Network and Graph Algorithms

Bruno Gonçalves

www.data4sci.com/newsletter

https://github.com/DataForScience/Networks

Networks and Graphs

Bibliography

Networks and Graphs

- Network: A system that is constituted by nodes (subcomponents) interconnected by links (connections, relationships, etc)

- Network vs Graphs: Roughly equivalent. "Network" is typically preferred by Physicists and Engineers while "Graph" is more common among Mathematicians.

- Mathematically, a network is a set of vertices V (nodes) and edges E (links).

- A link is a pair of nodes, $e = (v, v') \in E$

- In the case of undirected networks, the order of v and v'. does not matter. In the case of directed networks, (v, v') is a link from v to v'.
  If $(v, v') \in E$ and $(v', v) \in E$ the two nodes are reciprocally connected.

- In the case of weighted networks, links are also assigned with a weight function, characterizing the importance or weight of the link.

Facebook

Zachary Karate Club

Scientific Collaboration

Florentine Weddings

Face-to-Face Contact in a Primary School

The bridges of Koenigsberg

- Koenigsberg: A city in Germany famous for it's 7 bridges
- In 1735, it inspired the great mathematician Leonard Euler to consider a problem: Is it possible to find a walking path that crosses all 7 bridges exactly once before returning to the starting point?
- Euler quickly identified that the geometry of the islands and exact location of the bridges is irrelevant. Only their number and connections are important → the first Graph problem

The bridges of Koenigsberg

- Koenigsberg: A city in Germany famous for it's 7 bridges
- In 1735, it inspired the great mathematician Leonard Euler to consider a problem: Is it possible to find a walking path that crosses all 7 bridges exactly once before returning to the starting point?
- Euler quickly identified that the geometry of the islands and exact location of the bridges is irrelevant. Only their number and connections are important → the first Graph problem
- Solution: Such a path can only be found if two conditions are obeyed:
  - number of edges (bridges) must be even
  - either no or exactly 2 nodes can have an odd number of connections

No solution for Koenigsberg!

Mathematical Details

- Mathematical object, with set of nodes and edges
- Node - Individual Element
- Edge - Connection between element
- Degree - Number of edges connected to a node
- Weighted Edge - Edge with a weight associated
- Directed Edge - "One way street"

Graph Representations

- The fundamental concepts underlying graphs are simple: a set of items (nodes) and a list of connections (edges).

- Depending on the application many possible representations:

  - Edge List: Just the list of edges (assumes no disconnected nodes)

  - Adjacency Matrix: A matrix with non-zero elements only in the edges that are present (1, for unweighted networks, $\neq 1$ for weighted networks

  - Adjacency List/Dict: A list of all the nodes a given node is connected to (an empty list for disconnected nodes)

- Regardless of representation, if the network is undirected storage space can be saved by storing only one edge direction

Graph Representations

- Edge List: Just the list of edges (assumes no disconnected nodes). Common way to write graphs to file. Weights can be added as a third element in each edge.

$$\begin{bmatrix} (A,B) \\ (A,C) \\ (A,E) \\ (B,C) \\ (C,E) \\ (C,D) \\ (D,E) \end{bmatrix}$$

Graph Representations

- Adjacency Matrix: A matrix with non-zero elements only in the edges that are present (1, for unweighted networks, $\neq 1$ for weighted networks. Useful for analytical manipulations while inconvenient for practical use. Undirected networks result in symmetric matrices, while directed networks result in asymmetric ones

$$\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array}$$

- Adjacency List/Dict: A list of all the nodes a given node is connected to (an empty list for disconnected nodes). Similar concept to Python dictionary. List of connections can be expanded to include weights or any other data.

$$\begin{bmatrix} A: & B, C, E \\ B: & C \\ C: & D, E \\ D: & E \end{bmatrix}$$

Graph Properties

Components

- As we saw, a graph is a set of nodes connected by edges
- However, it is possible to have disconnected nodes (F)

Components

- As we saw, a graph is a set of nodes connected by edges
- However, it is possible to have disconnected nodes (F) and even larger disconnected parts of the network (E and D)
- Each piece of the graph is called a Component.
- When the largest component accounts for a substantial part of the nodes of the graph, it's called the Giant Connected Component.
- Graphs with a single component are called Connected
- In the case of directed graphs, we consider two cases:
  - Strongly-connected - Every node is accessible from every other node following the direction of the edges
  - Weakly-connected - Every node is accessible by disregarding edge directions

Degrees

- The most fundamental property of a node is its degree, k - the number of connections of a node

- In the case of directed networks, we distinguish between in-degree, k_{in} (number of incoming connections) and out-degree k_{out} (number of outgoing connections).

- Degree sequence - The ordered sequence of the degree of each node:

$$[3,2,4,2,3,0]$$

- Degree distribution - The relative frequency of each degree value:

$$P(k) = \frac{1}{N} \sum_{k_i \equiv k} 1$$

	Degree (k_i)
A	3
B	2
C	4
D	2
E	3
F	0

	P(k)
0	1/6
1	0
2	1/3
3	1/3
4	1/6

Degree Distributions

- Empirical Networks can have widely different degree distributions
- The most well known types are the Poissonian Distribution, Gaussian Distribution (Bell Curve) and Power-Law distribution (Broad tailed)
- Poissionian Distributions are characteristic of Random or Geometric Processes resulting in nodes having very similar numbers of connections
- Power-Law distributions are common in process where physical constraints aren't as important and processes such as Preferential Attachment (the rich get richer) can be more dominant
    - Results in the 80-20 rule - 80% of wealth is owned by just 20% of the population
    - Also popularize by Chris Anderson as "the long-tail" made possible by online markets

Degree Distributions

Barabasi (2002)

Degree Distributions

Poissonian

$$P(k) = \frac{\langle k \rangle^k}{k!} e^{-\langle k \rangle}$$

Gaussian

$$P(k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(k-\langle k \rangle)^2}{2\sigma^2}}$$

Power-law

$$P(k) = Ck^{-\gamma}$$

Fat-tailed Degree Distribution

Network or subgraph	Number of vertices	Number of edges	γ	References
Collaboration network of movie actors	212,250	61,085,555	2.3	[55]
'——' (another fitting of the same data)			3.1	[102]
Collaboration network of *Medline*	1,388,989	$1.028\ 10^7$	2.5	[13]
Collaboration net collected from mathematical journals	70,975	0.132×10^6	2.1	[15]
Collaboration net collected from neuro-science journals	209,293	1.214×10^6	2.4	[15]
Web of human sexual contacts[6]	2810	—	3.4	[132]

Nearest Neighbors and Correlations

- The nearest neighbors of a node are the ones to which it has (out-)links.
- The number of nearest neighbors is simply the (out-)degree of a node.
- The average degree of the nearest neighbors provides us with valuable information about the degree-degree correlations in the network.
- Let us define a matrix $E_{kk'}$ that simply counts how many edges there are between nodes of degree k and nodes of degree k'. This matrix has several useful properties:

$$\sum_{k} E_{kk'} = kN_k$$

$$\sum_{kk'} E_{kk'} = 2E = \langle k \rangle N$$

- We can use the last relation to find the joint probability:

$$P\left(k, k'\right) = \frac{E_{kk'}}{\langle k \rangle N}$$

Degree Correlations

- From which we can obtain the degree distribution (since $E_{kk'}$ contains all the connectivity information of the network).

$$P\left(k\right) = \frac{\langle k \rangle}{k} \sum_{k'} P\left(k, k'\right) \propto k^{-\gamma}$$

- The conditional probability that a node with degree k is connected to a node of degree k' is then:

$$P\left(k' | k\right) = \frac{\langle k \rangle}{k} \frac{P\left(k, k'\right)}{P\left(k\right)}$$

- Using this conditional probability we can easily calculate $\langle k_{nn}\left(k\right) \rangle$ the average degree of the nearest neighbors of a node of degree k

Degree Correlations

- The average degree of the nearest neighbors is then:

$$\langle k_{nn}(k)\rangle = \sum_{k'} k' P(k'|k)$$

- Depending on the behavior of this function, we classify networks into three classes:

Degree Correlations

- The average degree of the nearest neighbors is then:

$$\langle k_{nn}(k)\rangle = \sum_{k'} k' P(k'|k)$$

- Depending on the behavior of this function, we classify networks into three classes:

Social

Technological

Assortativity Coefficient

- Assortativity coefficient (r) - Pearson correlation coefficient of the degrees of the nodes at each end of an edge

- Positive r indicates that nodes of high (low) degree tend to connect preferentially to nodes of high (low) degree. Rich-club effect

- Negative r indicates that nodes of high (low) degree tend to connect preferentially to nodes of low (high) degree. Hierarchical/tree-like structure

- Properties:

  - $r \in [-1, +1]$

  - $r = +1$ - perfect assortative mixing

  - $r = 0$ - random (unpreferential/unassorted) mixing

  - $r = -1$ - perfect disassortative mixing

Empirical Network Assortativity

PRE 67, 026126 (2003)

	Group	Network	Type	Size n	Assortativity r	Error σ_r
	a	Physics coauthorship	undirected	52 909	0.363	0.002
	a	Biology coauthorship	undirected	1 520 251	0.127	0.0004
	b	Mathematics coauthorship	undirected	253 339	0.120	0.002
Social	c	Film actor collaborations	undirected	449 913	0.208	0.0002
	d	Company directors	undirected	7 673	0.276	0.004
	e	Student relationships	undirected	573	−0.029	0.037
	f	Email address books	directed	16 881	0.092	0.004
	g	Power grid	undirected	4 941	−0.003	0.013
Technological	h	Internet	undirected	10 697	−0.189	0.002
	i	World Wide Web	directed	269 504	−0.067	0.0002
	j	Software dependencies	directed	3 162	−0.016	0.020
	k	Protein interactions	undirected	2 115	−0.156	0.010
	l	Metabolic network	undirected	765	−0.240	0.007
Biological	m	Neural network	directed	307	−0.226	0.016
	n	Marine food web	directed	134	−0.263	0.037
	o	Freshwater food web	directed	92	−0.326	0.031

Friendship Paradox

EPJ Data Science 6, 4 (2017)

- Scott L. Feld, a sociologist noticed a strange phenomenon, in 1991: most people are less popular than their friends.

- This became known as the "Friendship Paradox" and is commonly described as meaning:

Your Friends Have More Friends Than You

- Since then, this paradox has been observed in many different empirical social networks, and even led to related paradoxes such as the Happiness Paradox

- A consequence of the combination of two factors:

  - Degree distributions with Hubs

  - Assortative mixing

Zachary Karate Club

- The social Network of a 70s Karate Club at a University
- A dispute among the two main instructors led to a split into two groups that was analyzed by Wayne Zachary in 1977.
- Good example of the Friendship Paradox and THE Classical example of Community Structure

Clustering Coefficient

- How likely are the friends of my friends to know each other?

- The possible number of pairwise combinations of k_i nodes is:

$$\binom{k_i}{2} \equiv \frac{k_i\left(k_i-1\right)}{2}$$

- And the clustering coefficient of node i is then:

$$C_i = \frac{2N_{\Delta i}}{k_i\left(k_i-1\right)}$$

where $N_{\Delta i}$ is the number of triangles of which node i is a part of (the number of my friends that are also friends of each other)

- Tendencially high in Social Networks:

	WWW	Citations	Co-author	Ham Radio	Prison	High School Romance
Number of Nodes	325729	396	81217	44	67	572
Randomness: r	0.57	0.63	4.7	5.0	∞	∞
Avg. In-Degree: m	4.6	5.0	.84	3.5	2.7	.83
Avg. Clustering	.11	.07	.16	.47	.31	-

typically one or more orders of magnitude higher than expected in a random graph

- An important factor in Community Structure

Network Modularity

- Empirically, it is clear that some network regions are denser than others
- Intuition:

  "Communities" should have more links within the community than without
- We define the modularity of a network as being:

$$Q = \sum_{i,j} \left[\frac{A_{ij}}{2m} - \frac{k_i k_j}{(2m)^2} \right]$$

- Where:
  - A_{ij} are the elements of the adjacency matrix
  - k_i, k_j are the degrees of nodes i and j, respectively
  - m is the total number of edges
  - The sum is taken only over nodes within the same community
- The modularity will be a large if all

Network Modularity

- Empirically, it is clear that some network regions are denser than others
- Intuition:

  "Communities" should have more links within the community than without

- We define the modularity of a network as being:

$$Q = \sum_{i,j} \left[\frac{A_{ij}}{2m} - \frac{k_i k_j}{(2m)^2} \right]$$

Probability that a node of degree k_i is connected to a node of degree k_j

- Where:
  - A_{ij} are the elements of the adjacency matrix
  - k_i, k_j are the degrees of nodes i and j, respectively
  - m is the total number of edges
  - The sum is taken only over nodes within the same community
- The modularity will be a large if all connected nodes belong to the same community

Community Structure

PNAS 103, 8577 (2006)

- Some networks are naturally divided into communities
- Finding communities in graphs is a hard problem with many real world applications:
  - Parallelization of Algorithms
  - Optimization of computer chip design
  - Distribution of MORPG players among game servers
  - Identifying social structure within a large social network
  - etc…

- Many community detection algorithms focus on different strategies to optimize the modularity for a given number of communities.
- See Phys. Rep.486, 75 (2010) for an extensive review of community detection algorithms

Paths and walks on Graphs

- Each edge represents a path connecting nodes i and j
- If we want paths of length 2, we must take two steps, and traverse two edges, so the total number of ways of going from node i to node k in two steps is:

$$p_{ik} = \sum_j a_{ij} a_{jk}$$

- where a_{ij} are the elements of the adjacency matrix. And in general:

$$P^{(n)} = A^n$$

- The diagonal elements of P indicate cycles, paths that start and end in the same node

$$A^2 = \begin{bmatrix} 3 & 1 & 2 & 2 & 1 \\ 1 & 2 & 1 & 1 & 2 \\ 2 & 1 & 4 & 1 & 2 \\ 2 & 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 & 3 \end{bmatrix}$$

Paths and walks on Graphs

- Computationally, we represent individual paths as sequences of nodes, so the path from 1 to 4 passing by 3 and 5 would be:

$$[1,3,5,4]$$

- A Path has no constraints on length other than the fact that it can only traverse each edge at most once

- A Walk is similar to a path, but has no limitations on the number of times it visits a given node or edge

- The Diameter of a graph is the average shortest path between all pairs of nodes

Network Diameter

- In 1967, the Social Psychologist Stanley Milgram decided to conduct an experiment to answer a simple question:

- How many friendship "steps" separate, on average, any two people in the US?

Network Diameter

- Choose participants in Kansas and Nebraska and ask them to try to send a letter to a stock broker in Boston, Massachusetts
- Pass letter in hand to someone they knew well and they think is likely to know the target
- Only 25% of the letters reached their destination
- Attrition causing longer chains to be down sampled
- Self-selection bias (seeds were chosen by answering a journal ad)
- Disconnected components?

6 degrees of separation

In the Nebraska Study the chains varied from two to 10 intermediate acquaintances with the median at five.

Six-Degrees of Kevin Bacon

oracleofbacon.org

Six-Degrees of Kevin Bacon

oracleofbacon.org

Bipartite Graph

Facebook Diameter

WebSci'12, 33 (2012)

- Is Facebook making the world smaller?
- 1 Billion nodes
- 4-degrees of separation
- Based on shortest path

Edge Weight

- So far we have focused exclusively on the number of edges, but edges can also have weights

- Edge weights can represent:
  - connection strength (close friend vs acquaintance)
  - connection frequency (daily calls vs yearly meetings)
  - physical distance (flight time between cities)
  - etc…
- Node Strength:
  - The sum of the weights of all the nodes connected to the node
  - Weighted equivalent to node degree
- Weights are strongly related with network topology!

Strength of Weak Ties

- In 1973, Mark Granovetter was interested in studying the job market.
- He performed interviews to find out how individuals found out about job opportunities.
- The results were surprising, as most people found out about their current jobs from acquaintances or friends of friends with whom they hadn't interacted with recently.
- This lead to a fascinating conclusion:

"It is argued that the degree of overlap of two individuals social networks varies directly with the strength of their tie to one another"

- Close relationships (strong ties) are more predominant within communities and valuable information can be gained by interacting with people outside your community (weak ties).

Strength of Weak Ties

FIG. 2.—Local bridges. *a*, Degree 3; *b*, Degree 13. ——— = strong tie; — — — = weak tie.

Degree Weight Correlations

PNAS 103, 2015 (2006)

- Strong correlations between degree and weights are also possible in specific kinds of networks
- A common example is the Airline Network that connects airports around the world
- The number of connections an airport has is strongly dependent on it's global importance
- Flights to important airports carry more passengers

Degree Weight Correlations

PNAS 103, 2015 (2006)

- The number of seats per connection (weight) depends just on the degrees:

$$\omega_{kk'} \propto \omega_0 \left(kk'\right)^{\theta}$$

- While the total traffic (strength) per node depends just on the degree:

$$T_k \propto Ak^{\theta+1}$$

- Relations between degrees and weights are common in transportation and distribution networks

Graph Algorithms

Erdős-Rényi Graph

- One of the simplest graph models
- Proposed in 1959 by two Hungarian mathematicians Paul Erdős and Alfréd Rényi.
- Start with N disconnected nodes
- Connect each pair of nodes with probability $p \ll 1$
- Each possible edge is present with probability p, so the average number of edges is:

$$E = \frac{p}{2} N (N-1)$$

- And the average degree is:

$$\langle k \rangle = \frac{2E}{N} = p\,(N-1)$$

- So to build a network with a give average degree, we need:

$$p = \frac{\langle k \rangle}{N}$$

Erdős-Rényi Graph

- With some algebra it can be shown that the degree distribution is:

$$P(k) = \frac{\langle k \rangle^k}{k!} e^{-\langle k \rangle}$$

- So all nodes have similar degrees -> No hubs!

Watts-Strogatz Model

- In 1998, Duncan J. Watts (Australian) and Steven Strogatz (American) proposed a simple model that was able to interpolate between a completely regular graph and an Erdős-Rényi type Random Graph

- Their model starts with a regular graph where each node is connected to its K nearest neighbors.

- Each edge is then rewired at random with probability p

Watts-Strogatz Model

- The original regular graph has a very **large clustering** coefficient, while the Erdős-Rényi Graph has almost no clustering.
- On the other hand, the regular graph has a very large diameter while the ER graph has a very small diameter
- The rewiring mechanism introduces long-distance edges that quickly shorten the graph diameter while at the same time still preserving the large clustering
- This intermediate state is known as a "Small-World" and it was the first time that a model was able to explain it.
- However, the degree distribution is exponential and still very far from real networks

Small-World Effect: Watts-Strogatz Model

Barabási-Albert Model

- In 1999 two Hungarian Physicists, Albert-László Barabási and Réka Albert proposed the Barabási-Albert model.

- It introduced the concept of Preferential Attachment as a mechanism to explain the origin of broad tail distributions observed in real world networks

- At each "time" step add a new node and connect it to m_0 previously existing nodes chose with probability:

$$\Pi_k = \frac{k_i}{\sum_i k_i}$$

- If $m_0 = 1$ the graph is a tree

- It's intrinsically unfair, with "oldest" nodes having higher degrees.

Barabási-Albert Model

- The Degree Distribution is a power-law:

$$P\left(k\right) \propto k^{-\gamma}$$

- So it contains large numbers of hubs

- It's average shortest path length is approximately logarithmic:

$$l \propto \frac{\ln N}{\ln \ln N}$$

Viral Spreading

- To get a better understanding of the differences among the various types of graphs let's consider a "simple" problem: Epidemic Spreading
- The simplest epidemic model is the Susceptible-Infectious (SI) model
- Susceptible (healthy) individuals become infectious (with a given probability β) when coming in contact with Infectious (sick) individuals

- With this simple mechanism everyone will eventually become infectious, but the graph properties will determine how the disease will progress

Viral Spreading - ER

Viral Spreading - WS

Connectivity Patterns Are Important!

Random Sampling

- You can consider the previous epidemic example as a way of sampling the connectivity patterns of a network. By observing how the infection spreads from one node to another we gain information about how nodes are connected to each other

- Sampling approaches are particularly useful when we are able to interact with a network but not able to observe it in its entirety.

- A generalization of this approach is known as Random Sampling where we imagine ourselves to be a drunkard trying to find our way home.

- At each step you must randomly chose one edge to follow

- And repeat the process upon reaching the new node.

- At most one new node/edge is visited at each time step but clustering and community structure can "trap" the walker for significant amounts of time

- The number of times a given node is visited is a measure of how "central" it is on the network

Snowball Sampling

- One improvement on Random Sampling is known as Snowball Sampling
- At each step visit ALL neighbors from your current location
- Commonly used in Social Science and Computer Science

    1. Start with a single node (or small number of nodes)
    2. Get the list of nearest neighbors
    3. For each neighbor get the nearest neighbors list
    4. Repeat for a fixed number of layers or until enough users have been connected

- Generates a connected component from each seed
- At each step a new "layer" is added to the network. The number of layers required to reach any node from the starting seed is the length of the shortest path connecting those two nodes

Random Walks and Markov Chains

- If we do have a complete view of the network, we can simulate a random walk on it by using a Markov Chains

- A Markov Chain is a mathematical process that is equivalent to the average behavior of a large number of random walk "repetitions"

- Markov Chain rely on a matrix representation of the connections where the edge weights are the probability of that edge being chosen. In particular, Markov Chains are:

  - Memoryless - "Future depends only on present state and not on past history"

- Have a stationary state if the connectivity matrix is:

  - Stochastic - Column vectors normalized to 1 (We always leave the current node)

  - Irreducible - All nodes are accessible (graph is connected)

  - Aperiodic - Return to node i does not occur periodically

- The stationary probability distribution across all nodes is the relative importance of each

PageRank

- "A page is important if it is pointed to by other important pages"

$$\pi_i = \sum_j a_{ji} \frac{\pi_j}{k_j}$$

- Stationary state of a discrete time random walk

$$\vec{\pi}\,(t+1) = \mathcal{T}\vec{\pi}\,(t)$$

$$\mathcal{T} = K_o^{-1} A^T$$

- Problems:

"Dangling nodes"

Connect dangling nodes to every other node

Slow convergence

Add damping factor

Power Method

$$\vec{\pi} = \mathcal{G}\vec{\pi}$$

Eigenvector corresponding to Eigenvalue 1.

Iterate...

$$\vec{\pi}^{(n+1)} = \mathcal{G}\vec{\pi}^{(n)}$$

Stop when:

$$\vec{\pi}^{(n+1)} \approx \vec{\pi}^{(n)}$$

Example:

Web Graph

Random-Walk Sample

- Perform a random walk on the graph
- Common in temporal networks:
  - Observe one node at a time
  - Each step corresponds to an edge traversal
- The natural way of "passively" observing different kinds of empirical networks
  - Cell phone calls
  - Social media retweets
  - etc
- Naturally slow to obverse the full graph

Snowball Sampling

- Commonly used in Social Science and Computer Science

    1. Start with a single node (or small number of nodes)
    2. Get “friends” list
    3. For each friend get the “friend” list
    4. Repeat for a fixed number of layers or until enough users have been connected

- Generates a connected component from each seed
- Quickly generates a *lot* of data/API calls

Graph Traversal Algorithms

Shortest paths and Distances

https://github.com/gboeing/osmnx

- A common problem in graph analysis is to determine the shortest path between two given nodes:
  - Minimum number of hops on a computer network

Shortest paths and Distances

https://github.com/gboeing/osmnx

- A common problem in graph analysis is to determine the shortest path between two given nodes:
  - Minimum number of hops on a computer network
  - Fastest drive path between two different locations on a city
  - etc
- There's a very rich literature on algorithms to identify the shortest paths and to measure the shortest distance between two nodes on a graph. We'll cover two in detail:
  - Dijkstra's Algorithm
  - Floyd-Warshall Algorithm

Dijkstra's Algorithm

- Algorithm created by Edsger W. Dijkstra, a Dutch computer scientist, in 1956 to find the shortest paths between nodes on a weighted graph

- Most common formulation finds the shortest path between a source node and every other node on the graph

- Dijkstra's algorithm proceeds in an incremental way in which we continuously update our best estimate for the distance between the origin and every other node. Define each node to be at an unknown (or infinite) distance from the origin. The origin is defined to be at distance 0 from itself.

en.wikipedia.org/wiki/Dijkstra's_algorithm

Dijkstra's Algorithm

- At each step:
  - Find the node i with the smallest current distance to the origin
  - Update the distance to each of it's nearest neighbors, j

$$dist\left(node_j\right) = \min\left[dist\left(node_j\right), dist\left(node_i\right) + w_{ij}\right]$$

  - Continue until the destination has been reached or we have calculated the distance to every other node
  - The shortest path can be calculated as well by tracking which was the node that resulted in a distance update

en.wikipedia.org/wiki/Dijkstra's_algorithm

Dijkstra's Algorithm

- Assumes weighted graphs
  - Unweighted graphs can be handled by assigning a weight of 1 to each edge
- All weights must be positive. Negative weights will cause infinite loops as you can always reduce your current "distance" estimate by traversing a negative weight edge.
- Can be particularly slow on some topologies.
- In unweighted graphs, it becomes similar to Breath First Search

en.wikipedia.org/wiki/Dijkstra's_algorithm

Encoding Paths

- As we saw in the case of Dijkstra's algorithm, we can recover the shortest path by keeping track of the node that causes a distance update:

$$dist\left(node_j\right) = \min\left[dist\left(node_j\right), dist\left(node_i\right) + w_{ij}\right]$$

- In this case, if $dist\left(node_j\right) > dist\left(node_i\right) + w_{ij}$ we update the "parent" node of $node_j$ to be $node_i$

- At the end, we will have an matrix $P = \left[p_{ij}\right]$ where the value of p_{ij} is the next step along the path from i to j.

- If i is directly connected to j then $p_{ij} \equiv j$ and, consequently, $p_{ii} = i$

Encoding Paths

- For this simple case we have:

$$P = \begin{bmatrix} 0 & 1 & 2 & 2 & 4 \\ 0 & 1 & 2 & 2 & 0 \\ 0 & 1 & 2 & 3 & 4 \\ 4 & 2 & 2 & 3 & 4 \\ 0 & 0 & 2 & 3 & 4 \end{bmatrix}$$

- From where we can recover the shortest path between D and A as $D \rightarrow E \rightarrow A$ since $p_{30} = 4$ and $p_{40} = 0$

- This kind of representation is fairly common in computer networks and serves as the basis for routing traffic on the internet. See https://en.wikipedia.org/wiki/Border_Gateway_Protocol. In this way, each node needs only to know which of it's nearest neighbors lies along the shortest path to any other destination. When the request reaches the next node, it can use any local updates that that node has received about how to move on to the next step.

Floyd-Warshall Algorithm

- The Floyd-Warshal Algorithm improves on Dijkstra's Algorithm that is able to handle negative weights as well as computing the full all-to-all paths and distances.

- It uses dynamic programming techniques to update the shortest path estimate between any two vertices, until the estimate is optimal.

- The Floyd-Warshal Algorithm uses two $N \times N$ matrices:
  - a distance matrix
  - a path matrix

- Start by initializing the distance and path matrices with the nearest neighbor information

- For each pair of nodes update the distance and path matrices by trying out every possible intermediate node

$$dist\left(node_i, node_j\right) = \min\left[dist\left(node_i, node_j\right), dist\left(node_i, node_k\right) + dist\left(node_k, node_i\right)\right]$$

en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm

Floyd-Warshall Algorithm

en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm

Applications to Empirical Networks

Temporal Networks

- Some networks change and evolve over time:
  - nodes joining and leaving
  - intermittent edges
  - edges being created and destroyed
- Examples:
  - Cell phone call network - edges only present during call
  - Disease spreading network - virus can only spread during face-to-face presence
  - peer to peer file sharing network - sharing only possible while both nodes are present
- Can be modeled as a different network (nodes and edges) at each point in time
- Aggregate network can behave very differently than individual networks

Temporal Networks

Multi-layer networks

- In some cases, it's possible to have multiple kinds of interactions across the same same set of nodes
- Each interaction type defines one different layer of the the network
- Information can spread within any layer and, sometimes, across layers
- Examples:
  - Online social networks - Like, share, comment, direct message, etc…
  - "Real world" social networks - face-to-face, email, call, sms, etc…
  - Travel networks - subway, train, bus, bike, etc…
  - Computerized power grid - Power network and Computer control network
- Network properties are strongly impacted by the interaction between the different layers
- Each layer tells only part of the story

Multi-layer networks

Multi-layer networks

Bipartite networks and Recommender Systems

- Two kinds of nodes
- Edges only allowed between nodes of different kinds
- Typically represent affiliation or collaboration:
  - actors in movies
  - authors in papers
  - etc
- Usually the first step is to project them to a single node type
- Projected networks contain similarity information for each kind of node

Bipartite networks and Recommender

$$A = \begin{array}{cc} & y \\ x & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$X = AA^T \qquad Y = A^T A$$

Bipartite Networks - Similarities

We can treat the adjacency matrix as a feature matrix!

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

And generate a new matrix based on the similarities between items

Bipartite Networks - Similarities

Dense

4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

Sparse

1.00	0.00	0.58	0.00	0.67	0.58
0.00	1.00	0.00	0.41	0.00	0.00
0.58	0.00	1.00	0.00	0.58	0.75
0.00	0.41	0.00	1.00	0.00	0.00
0.67	0.00	0.58	0.00	1.00	0.58
0.58	0.00	0.75	0.00	0.58	1.00

1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00

RECOMMENDER SYSTEM ALGORITHMS

INPUT

4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

Introduction to Recommender Systems
Machine learning Paradigms
Social Network-based Recommender Systems
Learning Spark
Recommender Systems Handbook
Recommender Systems and the Social Web

USER-BASED COLLABORATIVE FILTERING

1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00

(0.7 x) + (0.6 x) =

already rated by user

(0.7 x 4 + 0.6 x 5) / (0.7 + 0.6) = 4.5

(0.6 x 3) / 0.6 = 3.0

already rated by user

ITEM-BASED COLLABORATIVE FILTERING

1.00	0.27	0.79	0.32	0.98	0.00
0.27	1.00	0.00	0.00	0.34	0.65
0.79	0.00	1.00	0.69	0.71	0.18
0.32	0.00	0.69	1.00	0.32	0.49
0.98	0.34	0.71	0.32	1.00	0.00
0.00	0.65	0.18	0.49	0.00	1.00

(4 x) + (3 x) + (5 x) =

already rated by user

(0.8 x 4 + 0.7 x 5) / (0.8 + 0.7) = 4.5

already rated by user

(0.7 x 3) / 0.7 = 3.0

CONTENT-BASED FILTERING

1.00	0.00	0.58	0.00	0.67	0.58
0.00	1.00	0.00	0.41	0.00	0.00
0.58	0.00	1.00	0.00	0.58	0.75
0.00	0.41	0.00	1.00	0.00	0.00
0.67	0.00	0.58	0.00	1.00	0.58
0.58	0.00	0.75	0.00	0.58	1.00

(4 x) + (3 x) + (5 x) =

already rated by user

(0.4 x 3) / 0.4 = 3.0

already rated by user

(0.6 x 4 + 0.6 x 5) / (0.6 + 0.6) = 4.5

HYBRID

0.4 x UB CF) + (0.3 x IB CF) + (0.3 x CB) =

(0.4 x 4.5 + 0.3 x 4.5) / (0.4 + 0.3) = 4.5

(0.3 x 3.0 + 0.3 x 4.5) / (0.3 + 0.3) = 3.8

(0.4 x 3.0 + 0.3 x 3.0) / (0.4 + 0.3) = 3.0

OUTPUTS

FOUR RECOMMENDER ALGORITHMS ARE FED THE SAME INPUT AND PRODUCE DIFFERENT OUTPUTS.
THEIR DEFINITIONS OF USER AND/OR ITEM SIMILARITY ACCOUNT FOR MOST OF THE DIFFERENCES.

https://buildingrecommenders.wordpress.com

Events

data4sci.com/newsletter

Data Visualization with matplotlib and seaborn
Dec 4, 2019 - 5am-9am (PST)

Deep Learning From Scratch
Dec 11, 2019 - 5am-9am (PST)

Natural Language Processing (NLP) from Scratch
http://bit.ly/LiveLessonNLP - On Demand

END