is in contrast to traditional natural language processing (NLP) methods, such as bag-of-words and topic modeling, which represent words as discrete units or high-dimensional vectors.

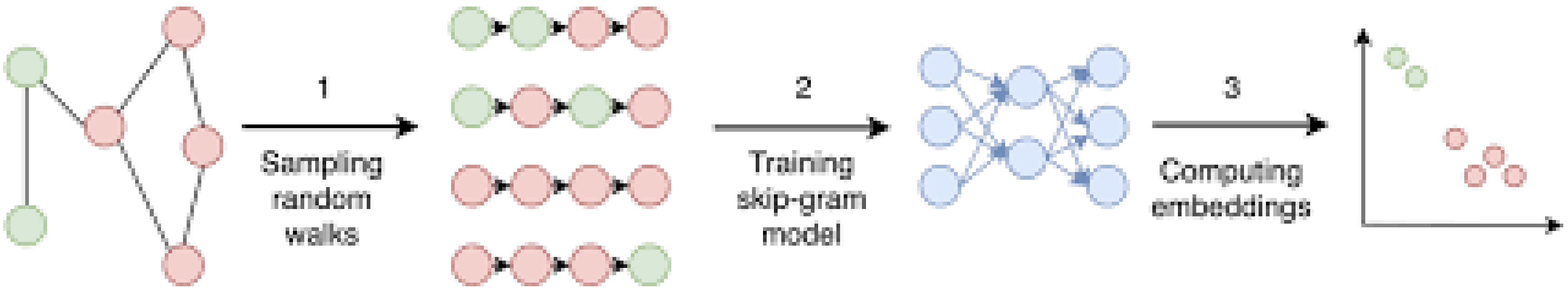


Hands-on exercise

- 🔗 Exercise notebook
- Complete only Section 5.

DeepWalk: Graph → Word2Vec 🚶

1. 🎲 **Random Walk Generation**:
2. 🎯 **Training Process**:
    - Generate k walks of length l per node
    - Feed walks to Word2Vec
    - Use skip-gram with hierarchical softmax



Phases of DeepWalk approach

Node2Vec: Sophisticated Random Walks 🎯

- 🎲 **Biased Random Walk**:

$$P(\text{next}|\text{current}, \text{previous}) \propto \begin{cases} \frac{1}{p} & \text{Return to previous} \\ 1 & \text{Move to neighbor} \\ \frac{1}{q} & \text{Explore further} \end{cases}$$

- 📊 **Parameters**:
  - p: Return parameter (lower = more backtracking)
  - q: Out-degree parameter (lower = more exploration)

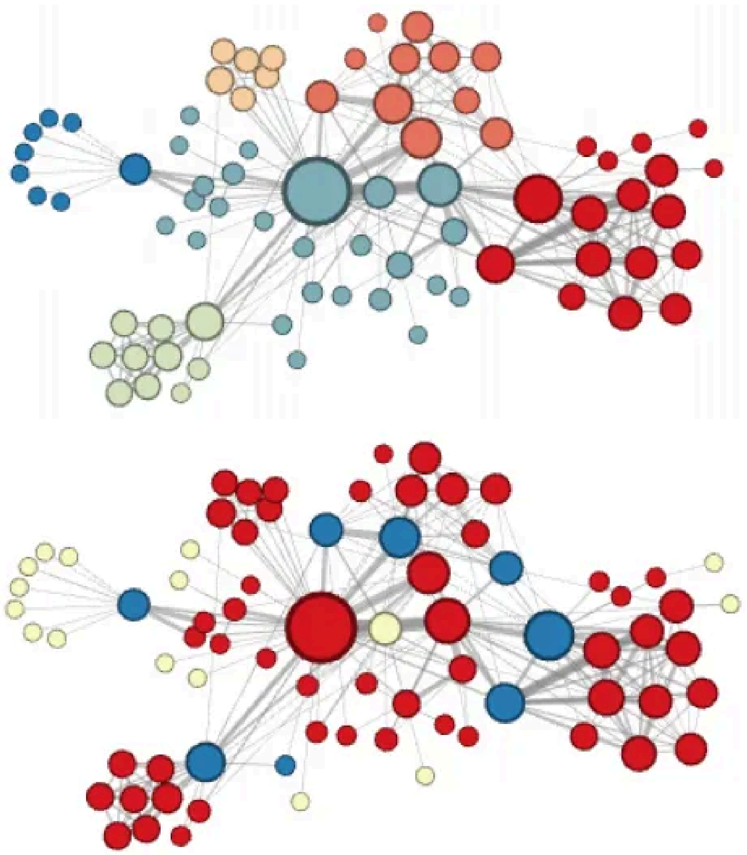


Figure 3: Complementary visualizations of Les Misérables coappearance network generated by *node2vec* with label colors reflecting homophily (top) and structural equivalence (bottom).

Hands-on exercise

- 🔗 Exercise notebook
- Complete only the third section.

Comparison: Spectral vs Neural 🤔

Aspect	Spectral	Neural
Theory	✅ Provably optimal for SBM	⚠️ Less theoretical understanding
Scale	❌ $O(n^2)$ computation	✅ Scales with edges
Space	✅ Euclidean	✅ Flexible (e.g., hyperbolic)
Implementation	✅ Simple	⚠️ Needs careful tuning

Software & Implementation 🛠️

- 🚀 **fastnode2vec**
  - Fast implementation but uses uniform negative sampling
- 🔧 **pytorch-geometric**
  - Full GNN suite with embedding support
  - Standard implementation quirks
- 📦 **gnn-tools / graphvec**
  - Research-focused implementations
  - Good for experimentation

Take-Home Messages 🎓

1. 📊 Spectral methods: mathematically elegant, limited by scale
2. 🧠 Neural methods: flexible, scalable, needs tuning
3. 🎯 Choice depends on:
   - Network size
   - Theoretical guarantees needed
   - Computational resources
   - Downstream task requirements