§4 Link Analysis

Qin Zhang
Learn the structure of the web
Motivations

Learn the structure of the web
Motivations

Learn the structure of the web

Power Law graphs
Can be generated by preference attachment
Markov Chains
Markov chain provides important life lessons

- Only your current position matters going forward, don’t worry about the past.
Life lessons

Markov chain provides important life lessons

• Only your current position matters going forward, don’t worry about the past.

• You just need to worry about one step at a time; you will get there eventually (or you won’t).
Markov chain provides important life lessons

- Only your current position matters going forward, don’t worry about the past.
- You just need to worry about one step at a time; you will get there eventually (or you won’t).
- In the limit, everyone has perfect karma.
Graph: $G = (V, E)$ is defined by a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E = \{e_1, e_2, \ldots, e_m\}$ where each edge $e_j$ is an unordered (or ordered in a directed graph) pair of edges $e_j = \{v_i, v_{i'}\}$ (or $e_j = (v_i, v_{i'})$).

Example:

Adjacent list representation

$$A = \begin{pmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}$$
A **Markov Chains** \((V, M, q)\) is defined by a set of nodes \(V\), a probability transition matrix \(M\), and an initial state \(q\).
A Markov Chains $(V, M, q)$ is defined by a set of nodes $V$, a probability transition matrix $M$, and an initial state $q$.

The initial state $q$ represents a probability distribution over which nodes we are located. For instance, if we are at state $b \in V$ (with probability 1) then $q^T = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$. 
A **Markov Chains** \((V, M, q)\) is defined by a set of nodes \(V\), a probability transition matrix \(M\), and an initial state \(q\).

The **initial state** \(q\) represents a probability distribution over which nodes we are located. For instance, if we are at state \(b \in V\) (with probability 1) then \(q^T = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]\).

Example: If we have a 10% chance of being in state \(a\), a 30% chance of being in state \(d\) and a 60% chance of being in state \(f\), then \(q^T = [0.1, 0, 0, 0, 0.3, 0, 0.6, 0, 0, 0]\).
A **Markov Chains** \((V, M, q)\) is defined by a set of nodes \(V\), a probability transition matrix \(M\), and an initial state \(q\).

The **initial state** \(q\) represents a probability distribution over which nodes we are located. For instance, if we are at state \(b \in V\) (with probability 1) then \(q^T = [0, 1, 0, 0, 0, 0, 0, 0, 0]\).

Example: If we have a 10% chance of being in state \(a\), a 30% chance of being in state \(d\) and a 60% chance of being in state \(f\), then \(q^T = [0.1, 0, 0, 0, 0.3, 0, 0.6, 0, 0]\).

In general we need to enforce that
1. Each \(q[i] \geq 0\)
2. \(\sum_i q[i] = 1\)
The **transition matrix** $M$ can be described as the normalized adjacently matrix. That is, each column $M_j = A_j / \|A_j\|_1$.

\[
M = \begin{pmatrix}
0 & 1/2 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\
1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\
1/3 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 0 \\
1/3 & 1/2 & 1/3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/3 & 0 & 0 & 1/3 & 1/2 & 0 \\
0 & 0 & 0 & 0 & 1/3 & 0 & 1/2 & 1 \\
0 & 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 \\
\end{pmatrix}
\]
Markov Chains (cont.)

The **transition matrix** $M$ can be described as the normalized adjacently matrix. That is, each column $M_j = A_j / \| A_j \|_1$.

$$M = \begin{pmatrix}
0 & 1/2 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\
1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\
1/3 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 0 \\
1/3 & 1/2 & 1/3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/3 & 0 & 0 & 1/3 & 1/2 & 0 \\
0 & 0 & 0 & 0 & 1/3 & 0 & 1/2 & 1 \\
0 & 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0
\end{pmatrix}$$

That is, each column $M_j = A_j / \| A_j \|_1$.

Now, given $q_0 = [0, 1, 0, 0, 0, 0, 0, 0, 0]^T$, we get

$$q_1 = Mq_0 = [1/2, 0, 0, 1/2, 0, 0, 0, 0]^T,$$ and further

$$q_2 = Mq_1 = MMq_0 = [1/6, 2/6, 2/6, 1/6, 0, 0, 0, 0]^T$$
Two ways of thinking

- It describes a random walk of a point starting at \( q \) (or in some state distribution described by \( q \)).
Two ways of thinking

- It describes a random walk of a point starting at \( q \)
  (or in some state distribution described by \( q \)).

  At each step it decides where to go next randomly based on the column of \( M \) describing the column its state corresponds to. It moves to exactly one new state. Then repeat.

- It describes probability distribution of a random walk.

  At each state, we only track the distribution of where it might be: this is \( q_n \) after \( n \) steps.

  Alternatively, we can consider \( M^n \), then for any initial state \( q_0 \), \( M^n q_0 \) describes the distribution of where \( q_0 \) might be after \( n \) steps.

  So entry \( M^n_{j,i} \) describes the probability that a point starting in \( j \) will be in state \( i \) after \( n \) steps.
A Markov chain is **ergodic** if \( \exists \) some \( t \) such that for all \( n \geq t \), then each entry in \( M^n \) is positive.
A Markov chain is **ergodic** if \( \exists \) some \( t \) such that for all \( n \geq t \), then each entry in \( M^n \) is positive.

**When is a Markov chain not ergodic?**
A Markov chain is **ergodic** if \( \exists \) some \( t \) such that for all \( n \geq t \), then each entry in \( M^n \) is positive.

**When is a Markov chain not ergodic?**

- It is **cyclic**. This means that it alternates between different sets of states every few steps. 

\[
\begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0 \\
\end{pmatrix}
\]
Ergodic

A Markov chain is **ergodic** if $\exists$ some $t$ such that for all $n \geq t$, then each entry in $M^n$ is positive.

**When is a Markov chain not ergodic?**

- It is **cyclic**. This means that it alternates between different sets of states every few steps. 

\[
\begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0
\end{pmatrix}
\]

- It has **absorbing and transient** states (only for directed graphs). If a random walk leaves some node in $T$ and lands in a state in $A$, then it never returns to any state in $T$. In this case, the nodes $A$ are absorbing, and the nodes in $T$ are transient.

\[
\begin{pmatrix}
0 & 1 & 0 \\
1 & 0 & 1 \\
0 & 0 & 0
\end{pmatrix}
\]
A Markov chain is **ergodic** if $\exists$ some $t$ such that for all $n \geq t$, then each entry in $M^n$ is positive.

**When is a Markov chain not ergodic?**

- **It is cyclic.** This means that it alternates between different sets of states every few steps. 
  $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$

- **It has absorbing and transient states** (only for directed graphs). If a random walk leaves some node in $T$ and lands in a state in $A$, then it never returns to any state in $T$. In this case, the nodes $A$ are absorbing, and the nodes in $T$ are transient. 
  $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$

- **It is not connected.** There are two sets of notes $A, B \subset V$ such that there is no possible way to transition from any node in $A$ to any node in $B$. 
  $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Property For an ergodic Markov Chain, let $M^* = M^n$ as $n \to \infty$ (will converge). Then for any starting state distribution $q$, we have $M^* q = q^*$ for some state distribution $q^*$.

Call $q^*$ the stable distribution
Problem: Each weight $v \in V$ has a weight $w(v)$ associated with it. Let $W = \sum_{v \in V} w(v) = W$. We want to sample a $v \in V$ with probability $w(v)/W$. 
**Problem:** Each weight $v \in V$ has a weight $w(v)$ associated with it. Let $W = \sum_{v \in V} w(v) = W$. We want to sample a $v \in V$ with probability $w(v)/W$.

**Difficulty**

- $V$ might be very large, and $W$ unknown.
- $V$ can be continuous, so $W = \int_{v \in V} w(v) \, dv$, even more difficult to compute.
**Problem:** Each weight \( v \in V \) has a weight \( w(v) \) associated with it. Let \( W = \sum_{v \in V} w(v) = W \). We want to sample a \( v \in V \) with probability \( w(v)/W \).

**Difficulty**
- \( V \) might be very large, and \( W \) unknown.
- \( V \) can be continuous, so \( W = \int_{v \in V} w(v)dv \), even more difficult to compute.

**Solution.** Design a special Markov chain so that \( q^*[v] = w(v)/W \) without knowing \( W \).
Metropolis Algorithm

1. Initialize $v_0$ to be an arbitrarily element in $V$.
2. Repeat
   
   Generate $u \sim K(v, \cdot)$.
   
   if $w(u) \geq w(v_i)$ then set $v_{i+1} = u$
   
   else with probability $w(u)/w(v)$, set $v_{i+1} = u$
   
   else set $v_{i+1} = v_i$

   Until “coverage”

Comments

1. $K$ is some notion of neighborhood/similarity. E.g., follow some transition matrix $M$.
2. The algorithm implicitly defines a Markov chain on the state space $V$. Thus $\exists$ some $t$ s.t. $i \geq t$, we have $\Pr[v_i = v] = w(v)/W$.
3. Officially, run for $t$ steps, take 1 sample; repeat. In practice, run for 1000 steps ("burn in"), take next 5000 steps as random samples.
Metropolis Algorithm

1. Initialize $v_0$ to be an arbitrarily element in $V$

2. Repeat
   
   Generate $u \sim K(v, \cdot)$. (If in a graph, think of a random neighbor)
   
   if $w(u) \geq w(v_i)$ then set $v_{i+1} = u$
   
   else with probability $w(u)/w(v)$, set $v_{i+1} = u$
   
   else set $v_{i+1} = v_i$

   Until “coverage”

Proof on board

Comments

1. $K$ is some notion of neighborhood/similarity. E.g., follow some transition matrix $M$.

2. The algorithm implicitly defines a Markov chain on the state space $V$. Thus $\exists$ some $t$ s.t. $i \geq t$, we have $\Pr[v_i = v] = w(v)/W$.

3. Officially, run for $t$ steps, take 1 sample; repeat. In practice, run for 1000 steps (“burn in”), take next 5000 steps as random samples.
Webpage Search
First question: define a similarity function

- **Take 1**: Using the cosine similarity between keywords searched and the set of words on the webpage.
First question: define a **similarity function**

- **Take 1**: Using the cosine similarity between keywords searched and the set of words on the webpage.

How could this be fooled?
Webpage similarity

First question: define a **similarity function**

- **Take 1**: Using the cosine similarity between keywords searched and the set of words on the webpage.

How could this be fooled?

Repeat the word pie 1000 times at the bottom of the page. Change the color of the text to the background color of the page, make it really small.
Webpage similarity

First question: define a **similarity function**

- **Take 1**: Using the cosine similarity between keywords searched and the set of words on the webpage.

  How could this be fooled?

  Repeat the word pie 1000 times at the bottom of the page. Change the color of the text to the background color of the page, make it really small.

- **Take 2**: Find pages include “pie”, but not too many times, maybe also include words typically occur together with word “pie”, e.g., apple, pan, hot, delicious, ...
Webpage similarity

First question: define a similarity function

- **Take 1**: Using the cosine similarity between keywords searched and the set of words on the webpage.

  How could this be fooled?

  Repeat the word pie 1000 times at the bottom of the page. Change the color of the text to the background color of the page, make it really small.

- **Take 2**: Find pages include “pie”, but not too many times, maybe also include words typically occur together with word “pie”, e.g., apple, pan, hot, delicious, ...
First question: define a similarity function

- **Take 1**: Using the cosine similarity between keywords searched and the set of words on the webpage.

  How could this be fooled?

  Repeat the word pie 1000 times at the bottom of the page. Change the color of the text to the background color of the page, make it really small.

- **Take 2**: Find pages include “pie”, but not too many times, maybe also include words typically occur together with word “pie”, e.g., apple, pan, hot, delicious, ... 

  How could this be fooled?

  Find the top several ranked pages, copy their source at the bottom of your page.
Idea 1: Pages are important if they are linked to by other important pages

Sounds like a chicken-egg problem, but seems very natural
Idea 1: Pages are important if they are linked to by other important pages

Sounds like a chicken-egg problem, but seems very natural

Idea 2: A page is important if a “random surfer” were likely to find it.

A random surfer starts on some page, clicks a random link on that page, and then goes to the next page. This continues (similar to a crawler).
Idea 1: Pages are important if they are linked to by other important pages

Sounds like a chicken-egg problem, but seems very natural

Idea 2: A page is important if a “random surfer” were likely to find it.

A random surfer starts on some page, clicks a random link on that page, and then goes to the next page. This continues (similar to a crawler).

The random surfer defines a Markov chain, which converges to distribution $q^* = M^* q$, gives the importance $q^*[v]$ to a webpage $v \in V$.

We come up with a ranking function

$$\text{Rank}(v, \text{query}) = \text{Magic}(q^*[v], \text{query}, \text{text}(v), \text{text}(e(v', v), q^*[v']))$$
Potential attack: Spam Farms – A way to fool Page Rank

1. We create a few *target* pages (which we want Google to rank highly)
2. We create many *corrupted* pages, and send links to target pages.
3. We have few target links to each other.

(Graph on board)
Potential attack: Spam Farms – A way to fool Page Rank

1. We create a few *target* pages (which we want Google to rank highly)
2. We create many *corrupted* pages, and send links to target pages.
3. We have few target links to each other.

(Graph on board)

Then how can Google defeat spam farms?
Spam Farms

**Potential attack:** Spam Farms – A way to fool Page Rank

1. We create a few *target* pages (which we want Google to rank highly)
2. We create many *corrupted* pages, and send links to target pages.
3. We have few target links to each other.

(Graph on board)

Then how can Google defeat spam farms?

1. Search for structures to identify and blackball such spam farms. (Not an easy task, still an ongoing battle)
2. Give trust weights.
   - High: Wiki, .edu, .mil, .gov, and very high PageRank pages
   - Low: blogs, twitter, pages with comments
   
   ...
Random Walk Enough?

- **IN** 44 million pages
- **Central core** 56 million pages
- **OUT** 44 million pages
- **Tendrils and tubes** 44 million pages
- **Disconnected components** 17 million pages
Page Rank Algorithm

We start, say, from Google.com \( q_0 = [0, \ldots, 0, 1, 0, \ldots, 0] \). In the iterative step, we compute a new vector estimate of PageRanks \( q_{i+1} \) from the current PageRanks estimate \( q_i \), and the transition matrix \( M \)

\[
q_{i+1} = \beta M q_i + (1 - \beta) e / n,
\]

where \( \beta \) is a chosen constant, usually in \([0.8, 0.9]\), \( e \) is a vector of all 1’s, and \( n \) is the total number of nodes in the web graph.

The first part is just a random walk, and the second part is a new random surfer at a random page, to avoid deadlocks.
Personalized Page Rank Algorithm

We start, say, from Google.com $q_0 = [0, \ldots, 0, 1, 0, \ldots, 0]$. In the iterative step, we compute a new vector estimate of PageRanks $q_{i+1}$ from the current PageRanks estimate $q_i$, and the transition matrix $M$

$$q_{i+1} = \beta M q_i + (1 - \beta) e/n,$$

where $\beta$ is a chosen constant, usually in $[0.8, 0.9]$, $S \subseteq [n]$, $e_S$ is a vector that has 1 in all coordinates in $S$, and 0 elsewhere. $S$ could be, for example, webpages on a certain topic.
HITS (Hypertext-Induced Topic Selection)

Next few slides are borrowed from Leskovec’s course
SALSA: similar as HITS, assign two scores to each node \( v \), called the **hub score** \( h_v \), and the **authority score** \( a_v \).

The two scores are related to each other.

\[
  h_v = \sum_{(v,x) \in E} \frac{a_x}{\text{indeg}(x)}, \quad a_x = \sum_{(v,x) \in E} \frac{h_v}{\text{outdeg}(v)}
\]
SALSA: similar as HITS, assign two scores to each node \( v \), called the hub score \( h_v \), and the authority score \( a_v \).

The two scores are related to each other.

\[
\begin{align*}
  h_v &= \sum_{(v, x) \in E} \frac{a_x}{\text{indeg}(x)}, \\
  a_x &= \sum_{(v, x) \in E} \frac{h_v}{\text{outdeg}(v)}
\end{align*}
\]

A forward-backward random walk, where the walk alternates between forward and backward steps.