Graph Learning (SD212)

William Liaw

22 juin 2024

Contents

Preliminary **3**
0.1 Introduction . . . 3
0.2 Sparse matrix . . . 3
0.3 Graphs as Sparse Matrices . . . 4

1 Graphs Structure **4**
1.1 Friendship paradox . . . 4
1.2 Scale-free property . . . 4
1.3 Small-world property . . . 5
1.4 Clustering property . . . 5

2 Page Rank **5**
2.1 Random walk . . . 5
2.2 PageRank . . . 6
2.3 Personalized PageRank . . . 6
2.4 Case of bipartite graphs . . . 6
2.5 Applications . . . 7

3 Graph Clustering **7**
3.1 Modularity . . . 7
3.2 The Louvain algorithm . . . 8
3.3 Cluster strengths . . . 8
3.4 Resolution . . . 8
3.5 Extensions . . . 9

4 Hierarchical Clustering **9**
4.1 Notion of dendrogram . . . 9
4.2 Divisive algorithm . . . 9
4.3 Agglomerative algorithm . . . 9
4.4 Extensions . . . 10

5 Heat Diffusion **10**
5.1 Heat diffusion . . . 10
5.2 Dirichlet problem . . . 11
5.3 Applications . . . 11
5.4 Extensions . . . 11

6 Spectral Embedding **11**
6.1 Laplacian matrix . . . 11
6.2 Transition matrix . . . 12
6.3 Spectral embedding . . . 12
6.4 Algorithms . . . 12
6.5 Extensions . . . 13

7 Graph Neural Networks **13**
7.1 Background on neural networks 13
7.2 Graph neural networks 14
7.3 Variants 15

Preliminary

0.1 Introduction

- Graph data
  - Infrastructure
  - Communication
  - Information
  - Social networks
  - Marketing
  - Text analysis
  - Health
- Graph mining:
  - Objective: understand learn, exploit the graph structure
  - Typical applications:
    - Information retrieval
    - Content recommendation
    - Link prediction
    - Anomaly detection
    - Label propagation
    - Data visualization
- Nodes: n
- Edges: m
- Density: $\frac{m}{\binom{n}{2}} \approx \frac{m}{n^2}$
- Golden rule of Python programming: vectorise

0.2 Sparse matrix

- Real graphs are sparse (low density)
  - Adjacency matrix entries are essentially zeroes

$$\begin{bmatrix} 5 & 6 & 9 & & 2 & 2 & & 4 \\ 5 & & & & 7 & & & \\ & & & & & & & \\ & 5 & & & & 3 & & \\ 6 & & & & & & 1 & 3 \\ & & 5 & & & & 9 & \end{bmatrix}$$

- shape $= (6, 8)$
- Coordinate format

$$\begin{aligned} \text{data} &= (5, 6, 9, 2, 2, 4, 5, 7, 5, 3, 6, 1, 3, 5, 9) \\ \text{row} &= (0, 0, 0, 0, 0, 0, 1, 1, 3, 3, 4, 4, 4, 5, 5) \\ \text{col} &= (0, 1, 2, 4, 5, 7, 0, 4, 1, 5, 0, 6, 7, 2, 6) \end{aligned}$$

- Compressed Sparse Row (CSR)

$$\begin{aligned} \text{data} &= (5, 6, 9, 2, 2, 4, 5, 7, 5, 3, 6, 1, 3, 5, 9) \\ \text{indices} &= (0, 1, 2, 4, 5, 7, 0, 4, 1, 5, 0, 6, 7, 2, 6) \\ \text{indptr} &= (0, 6, 8, 8, 10, 13, 15) \end{aligned}$$

- Compressed Sparse Columns (CSC)

$$\begin{aligned} \text{data} &= (5, 6, 9, 2, 2, 4, 5, 7, 5, 3, 6, 1, 3, 5, 9) \\ \text{indices} &= (0, 1, 4, 0, 3, 0, 5, 0, 1, 0, 3, 4, 5, 0, 4) \\ \text{indptr} &= (0, 3, 5, 7, 7, 9, 11, 13, 15) \end{aligned}$$

- List of lists

$$\text{data} = [[5,6,9,2,2,4],[5,7],[],[5,3],[6,1,3],[5,9]]$$
$$\text{rows} = [[0,1,2,4,5,7],[0,4],[],[1,5],[0,6,7],[2,6]]$$

- Use cases

Fast…	COO	CSR	CSC	LIL
Dot product		X	X	
Arithmetic		X	X	
Row slicing		X		
Column slicing			X	
Modification				X
Loading	X			

0.3 Graphs as Sparse Matrices

- Adjacency matrix: $1 \Leftrightarrow$ node i connected to node j
- Weighted graph: adjacency matrix with $w \Leftrightarrow$ node i connected to node j with weight w
- Bipartite graph:
  - B adjacency matrix (rectangular matrix): node i from first group connected to node j of the second group
  - Adjacency matrix: $A = \begin{pmatrix} \mathbf{0} & B \\ B^T & \mathbf{0} \end{pmatrix}$

1 Graphs Structure

1.1 Friendship paradox

- Consider a graph of n nodes m edges, undirected, without self-loops: $X \in 1, \dots, n$
- Node sampling: $P(X = j) = \frac{1}{n}$
  - Degree: $P(D = k) = \sum_j P(X = j) 1_{d_j = k} = \frac{1}{n} \sum_{j=1}^{n} 1_{d_j = k}$
    - Expectation: $\mathbb{E}(D) = \sum_k k P(D = k) = \frac{2m}{n}$
- Edge sampling: $P'(X = j) = \frac{1}{2m} \sum_i A_{ij} = \frac{d_j}{2m}$
  - Degree: $P'(D = k) = \sum_j P'(X = j) 1_{d_j = k} = \frac{k}{2m} \sum_j 1_{d_j = k}$
    - Expectation: $\mathbb{E}'(D) = \frac{\mathbb{E}(D^2)}{\mathbb{E}(D)} \geq \mathbb{E}(D)$ with equality only if the graph is regular
- Neighbor sampling: $P''(X = j) = \frac{1}{n} \sum_i P_{ij}$
  - Probability of choosing neighbor j from node i: $P_{ij} = \frac{A_{ij}}{d_i}$
  - Degree: $P''(D = k) = \sum_j P''(X = j) 1_{d_j = k} = \frac{1}{n} \sum_{i,j} 1_{d_j = k} P_{ij}$
    - Expectation (friendship paradox): $\mathbb{E}''(D) = \frac{1}{2n} \sum_{i,j} (\frac{d_i}{d_j} + \frac{d_j}{d_i}) A_{ij} \geq \mathbb{E}(D)$

1.2 Scale-free property

- Degrees typically have a power law, or Pareto: $P(D \geq k) \approx \left(\frac{k_{\min}}{k}\right)^{\alpha}$, minimum degree $k_{\min}$, power exponent $\alpha \in (1, 2]$ typically
  - Linear in log-scale
  - $\mathbb{E}(D) = \frac{\alpha}{\alpha - 1} k_{\min}, \text{var}(D) = +\infty \therefore$ the average degree is not informative
- Random graph: n nodes connected with probability p. Thus, $A_{ij} \sim \text{Bernoulli}(p)$ for $i < j$
  - $D \sim B(n - 1, p)$

  - $n >> 1, np \to \lambda, D \approx P(\lambda)$
  - $P''(D = k) \propto kP(D = k) \propto P(D = k|D \geq 1)$
  - $\mathbb{E}''(D) = \mathbb{E}(D) + 1$
- Power law graphs
  - $\mathbb{E}''(D) = \mathbb{E}(1 + \text{cv}^2)$
    - Coefficient of variation: $\text{cv} = \frac{\sigma(D)}{\mathbb{E}(D)} = \frac{1}{\alpha(\alpha-2)}, \alpha > 2$, infinite if $\alpha \leq 2$
      - $\text{cv} > 1$ for most real graphs
- Preferential attachment (the rich get richer): start with a clique of $d \geq 1$ nodes and add new nodes one at a time, each of degree d
  - $n \to \infty \Rightarrow$ power law

1.3 Small-world property

- Small-world property: any pair of nodes is connected by some short path compared to the size of the graph
  - A graph of n nodes has the small-world property if the length of the shortest path between two nodes of the graph is small compared to n. This length is typically (at most) logarithmic in n
- Erdös-Rényi graph: the degree distribution is approximately Poisson ($\frac{\lambda^k e^{-\lambda}}{k!}$, mean λ, variance λ) with $\lambda \approx np$, where p is the probability of connection between any two nodes
  - Distribution is independent of n, so path length is logarithmic in n
- Power law distribution, for $\alpha < 3$, the average path length remains finite ($O(1)$) when n growths to infinity
- Erdös number
- Bacon number: Erdös number concept to the Hollywood movie industry
  - Kevin Bacon has a Bacon number of 0
- Planar graphs: shortest paths of order $O(\sqrt{n})$

1.4 Clustering property

- Clustering coefficient: fraction of closed triangles $C = \frac{3 \, \#\text{triangles}}{\sum_i \binom{d_i}{2}}$
- Local clustering coefficient: fraction of closed triangles containing i: $C_i = \frac{\#\text{triangles containing } i}{\binom{d_i}{2}}$
  - $C = \frac{\sum_i \binom{d_i}{2} C_i}{\sum_i \binom{d_i}{2}}$
- Clustering does not emerges from randomness, unlike the small-world property
  - Random graph: expected clustering coefficient is p, which is typically very low

2 Page Rank

- Identify most important node s in a graph

2.1 Random walk

- Consider a directed graph $G = (V, E)$ of n nodes m edges, with adjacency matrix A without sink
  - Vector of out-degrees: $d^+ = A\mathbf{1}$
  - Vector of in-degrees: $d^1 = A^T\mathbf{1}$
  - Sink node: $d_i^+ = 0$
- Random walk: select a node, then walk t steps at random
  - Markov chain
  - $\pi_t \sim X_t \in V, \pi_{t+1} = \pi_t P$
    - Transition matrix: $P = D^{-1}A$, $D = \text{diag}(d^+)$, adjacency matrix A
      - Probability of going from node i to node j: $P_{ij} = \frac{A_{ij}}{d_i^+}$
  - For a strongly connected graph: $\lim_{t \to +\infty} \pi_t = \pi = \pi P$

    - π is the unique solution, up to a normalization constant (unique left eigenvector of P for the eigenvalue 1, such that $\pi\mathbf{1} = 1$, indeed π is a vector of probabilities)
- Stationary distribution: yields approximate stationary distribution with complexity $O(Km)$ in time, $O(n)$ in memory
  - Do $\pi \leftarrow \frac{1}{n}(1, \dots, 1)$
  - For $t = 1, \dots, K, \pi \leftarrow \pi P$
- Undirected graphs ($d = d^+ = d^-$): $\pi \propto d^T$
- Accounting for sinks: replace rows of zeroes by rows of ones: $A + \mathbf{1}_s \mathbf{1}^T, s = \{i : d_i^+ = 0\}$

2.2 PageRank

- Accounting for traps: walk with probability α (damping factor), probability to restart with some probability $1 - \alpha$
- Irreducible (graph is fully connected) Markov chain with transition matrix: $P^{(\alpha)} = \alpha P + (1 - \alpha)\frac{\mathbf{1}\mathbf{1}^T}{n}$
- PageRank vector: unique solution to the equations $\pi^{(\alpha)} = \pi^{(\alpha)} P = \alpha \pi^{(\alpha)} P + (1 - \alpha)\frac{\mathbf{1}^T}{n} = (1 - \alpha) \sum_{t=0}^{+\infty} \alpha^t \pi_t$, with π_0 uniform
  - No restarts: $\alpha \to 1 \Rightarrow \pi^{(\alpha)} \to \pi$
  - Frequent restarts: $\alpha \to 0 \Rightarrow \pi^{(\alpha)} \to \pi_0 + \alpha(\pi_1 - \pi_0) + o(\alpha)$
    - π_1 neighbor sampling
- PageRank
  - Do $\pi \leftarrow \frac{1}{n}(1, \dots, 1)$
  - For $t = 1, \dots, K, \pi \leftarrow \alpha \pi P + (1 - \alpha)\frac{1}{n}(1, \dots, 1)$
- Path length before restart (in the absence of sinks) has a geometric distribution ($(1-p)^k p$, mean $\frac{1-p}{p}$, variance $\frac{1-p}{p^2}$) with parameter $1 - \alpha$: $L = \frac{\alpha}{1-\alpha}$
  - $\alpha = 0.95 \Rightarrow L \approx 5.7$

2.3 Personalized PageRank

- Let μ be some distribution on $S \subset V$ (e.g. unifor,)
- Forced restarts: $P_{ij} = \begin{cases} \frac{A_{ij}}{d_i^+} & \text{if } d_i^+ > 0 \\ \mu_j & \text{otherwise} \end{cases}$
- Random restarts: $P^{(\alpha)} = \alpha P + (1 - \alpha)\mathbf{1}\mu$
- PageRank
  - Do $\pi \leftarrow \mu$
  - For $t = 1, \dots, K, \pi \leftarrow \alpha \pi P + (1 - \alpha)\mu$
- In the absence of sinks $\pi^{(\alpha)} = \sum_{s \in S} \mu_s \pi_s^{(\alpha)}$, Personalized PageRank vector associated with s $\pi_s^{(\alpha)}$

2.4 Case of bipartite graphs

- $G = (V_1, V_2, E)$, $n_1 = |V_1|$, $n_2 = |V_2|$, $A = \begin{pmatrix} \mathbf{0} & B \\ B^T & \mathbf{0} \end{pmatrix}$, $B$is the adjacency matrix of dimension $n_1 \times n_2$, $d = A\mathbf{1} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}$, $D = \text{diag}(d) = \begin{pmatrix} D_1 & \mathbf{0} \\ \mathbf{0} & D_2 \end{pmatrix}$, $P = D^{-1}A = \begin{pmatrix} \mathbf{0} & P_1 \\ P_2 & \mathbf{0} \end{pmatrix}$
- $\pi_1 = \pi_1 P_1 P_2$
- $\pi_2 = \pi_2 P_2 P_1$
- Connected graph: $\pi_1 \propto d_1$, $\pi_2 \propto d_2$
- Only restart from V_1
  - $P^{(\alpha)} = \alpha P + (1 - \alpha)\mathbf{1}\mu_1$
  - $\pi^{(\alpha)} = \pi^{(\alpha)} P$
  - $\pi^{(\alpha)} = \begin{pmatrix} \pi_1^{(\alpha)} \\ \pi_2^{(\alpha)} \end{pmatrix}$
    - $\pi_1^{(\alpha)} = (1 - \alpha) \sum_{t \in 2\mathbb{N}} \alpha^t \pi_1(t)$

        - $\mathbf{1}^T \pi_1^{(\alpha)} = \frac{1}{1+\alpha}$
        - $\pi_1^{(\alpha)} = \alpha \pi_2^{(\alpha)} P_2 + (1-\alpha)\mu_1$
    - $\pi_2^{(\alpha)} = (1-\alpha) \sum_{t \in 2\mathbb{N}+1} \alpha^t \pi_2(t)$
        - $\mathbf{1}^T \pi_2^{(\alpha)} = \frac{\alpha}{1+\alpha}$
        - $\pi_2^{(\alpha)} = \alpha \pi_1^{(\alpha)} P_1$
- BiPageRank
    - Do $\pi_1^{(\alpha)} \leftarrow \frac{1}{n_1}(1, \dots, 1), \pi_2^{(\alpha)} \leftarrow 0$
    - For $t = 1, \dots, K$
        - $\pi_1 \leftarrow \alpha \pi_2 P_2 + (1-\alpha)\mu_1$
        - $\pi_2 \leftarrow \alpha \pi_1 P_1$
- $\alpha \to 1$
    - $\pi_1^{(\alpha)} \to \pi_1 \propto d_1^T$
    - $\pi_2^{(\alpha)} \to \pi_2 \propto d_2^T$
- $\alpha \to 0$
    - $\pi_1^{(\alpha)} \to \pi_1(0)(1-\alpha) + \alpha^2 \pi_1(2) + o(\alpha^2)$
    - $\pi_2^{(\alpha)} \to \pi_2(1)\alpha + o(\alpha)$
- Co-neighbor graph: $G = (V_1, E_1)$
    - PageRank vector $\pi_1^{(\alpha)}$
    - $A_1 = B D_2^{-1} B^T$
- PageRank of nodes in the bipartite graph with damping factor α = PageRank in the co-neighbor graph with damping factor α^2
- Directed graphs as bipartite graphs: walk alternatingly in forward and backward directions

2.5 Applications

- Recommendation
- Classification
- Clustering

3 Graph Clustering

- Identify relevant groups of nodes in a graph
- Also known as community detection
- Unsupervised learning
    - Information retrieval
    - Content recommendation
    - Link prediction
    - Anomaly detection
- Clustering of a graph $G = (V, E)$ is any function $C : V \to \{1, \dots, K\}$

3.1 Modularity

- Volume of the graph $v = 2m$ (in the absence of self-loops)
- Kronecker symbol δ
- $Q(C) = \frac{1}{v} \sum_{i,j \in V} \left(A_{ij} - \frac{d_i d_j}{v} \right) \delta_{C(i),C(j)} \in [-1, 1]$
    - Reference graph: $\hat{A} = \frac{dd^T}{v}$
        - $\hat{A}\mathbf{1} = d$
        - $\mathbf{1}^T \hat{A} \mathbf{1} = v$
- $Q(C) = \sum_{i,j \in V} (p(i,j) - p(i)p(j)) \delta_{C(i),C(j)}$
    - Edge sampling induces a probability distribution on node pairs: $p(i,j) = \frac{A_{ij}}{v}$
    - Marginal distribution: $p(i) = \frac{d_i}{v}$

- $Q(C) = \sum_k (\frac{m_k}{m} - (\frac{v_k}{v})^2)$
  - Edge sampling induces a probability distribution on clusters: $p_C(k,l) = \sum_{i,j:C(i)=k,C(j)=l} p(i,j)$
    - $p_C(k,k) = \frac{m_k}{m}$, number of edges in cluster k m_k
  - Marginal distribution: $p_C(k) = \sum_{i:C(i)=k} p(i) = \frac{v_k}{v}$, volume of cluster k $v_k = \sum_{i:C(i)=k} d_i$
  - Simpson index: $S = \sum_k (\frac{v_k}{v})^2$
    - $S \to \frac{1}{K} \Leftrightarrow$ most diverse
    - $S \to 1 \Leftrightarrow$ least diverse
    - Standard measure of diversity/concentration in biology
    - Maximum modularity with K clusters: $1 - \frac{1}{K}$
- Difference between a fit metric and a diversity metric
- Aggregate graph
  - $A_C = MAM^T$
    - Membership matrix M with $M_{ik} = 1$ if node i belongs to cluster k
  - Weighted edges and self-loops according to the edges between and inside clusters
  - The weight of the edge between nodes k and l is the total weight of edges beween clusters k and l in the original graph
  - Has self-loops, with a weight of the self-loop of node k equal to the total weight of self-loops in cluster k in the original graph, plus twice the total weight of regular edges within cluster k in the original graph
  - Modularity is preserved

3.2 The Louvain algorithm

- $\max_C Q(C)$
  - Combinatorial problem
  - NP-hard
- Louvain algorithm: greedy algorithm
  - Initialization: $C \leftarrow$ identity
  - Maximization: while modularity $Q(C)$ increases, update C by changing the cluster of each node
  - Aggregation: merge all nodes belonging to the same cluster into a single node, update the weights and apply the previous step to the aggregate graph

3.3 Cluster strengths

- Cluster strength: probability to stay cluster k after one move
  - $\rho_k = \frac{\text{total internal degree}}{\text{total degree}} = \frac{2m_k}{v_k}$
- π_k probability to be in cluster k
  - $Q(C) = \sum_k \pi_k(\rho_k - \pi_k)$

3.4 Resolution

- For a large number of clusters of (approximately) equal weights: $\sum_k (\frac{v_k}{v}) \approx \frac{1}{K} \approx 0$
  - Modularity is not able to detect high-resolution clusterings
- Parameter $\gamma > 0$ that controls the fit-diversity trade-off: $Q_\gamma(C) = \frac{1}{v} \sum_{i,j \in V} \left(A_{ij} - \gamma \frac{d_i d_j}{v} \right) \delta_{C(i),C(j)} \in [-1, 1]$
  - The resolution limit, beyond which all clusters have size 1, is the maximum link strength: $\gamma = \max_{i \neq j} \sigma(i,j)$
  - The first node pair i, j merged by Paris is that merged by Louvain at the resolution limit

3.5 Extensions

- Directed graphs:
  - $Q(C) = \frac{1}{v} \sum_{i,j \in V} \left(A_{ij} - \frac{d_i^+ d_j^-}{v} \right) \delta_{C(i),C(j)} \in [-1, 1]$
  - $Q(C) = \sum_k (\frac{m_k}{m} - \frac{v_k^+ v_k^-}{v^2})$
- Bipartite graphs
  - Undirected: $A = \begin{pmatrix} \mathbf{0} & B \\ B^T & \mathbf{0} \end{pmatrix}$
  - Directed: $A = \begin{pmatrix} \mathbf{0} & B \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$

4 Hierarchical Clustering

- Divisive algorithms: e.g. through successive k-means
- Agglomerative algorithms: successive merges of the closest clusters $a, b \subset 1, \dots, n$

Linkage	$d(a, b)$
Single	$\min_{i \in a, j \in b} \|x_i - x_j\|$
Complete	$\max_{i \in a, j \in b} \|x_i - x_j\|$
Average	$\frac{1}{\|a\|\|b\|} \sum_{i \in a, j \in b} \|x_i - x_j\|$
Ward	$\frac{\|a\|\|b\|}{\|a\|+\|b\|} \|g_a - g_b\|^2$

- Local search by the nearest-neighbor chain

4.1 Notion of dendrogram

- Typically a binary tree, with nodes as leaves
- Lower the merge, the stronger the cluster
- In general are regular, non-decreasing sequence of heights departing from the leaves

4.2 Divisive algorithm

- Hierarchical Louvain
  - HierarchicalLouvain(G):
    - clusters ← Louvain(G)
    - If |clusters| > 1
      - graphs ← GetSubgraphs(G, clusters)
      - Return [HierarchicalLouvain(S) for S in graphs]
    - Else:
      - Return [nodes(G)]

4.3 Agglomerative algorithm

- Edge sampling: $p(i, j) = \frac{A_{ij}}{v}$
- Marginal distribution: $p(i) = \frac{d_i}{v}$
- Conditional distribution: $p(i|j) = \frac{A_{ij}}{d_j}$
- Link strength: $\sigma(i, j) = \frac{p(j|i)}{p(j)} = \frac{p(i|j)}{p(i)} = \frac{p(i,j)}{p(i)p(j)} = v \frac{A_{ij}}{d_i d_j}$
- Paris (Pairwise Agglomeration Induced by Sampling) algorithm
  - For $t = 1, \dots, n-1$
    - $i, j \leftarrow \arg \max_{i,j \in V, i \neq j} \sigma(i, j)$

        * Merge i, j into node $n + t$
        * Update σ
- New (weighted) adjacency matrix:
  - $A_{i\cup j,k} \leftarrow A_{i,k} + A_{j,k}, \forall k \in V \backslash \{i, j\}$
  - $A_{i\cup j,i\cup j} \leftarrow A_{i,i} + A_{j,j} + 2A_{i,j}$
- New sampling distribution:
  - $p(i \cup j, k) = p(i, k) + p(j, k), \forall k \in V \backslash \{i, j\}$
  - $p(i \cup j, i \cup j) = p(i, i) + p(j, j) + 2p(i, j)$
- New link strengths:
  - $p(i \cup j) = p(i) + p(j)$
  - $\sigma(i \cup j, k) = \frac{p(i)}{p(i)+p(j)}\sigma(i, k) + \frac{p(j)}{p(i)+p(j)}\sigma(j, k), \forall k \neq i, j$
- Distance: $d(i, j) = \frac{1}{\sigma(i,j)} = \frac{d_i d_j}{vA_{ij}}$
  - $d(i \cup j, k) \geq \min(d(i, k), d(j, k))$
  - Consequence: the distances of successive merges is non-decreassing so that the dendogram is regular (no inversion)
- Paris with the NN chain
  - While $|V| > 1$:
    * Take a node at random
    * Build the chain of nearest-neighbors
    * Merge the two last nodes of this chain
    * Update σ
    * Restart the chain

4.4 Extensions

- Case of weighted graphs
  - $\sigma(i, j) = v\frac{A_{ij}}{w_i w_j} = v\frac{A_{ij}+A_{ji}}{d_i^+ d_j^- + d_j^+ d_i^-}$
- Bipartite graphs (undirected)

5 Heat Diffusion

- Ranking both hot and cold sources
- Semi-supervised classification

5.1 Heat diffusion

- T_i temperature of node i
- By heat exchanges along the edges $\frac{dT}{dt} = \sum_j A_{ij}(T_j - T_i) = AT - DT = -LT$
- Laplacian matrix: $L = D - A$
  - Symmetric
  - Positive semi-definite
  - Discrete differential operator $L = \nabla\nabla^T$, the $n \times m$ (directed) incidence matrix of the graph ∇
  - Given some arbitrary direction of the edges, the incidence matrix applied to the vector T gives the temperature difference over the edges: $\nabla T = [T_j - T_i]_{i\to j}$
- Continuous time: $T(t) = e^{-LT}T(0)$
  - Conservation: $\frac{dT}{dt} = 0$
  - Equilibrium: $LT = 0$
    * Vector T is said to be *harmonic*
    * $AT = DT \Rightarrow T_i = \frac{1}{d_i}\sum_{j\in V} A_{ij}T_j, \forall i \in V$
    * Graph is connected: solution is a constant vector; all nodes have the same temperature at equilibrium
      · Each node at equilibrium is the average temperature in the initial state

        * Graph is not connected: the solution is a constant vector per connected component, with temperature in each connected component equal to the average temperature in this connected component in the initial state
- Spectral analysis: $L = U\Lambda U^T$, if the graph is connected, then $0 = \lambda_1 < \lambda_2 \leq ...$ and the convergence is exponential at rate λ_2: $e^{-Lt} = Ue^{-Lt}U^T \to \frac{\mathbf{1}\mathbf{1}^T}{n}$
- Diffusion in discrete time: $T(t+1) = ((1-\alpha)I + \alpha P)T(t), \forall t = 1, 2, ...$, damping factor α
  - $T(t) = ((1-\alpha)I + \alpha P)^t T(0)$
  - Equilibrium:
    * $\Delta T = 0$
    * $\forall \geq 0, \tilde{T}(t) = \tilde{T}(0) = \frac{\sum_i d_i T_i(t)}{\sum_i d_i}$
- Spectral analysis: $P = V\Gamma V^T D$, if the graph is connected, then $\gamma_1 = 1 > \gamma_2 \geq ...$ and the convergence is exponential at rate $\max_{k\geq 2} |1 - \alpha + \alpha\gamma_k|$: $((1-\alpha)I + \alpha P)^t = V((1-\alpha)I + \alpha\Gamma)^t V^T D \to \mathbb{1}\mathbb{1}^T D$

5.2 Dirichlet problem

- Dirichlet problem: $\forall i \notin S, (LT)_i = 0 \Leftrightarrow T_i = (PT)_i$
  - $T_i = \sum_j P_{i\to j} T_j$
  - Dirichlet energy: $E = \frac{1}{2} T^T L T = \frac{1}{2} \|\nabla T\|^2$
  - There is at most one solution
  - The solution is the weighted average of temperatures of nodes in S with weights given by the hitting probabilities of each node in S
- Free diffusion: $t < +\infty$, average temperature is preserved
- Dirichlet: $t \to +\infty$, temperatures between minimum and maximum

5.3 Applications

- Classification
  - Seeds with label 1: hot sources
  - Seeds with label 0: cold sources
  - Prediction by thresholding: $\bar{T} = \frac{1}{n} \sum_i T_i$
- Multi-label classification: one-against-all strategy
  - Seeds with label l: hot sources
  - Seeds with label other labels: cold sources
  - Assign the label of highest temperature

5.4 Extensions

- Bipartite graphs: heat diffusion in discrete time may require averaging two subsequent time-steps
- Directed graphs: $T(s+1) = PT(s)$

6 Spectral Embedding

- Represent node of a graph in a space

6.1 Laplacian matrix

- Laplacian matrix $L = U\Lambda U^T$
  - $U = (u_1, ..., u_n)$, with $U^T U = I$, $u_1 \propto 1$
  - $\Lambda = \text{diag}(\lambda_1, ..., \lambda_n)$, with $\lambda_1 = 0 \leq \lambda_2 \leq \cdots \leq \lambda_n$
- The multiplicity of the eigenvalue $\lambda = 0$ of the Laplacian matrix L is equal to the number of connected components of the graph
- Dynamics: $T(t) = e^{-Lt}T(0), \forall t \geq 0$, with $e^{-Lt} = Ue^{-Lt}U^T$
- When the graph is disconnected, use $A' = A + \frac{\mathbf{1}\mathbf{1}^T}{n}$, $D' = D + I$

6.2 Transition matrix

- Transition matrix $P = D^{-1}A$
  - Stochastic matrix: $P \geq 0$ and $P\mathbf{1} = \mathbf{1}$
- $P = V\Gamma V^T D$
  - $U = (u_1, \dots, u_n)$, with $V^T DV = I$, $v_1 \propto 1$
  - $\Gamma = \text{diag}(\gamma_1, \dots, \gamma_n)$, with $\gamma_1 = 1 \geq \gamma_2 \geq \dots \geq \gamma_n \geq -1$
- The multiplicity of the eigenvalue $\gamma = 1$ of the transition matrix L is equal to the number of connected components of the graph
- Dynamics: $T(t) = P^t T(0), \forall t \geq 0$, with $P^t = V\Gamma^t V^T D$
- The transition matrix of a bipartite graph has a symmetric spectrum: γ eigenvalue $\Leftrightarrow$ $-\gamma$ eigenvalue

6.3 Spectral embedding

- $\min_X \sum_{i,j \in V} A_{ij} \|X_i - X_j\|^2$
- $\text{tr}(X^T LX) = \frac{1}{2} \sum_{i<j} A_{ij} \|X_i - X_j\|^2$
- Embedding $X = (u_2, \dots, u_{K+1})$ given by the first K eigenvectors (except the first) of the Laplacian matrix L
  - The spectral embedding is optimal: $X = \arg \min_{X: X^T \mathbf{1} = 0, X^T X = I_K} \text{tr}(X^T LX)$
  - The embedding is centered: $\sum_{i=1}^{n} X_i = 0$
  - Mechanical system: put nodes on a line at positions $x_1, \dots, x_n \in \mathbb{R}$
    - Nodes: particles
    - Edges: (atractive) springs
    - Potential energy: $E = \frac{1}{2} \sum_{1<j} A_{ij}(x_i - x_j)^2 = \frac{1}{2} x^T Lx$
  - Harmonic oscillator
    - Let the system evolve, assuming unit masses, starting from positions $x_1, \dots, x_n \in \mathbb{R}$
    - $\ddot{x}_i = \sum_j A_{ij}(x_j - x_i), \forall i \Leftrightarrow \ddot{x} = -Lx$
    - Eigenvectors of $L \rightarrow$ eigenmodes
    - Eigenvalues of $L \rightarrow$ levels of energy
    - The most interesting eigenmodes are those of lowest energy (equivalently, of lowest eigen frequency)
- Embedding $X = (v_2, \dots, v_{K+1})$ given by the first K eigenvectors (except the first) of the transition matrix P
  - The spectral embedding is optimal $X = \arg \min_{X: X^T d = 0, X^T DX = I_K} \text{tr}(X^T LX)$
  - The weighted embedding is centered: $\sum_{i=1}^{n} d_i X_i = 0$
  - Harmonic oscillator $D\ddot{x}_i \sum_j A_{ij}(x_j - x_i), \forall i \Leftrightarrow \ddot{x} = -(I - P)x$
    - Eigenvectors of $P \rightarrow$ eigenmodes
    - $1-$ eigenvalues of $P \rightarrow$ levels of energy

6.4 Algorithms

- Need to compute the first eigenvectors of some matrix M (either the Laplacian L or the normalized Laplacian $D^{-\frac{1}{2}} LD^{-\frac{1}{2}}$)
  - $x \leftarrow \frac{Mx}{\|Mx\|}$
- Lanczos' algorithm
  - Power iteration
- Halko's algorithm
  - Random projection
  - Power iteration
  - QR decomposition

- The adjacency matrix becomes dense but with a nice sparse + low rank structure

6.5 Extensions

- Weighted graphs
- Bipartite graphs
- Directed graphs
  - See directed graph as a bipartite graph

7 Graph Neural Networks

7.1 Background on neural networks

- Supervised learning: predict the label (classification) or the value (regression) by training
  - Formally: learn some mapping $f : x \mapsto y$ minimizing: $\frac{1}{n}\sum_{i=1}^{n} l(y_i, f(x_i))$
    - $x \in \mathbb{R}^d$
    - $y \in \{0,1\}, \{1,\dots,K\}\text{or}\mathbb{R}$
    - $(x_1, y_1), \dots, (x_n, y_n)$ are the training examples
    - l is the loss function
- Binary classification: $x \in \mathbb{R}^d, y \in \{0,1\}$
  - Logistic regression: probability tht $y = 1$ for sample x: $p = \sigma(w^T x) \in [0,1]$, weight vector $w \in \mathbb{R}^d$ (to be learned)
    - Logistic function $\sigma(u) = \frac{1}{1+e^{-u}}$
  - Bias term: Logistic regression: probability tht $y = 1$ for sample x: $p = \sigma(w^T x + b) \in [0,1]$, weight vector $w \in \mathbb{R}^d$ and bias term $b \in \mathbb{R}$ (to be learned)
  - Loss function: binary cross-entropy
    - For one sample: $-y\log(p) - (1-y)\log(1-p)$
    - For n samples: $-\sum_{i=1}^{n}(y_i \log(p_i) + (1-y_i)\log(1-p_i))$
  - Objective: find w and b minimizing: $L = -\sum_{i=1}^{n}(y_i \log(p_i) + (1-y_i)\log(1-p_i))$, with $p_1 = \sigma(w^T x_1 + b), \dots, p_n = \sigma(w^T x_n + b)$
  - Regularization: find w and b minimizing: $L = -\sum_{i=1}^{n}(y_i \log(p_i) + (1-y_i)\log(1-p_i)) + \frac{\lambda}{2}(\|w\|^2 + b^2)$, with $p_1 = \sigma(w^T x_1 + b), \dots, p_n = \sigma(w^T x_n + b)$, hyperparameter λ
  - Gradient descent: optimization problem $\arg\min_{w,b} L$
    - Algorithm: iterate over, learning rate α:
      - $w \leftarrow w - \alpha\frac{\partial L}{\partial w}$
      - $b \leftarrow b - \alpha\frac{\partial L}{\partial b}$
    - For one sample:
      - $\frac{\partial L}{\partial w} = (p-y)x$
      - $\frac{\partial L}{\partial b} = p - y$
    - For multiple samples:
      - $\frac{\partial L}{\partial w} = \lambda w + \sum_{i=1}^{n}(p_i - y_i)x_i$
      - $\frac{\partial L}{\partial b} = \lambda b + \sum_{i=1}^{n} p_i - y_i$
- Multi-class: $x \in \mathbb{R}^d, y \in \{1,\dots,K\}$
  - Softmax regression: for $k = 1,\dots,K$, probability that $y = k$ for sample x: $p(k) = \frac{e^{w_k^T x + b_k}}{e^{w_1^T x + b_1} + \dots + e^{w_K^T x + b_K}}$, weight vectors $w_1, \dots, w_K$, bias terms $b_1, \dots, b_K$ (to be learned)
  - Loss function: cross-entropy
    - For one sample: $-\sum_{k=1}^{K} 1_{\{y=k\}} \log(p(k))$, $p(k) \propto e^{w_k^T x + b_k}$

- Objective: find $w_1, \dots, w_K$ and $b_1, \dots, b_K$ minimizing: $L = -\sum_{i=1}^{n}\sum_{k=1}^{K} 1_{\{y=k\}} \log(p_i(k)) + \frac{\lambda}{2}\sum_{k=1}^{K}(\|w_k\|^2 + b_k^2)$, with $p(k) \propto e^{w_k^T x + b_k}$, hyperparameter λ
- Gradient expression
  - For one sample:
    - $\frac{\partial L}{\partial w_k} = (p(k) - 1_{\{y=k\}})x$
    - $\frac{\partial L}{\partial b_k} = p(k) - 1_{\{y=k\}}$
  - For n samples with regularization:
    - $\frac{\partial L}{\partial w_k} = \lambda w_k + \sum_{i=1}^{n}(p_i(k) - 1_{\{y_i=k\}})x_i$
    - $\frac{\partial L}{\partial b_k} = \lambda b_k + \sum_{i=1}^{n}(p_i(k) - 1_{\{y_i=k\}})$

- Neural network: a composition of functions of the form $x \mapsto \sigma(Wx + b)$
  - Each such function is a layer of the network
  - The output of the neural network is a probability distribution (for classification) or value (for regression)
  - Activation functions
    - Logistic function: $u \mapsto \frac{1}{1+e^{-u}}$
    - ReLU function: $u \mapsto \max(u, 0)$
  - Objective: find weights matrices $W_1, \dots, W_K$ and bias vectors $b_1, \dots, b_K$ minimizing: $L = -\sum_{i=1}^{n}\sum_{k=1}^{K} 1_{\{y=k\}} \log(p_i(k)) + \frac{\lambda}{2}\sum_{l=1}^{N}(\|W_l\|^2 + b_l^2)$, with $p_i = f_N \circ \dots \circ f_1(x_i)$, $f_l(x) = \sigma_l(W_l x + b_l)$, hyperparameter λ
  - Parameters to learn

Layer	Weights W	Biases b
1	$d_1 \times d$	d_1
2	$d_2 \times d_1$	d_2
$\vdots$	$\vdots$	$\vdots$
N	$K \times d_{n-1}$	K

  - Backpropagation:
    - Single layer graph neural network
      - $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial U}\frac{\partial U}{\partial W}$
      - $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial U}\frac{\partial U}{\partial b}$
    - 2-layer graph neural network
      - Layer 2: $\frac{\partial L}{\partial W_2}, \frac{\partial L}{\partial b_2}$
      - Layer 1: $\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial b_1}$

7.2 Graph neural networks

- Learn node embeddings, using:
  - Node features → neural net
  - Graph → message passing (cf. diffusion)
- A graph neural network is:
  - A composition of a diffusion step: $X \mapsto U = PX$, transition matrix $P = D^{-1}A$ and matrix of features X (dimension $n \times d$)
  - Followed by a linear transformation: $U \mapsto U' = UW^T + \mathbf{1}b^T$
  - Followed by an activation function: $U' \mapsto V = \sigma(U')$
    - Each such function is a layer of the network
  - The output of the graph neural network is a probability distribution (for classification) or a value (regression)

- Objective: find weights matrices $W_1, \dots, W_K$ and bias vectors $b_1, \dots, b_K$ minimizing: $L = -\sum_{i \in S} \sum_{k=1}^{K} 1_{\{y=k\}} \log(p_i(k)) + \frac{\lambda}{2} \sum_{l=1}^{N} (\|W_l\|^2 + b_l^2)$, with $p_i = f_N \circ \dots \circ f_1(x_i)$, $f_l(x) = \sigma_l(PXW_l^T + \mathbf{1}b_l^T)$, hyperparameter λ
- Parameters to learn

Layer	Weights W	Biases b
1	$d_1 \times d$	d_1
2	$d_2 \times d_1$	d_2
$\vdots$	$\vdots$	$\vdots$
N	$K \times d_{n-1}$	K

- Backpropagation:
  - Single layer graph neural network
    - $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial U} \frac{\partial U}{\partial W}$
    - $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial U} \frac{\partial U}{\partial b}$
  - 2-layer graph neural network
    - Layer 2: $\frac{\partial L}{\partial W_2}, \frac{\partial L}{\partial b_2}$
    - Layer 1: $\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial b_1}$
- GNN as
  - a neural network → use an empty graph
  - an embedding technique → the last (hidden) layer
  - a diffusion process → use on-hot encoding of labels + identity mapping (no training, $W = I, b = 0$)

7.3 Variants

- $X \mapsto U = PX \mapsto U' = UW^T + \mathbf{1}b^T \mapsto V = \sigma(U')$
- Message passing:
  - Replace tha transition matrix by
    - $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ (symmetric normalization)
    - $I + P$ (add self-embedding)
    - (I, P) (concatenate self-embedding) → GraphSAGE
- Sampling
  - Replace the transition matrix P by $P = \tilde{D}^{-1}\tilde{A}$ where:
    - $\tilde{A}$ is the adjacency matrix of a sampled graph (e.g. at most k neighbors per node)
    - The sampling can depend on the layer → GraphSAGE
- Normalization
  - Normalize V so that each embedding lies on the unit sphere: $V \mapsto V' = \frac{V}{\|V\|}$ → GraphSAGE
- Pooling
  - From node embedding to graph embedding: $X \mapsto U = PX \mapsto U' = UW^T + \mathbf{1}b^T \mapsto V = \sigma(U') \mapsto \frac{\mathbf{1}^T V}{n}$
  - Each sample = one graph
- Link prediction
  - From node embedding to link prediction: $X \mapsto U = PX \mapsto U' = UW^T + \mathbf{1}b^T \mapsto V = \sigma(U') \mapsto S = \sigma(VV^T)$
  - S is a similarity matrix of size $(n \times n)$
  - $S_{i,j}$ is the probability that a link exists between nodes i and j