Björn Bebensee (2019-21343)

Graphs

Densification power law

Number of edges $E(t)$ and the number of nodes $N(t)$: $E(t) \propto N(t)^a$ with $a \in [1,2]$. $a = 1 \Rightarrow$ constant out-degree (linear growth), $a = 2 \Rightarrow$ quadratic growth. Typically $a > 1$, densification, shrinking diameter.

Erdös-Rényi random graph model

Node number $n \in \mathbb{N}$, $p \in [0,1]$. Create n nodes, include edge between any two nodes w/ prob. p. Degree dist. = Poisson, can be described by $P(k) \sim \frac{e^{-\lambda}\lambda^k}{k!}$ where k is the degree. However: real graph's degree dist. follows power law: $P(k) \sim k^{-r}$.

Preferential attachment

New nodes attach preferentially to well-connected nodes. Link to existing node i by $P_i = \frac{d(i)}{\sum_j d(j)}$. Robust against random failure, vulnerable to attacks.

Community connection model

Generates many CCs. Joining node connects to any host with p_{host}, begins random walk with p_{step} until there are no more steps. Repeat for all hosts. Describes rebel probability well. $p_{Rebel} \propto s^{ad}$ with d deg newcomer, s rel. size of DC

Power laws in the internet

PL1 (**rank exponent**): The out-degree d_v of a node v is proportional to the rank of a node r_v to the power of a constant $\mathcal{R}$: $d_v \propto r_v^{\mathcal{R}}$

PL2 (**out-degree exponent**): The frequency f_d of an out-degree d is proportional to the out-degree to the power of a constant $\mathcal{O}$: $f_d \propto d^{\mathcal{O}}$

PL3 (**eigen exponent**): The eigenvalues λ_i of a graph are proportional to the order i to the power of a constant $\mathcal{E}$: $\lambda_i \propto i^{\mathcal{E}}$

Generation of power law distributions

PL distribution: $p(x) = Cx^{-a}\ \forall\, x \geq x_{\min}$ for constant C. Estimate exponent a:

$$a = 1 + n\left(\sum_{i=1}^{n} ln\left(\frac{x_i}{x_{\min}}\right)\right)^{-1}$$

Chinese restaurant process (Yule distribution): Newcomer sits down at existing table, prefers large groups. New table with probability $\frac{1}{m}$. This is also referred to as a *Yule process*.

Combination of exponentials: Radioactive decay: half-life $-a$ and $p(y) = e^{ay}$. Russian roulette: capital x increases every time they survive with $x \sim e^{by}$. Final cap. PL dist. Monkey typing on a typewriter: frequency of the x-th most frequent word is x^{-a}, i.e. power law.

Random walks: steps required to arrive at the same position (*inter-arrival time*) follows power law.

Random multiplication: Starting C dollars, interest rate $s(t)$ for each year t, we get $C(t) = C(t-1)(1+s(t))$. We have $\log C(t) = \log C + \log.. + \log..$ which is a Gaussian distribution and thus $C(t) = \exp(\text{Gaussian})$ which is a lognormal distribution. The lognormal distribution looks like a power law in its tail distribution (but strictly is not a power law).

Fragmentation: Stick of length 1, recursively break at random point $0 \leq x \leq 1$. Resulting length distribution is lognormal (analogous to random multiplication).

Spectral analysis

$I = CV = \frac{V}{R} \Leftrightarrow V = RI, C = \frac{1}{R}$

In Series: $R = R_1 + \ldots + R_m$

In parallel: $R = \frac{1}{\frac{1}{R_1} + \ldots + \frac{1}{R_m}}$

Effective resistance: $R_{ab} = \frac{V_{ab}}{I_{ab}}$

Kirchhoff's law

flow in = flow out. Given $I_1 = C_1(V_1 - V)$, $I_2 = C_2(V - V_2)$, then $I_1 = I_2$ and $V = \frac{C_1}{C} V_1 + \frac{C_2}{C} V_2$

Weights C for conductance C_{ij} and $C_i = \sum_{(i,j)\in E} C_{ij}$. The adjacency matrix A is $A_{ij} = C_{ij}$ if $(i,j) \in E$ and 0 otherwise. Laplacian matrix L is $L = D - A$ where D is the diagonal matrix with entries $D_{ii} = C_i$. Then,

$$L_{ij} = \begin{cases} C_i & \text{if } i = j \\ -C_{ij} & \text{if}(i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Let V be a vector containing the voltages for all nodes. Then $(LV)_i$ is the residual current at node i. Assuming $I = 1$ effective resistance is solution of

$$LV = \begin{bmatrix} 1 \\ 0 \\ \cdots \\ 0 \\ -1 \end{bmatrix}$$

as $R_{1n} = \frac{V}{I} = V = V_1 - V_n$.

Random walks and electric networks

Interpretation of voltage: Random walk from x to b, h_x probability of visiting a before b. Then

$h_a = 1, h_b = 0$

$h_x = \sum_y h_y P_{xy}$

where probability P_{xy} of choosing to go along edge (x,y) is the weight of the edge as a fraction over the sum of the weights of all of x's edges.

Assume $V_a = 1$ and $V_b = 0$, then

$V_x = \sum_y V_y \frac{C_{xy}}{C_x} = \sum_y V_y P_{xy}$

and thus $h = V$ as h and V are harmonic with the same boundary values. Then $V_x = h_x$ and can be measured.

Interpretation of current: Random walk from a to b. Let u_x expected number of visits to a point x before reaching b. Then $u_b = 0$ and

$u_x = \sum_y u_y P_{yx} = \sum_y u_y \frac{C_y}{C_x} P_{yx} = \sum_y u_y \frac{C_x}{C_y} P_{xy}$

and it follows that $\frac{u_x}{C_x} = \sum_y \frac{u_y}{C_y} P_{xy}$

Now let $V_a = \frac{u_a}{C_a}$ and $V_b = 0$, then $V_x = \sum_y V_y P_{xy}$ and $V_x = \frac{u_x}{C_x}$. The current i_{xy} is given by

$$i_{xy} = (V_x - V_y)C_{xy} = u_x P_{xy} - u_y P_{yx}$$

expected # crossings:

(x $\to$ y) $u_x P_{xy}$

(y $\to$ x): $u_y P_{yx}$

expected net crossings: i_{xy}

$\Rightarrow$ current from x to y (set $V_a = \frac{u_a}{C_a}$ at a and $V_b = 0$ at b)

Random walks on graphs

W_{ij} the weight of an edge (i,j), W_i sum of to i incident edge weights

Random walk probabilities: $P_{ij} = \frac{W_{ij}}{W_i}$

Hitting time: $H(i,j)$ expected number of steps before node j is visited starting from i. $H(i,j)$ is not equal $H(j,i)$.

Commute time: $k(i,j) = H(i,j) + H(j,i)$. Symmetric.

Compute hitting time: $H(x) = H(x,b)$ for a fixed b expected number of steps to reach b from x. $H(b) = 0$. Then

$H(x) = 1 + \sum_y H_y P_{xy} = 1 + \sum_y H(y) \frac{W_{xy}}{W_x}$

Compute commute time: Let$C = \sum_i C_i$, then

$k(i,j) = C \times (\text{Effective resistance})_{ij}$

HITS algorithm

Given a root set, expand to obtain base set by one move forward and backward. Let h, a be vectors with the hub and authority scores of all nodes.

Authority scores:

$$a_i = \sum_{j:(j,i)\in E} h_j \qquad a = A^T h$$

Hub scores:

$$h_i = \sum_{j:(i,j)\in E} a_j \qquad h = Aa$$

Starting from random a', h' iterate until we converge. Solutions are the left- and right-singular vectors of the adjacency matrix A with the strongest singular values.

PageRank

Random walk, popular = high steady state probability (SSP). High SSP if connected with many high SSP nodes. A adjacency matrix, B the column-normalized transition matrix (stochastic). Note: B is transposed so that the columns represent "from" and the rows represent "to".

Perron-Frobenius Theorem: $\exists p : Bp = \lambda p$ where λ is the highest eigenvalue and $\lambda = 1$ (column-normalized).

Power iteration: Start p_t, get $p_{t+1} = Bp_t$. Converges to eigenvector with largest eigenvalue which is exactly p.

B not irreducible (not all nodes reachable): add edges to all other nodes with transition probability $1-c$. Thus

$$p = cBp + \frac{(1-c)}{n} 1 = \frac{(1-c)}{n}(I - cB)^{-1} 1$$

Alternatively we can write the modified transition matrix as

$$M = cB + \frac{1-c}{n} 1\, 1^T$$

and compute p through power iteration:

$$p = Mp$$

where p denote the SSP and PageRank scores of M.

Random walk with restart

Compute proximities of other nodes given query node. Application: find caption for given image. Like PageRank except jump back to the starting node with probability $1-c$. Thus, we have

$$p_k = cBp_k + (1-c)e_k.$$

Link prediction

Graph distance: negated shorted path length

Common neighbors: $|\Gamma(x) \cap \Gamma(y)|$, but high degree $\to$ high score

Jaccard's coefficient: $\frac{|\Gamma(x)\cap\Gamma(y)|}{|\Gamma(x)\cup\Gamma(y)|}$

Adamic/Adar: $\sum_{z\in\Gamma(x)\cap\Gamma(y)} \frac{1}{\log|\Gamma(z)|}$

small x: $\frac{1}{\log x} \gg \frac{1}{x}$, large x: $\frac{1}{\log x} \approx \frac{1}{x}$

Preferential attachment: $|\Gamma(x)| \cdot |\Gamma(y)|$.

Katz proximity: $\sum_{l=1}^{\infty} \beta^l \cdot |\text{paths}_{x,y}^{\langle l\rangle}|$

where $\text{paths}_{x,y}^{\langle l\rangle}$ paths from x, y of length l.

$(M^k)_{ij}$ gives number of paths of length k.

$$I + [\beta M + \beta M^2 + \ldots + \beta M^{\infty}] = (I - \beta M)^{-1} - I$$

Hitting time/commute time: hitting time: $-H_{x,y}$

HT stationary-normed: $-H_{x,y} \cdot \pi_y$

commute time: $-(H_{x,y} + H_{y,x})$

CT stationary-normed: $-(H_{x,y} + H_{y,x}) \cdot \pi_y$

where π_y proportion of the time at node y.

Rooted PageRank: Define score to be stationary probability of y in a random walk that returns to x with probability α at each step, moving to a random neighbor with probability $1-\alpha$.

SimRank: Fixed point of: nodes similar $\Leftrightarrow$ similar neighbors.

$$\begin{cases} 1 & \text{if } x = y \\ \gamma \cdot \frac{\sum_{a\in\Gamma(x)}\sum_{b\in\Gamma(y)} sim(a,b)}{|\Gamma(x)|\cdot|\Gamma(y)|} & \text{otherwise} \end{cases}$$

For random walk SimRank is the expected value of γ^l where l is a random variable giving the time at which random walks starting from x and y first meet.

Low rank approximation: Low-rank approximation M_k of the adjacency matrix M (noise reduction technique). Then use Katz measure etc. on M_k.

Unseen bigram: Augment score with $\text{score}(z,y)$ for similar nodes $z \in S_x^{\langle\delta\rangle}$:

weighted: $|\{z : z \in \Gamma(y) \cap S_x^{\langle\delta\rangle}\}|$

unweighted: $\sum_{z\in\Gamma(y)\cap S_x^{\langle\delta\rangle}} \text{score}(x,z)$

Clustering: Cluster the graph and delete weak edges, then recompute the similarity score $sim(x,y)$ and only new links in the same cluster will be predicted.

Adamic/Adar $\approx$ Katz $\approx$ low rank inner product

Jaccard $\approx$ rooted PageRank $\approx$ SimRank

Small world: graph distance does not work.

Breadth: random predictor worsens.

Triangle Counting

Matrix multiplication: $O(n^3)$

Fast matrix multiplication: $O(n^{2.376})$

Node iterator: $\sum_{v\in V} \binom{d(v)}{2} = O(nd_{\max}^2)$

Edge iterator: $\sum_{(u,w)\in E} d(u) + d(w) = \sum_{v\in V} d(v)^2$

Forward algorithm: $\theta(n^{1.5})$ (optimal), space in $\theta(n)$.

Number of triangles: $\Delta(G) = \frac{1}{6}\sum_i \lambda_i^3$

Proof: α_{ii} of A^3 triangles of i. $tr(A^3)$ 3× # triangles (3 participating nodes). Undirected: double counting. Thus: $\Delta(G) = \frac{1}{6}\text{trace}(A^3)$. λ EV of A then λ^k EV of A^k $(k \geq 1)$. With $\sum_{i=1}^n \lambda_i = \text{trace}(A)$ obtain $\Delta(G) = \frac{1}{6}\sum_{i=1}^n \lambda_i^3$.

Local triangles: $\Delta_i(G) = \frac{\sum_j \lambda_j^3 u_{i,j}^2}{2}$

where $u_{i,j}$ the i-th entry of j-th eigenvector.

Proof: A_{nxn} is symmetric so $A = U_n \Sigma U_n'$, where Σ diagonal with $\text{diag}(\Sigma) = \vec{\Lambda}_n$ (all EVs real and U_n orthogonal. Thus $A^3 = U_n \Sigma^3 U_n'$) and that each triangle is counted twice.

Lanczos method: top-k eigenvalues. Mean required required to achieve ≥95% accuracy is 6.2.

Doulion's algorithm: sampling-based, construct smaller G': keep edge with p, discard $1-p$. Count the triangles in G' and multiply count by $\frac{1}{p^3}$ (probability of a triangle remaining in G' is p^3 as all three edges have to survive).

MapReduce

kNN: Map: k-nearest neighbors to p in input, Reduce: k-nearest neighbors global

kMeans: Map: assign closest centroid, Reduce: update centroids

Join: Compute $R(A,B) \bowtie S(B,C)$, map: $R(a,b) \to (b,(a,R)), S(b,c) \to (b,(c,S))$, reduce: match same keys b output (a,b,c)

Communication cost: total I/O of all processes

Elapsed communication cost: max I/O along any path

Elapsed computation cost: max running time, i.e. total running time