Lecture 11: Graphs and Rankings

wbg231

January 2023

- about half this lecture was spent on material from the last lecture

1 page rank

- early search engines relied on matching query text
- pages were indexed and searched
- this was super game-able

spam attacks

- a website could just put a lot of popular search terms on there webpage and text matching search engine would rank that page highly even if it had no actual content

page rank uses the network

- the key insight is that the structure of the web has informative content
- web page publishers are more likely to publish links to pages they trust
- it is easy to make a spammy page,but hard to get others to link to it

the random surfer model

- think of the web as a directed graph nodes are pages, edges are links
- we model the users activity as a random walk
- that is given they are currently at page u they have a uniform chance of going to any page linked to by page u ie

$$P(u|v) = \frac{\text{number of links from u to v}}{\text{total number of links out of u}}$$

- uses are more likely to land in pages with high in degrees (that is linked to by many other pages)

markov chains

- Define a markov matrix $M \in \mathbb{R}^{N \times N}$ where n is the total number of pages we are considering and $M_{v,u} = P(v|u) =$ the likelihood of going from page u to page v
- M is a stochastic matrix ie it is non-negative and has columns that sum to 1
- let $p \in \mathbb{R}^n$ be a state or probability vector that that all enteries of p are non-negative and it sums to 1
- so $Mp \in \mathbb{R}^n$ is a marginal probability vector containing the probability of ending up at any page in the next step regardless of where you started this step (this is also a state vector)

steady state

- a steady state dissolution is a state vector $v : v = Mv$ that is a eigenvector associated to eigenvalue 1
- this will only occour if the graph is strongly connected and acyclic or (aperiodic) that is there are no points where it can get stuck in a loop (both of these are easy to force on the matrix)
- so $PageRank(u) = p[u] =$ the probability of random surfer being at node u in the steady state
- we can find this steady state with a nice property of eigenvectors with value 1 power iterations to find the steady state eigenvector that is regardless of the starting state of the distribution $p = lim_{t \to \infty}(Mv)^t$ where v is an arbitrary vector and p is our steady state matrix
- keep in mind when working in python $vals, vecs = np.linalg.eig(m)$ vecs will be ℓ_2 normalized that is $v = \frac{v}{||v||_2}$ (will hold) but probability distributions are ℓ_1 normalized that is $p = \frac{v}{||v||_1} \neq v$ so just divide the vector v by it's ℓ_1 norm

matrix vector multiplication parallelism

- note that each output vertex $Mp_v =< M[v], p >$ is independent $\iff$ $p_v = \lim_{t \to \infty}(M[u, :]u)^t$ is independent so we can Parallelize over the rows of our eigenvectors

- we only need to generate keys for the observed edges (that is if we know one web page does not lead to another we can ignore it)
- the number of vertices linking to webpage v is the in degree of v

the graph must be connected

- if the graph is not connected (meaning there must be one some path confecting all vector in at least one direction) the rank is less than the total number of pages and thus there is not a unique stationary distribution
- if a graph has no outgoing edges we can just say that all edges lead back ot it's self to keep the matrix well formed
- nodes that don't lead any where (called sinks or spider traps) will dominate the page rank steady state described above
- so if the graph is not strongly connected (that is there is a path going in and out of each node) or there are sinks (ie pages with no outgoing links) we can just add a teleportation parameter
- teleportation with probability a follow a link as normal, and with probability (1-a) jump to uniformly at random to a webpage
- that is at each step $M[u,v] = a * M[u,v] + (1-a)\frac{1}{n}$

page rank and search

- page rank scores based on the number of links pointing to a node, not the content of that node
- so a basic idea could be use some similarly search method to do a text search for candidate then rank those pairs using page rank

personalized page rank

- uniform teleportation is not realistic jumping to any page does not described how people work
- let e be our teleportation vector then our power iteration problem becomes

  $$p = (M')p = (a*M+(1-a)\frac{1}{n}ee^t)p = a*MP+(1-a)\frac{1}{N}ee^tp = a*MP+(1-a)\frac{1}{N}e = a*MP+(1-a)q$$

  where q is our personalization probability vector
- and q is the probabilty distribution of a user going to some set of sites they like

Efficiency of personalized page rank

- personalized page rank requires re-running power iteration to compute

$$p = a * Mp + (1 - a) * q$$

- if we fix q this gets a lot faster

Distributed page rank

- the core computation is matrix multiplication (can be done with map reduce) or spark pretty easily

social impacts of search

issues with page rank

- the ric get richer, a popular page will become more popular, it is hard for new pages to gain traction, similar to cold start
- link popularity may not be the same as accuracy
- steady state distributions are not how people behave in reality
- the model is not explainable users