

Lecture #8:

Markov Chain Applications

Randy Cogill
SYS 605 - Stochastic Systems
Fall 2007

2

Google's PageRank

- **PageRank** was first major innovation of the Google search engine
- Give every web page an 'importance' score...
- ...return search results ranked by importance
- **Idea:** Relevant factors in ranking a web page:
 - How many pages link to it?
 - How important are the pages linking to it?
 - How 'diluted' is each incoming link?

Google's PageRank (cont.)

- The rank of page i is defined as

$$[\pi]_i = \sum_{j \in \mathcal{I}(i)} \frac{[\pi]_j}{|\mathcal{O}(j)|}$$

- $\mathcal{I}(i)$ is the set of pages linking to page i
- $\mathcal{O}(i)$ is the set of pages that page i links to
- How do we solve for π ?

Google's PageRank (cont.)

- Assume every page has an outgoing link
- In terms of matrices, π satisfies $P\pi = \pi$

- The matrix P has entries

$$[P]_{ij} = \begin{cases} \frac{1}{|\mathcal{O}(j)|} & \text{if } i \in \mathcal{O}(j) \\ 0 & \text{otherwise} \end{cases}$$

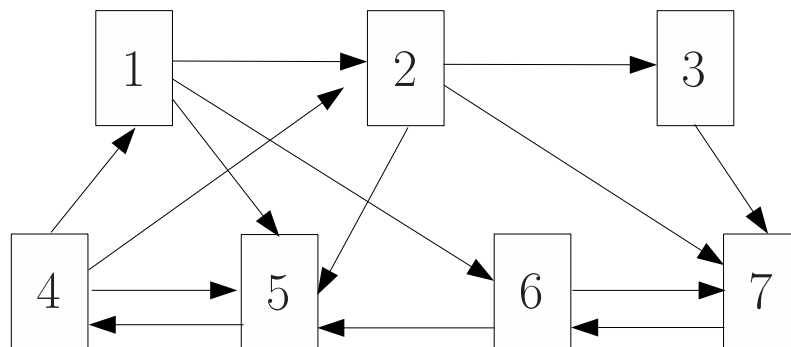
- Finding π seems to require an enormous eigenvector calculation...

Google's PageRank (cont.)

- The matrix P is a stochastic matrix
- Can interpret P as transition matrix of following MC:
 - Each page is a state
 - Move from one page to the next by following a random link
- In practice PageRank works by:
 - Moving among pages by randomly following links...
 - Keeping track of the number of visits to each page...
 - Ranking of page i given by fraction of visits to page i

PageRank Example

- Consider the following simple web graph:



- The following scoring of pages satisfies $P\pi = \pi$:

$$\pi = [0.075 \ 0.1 \ 0.033 \ 0.225 \ 0.225 \ 0.183 \ 0.158]^T$$

- Pages 4 and 5 are the highest ranking pages

Problems with PageRank

- If web graph is not connected, π is not unique
- That is, might have multiple recurrent classes
- A loop of links creates a small recurrent set of pages
- Also, we had to assume all pages had outgoing links...
- How do we address these problems?

Modified PageRank

- Place a self-link on every page
- In each step, jump to an arbitrary web page with small probability
- The resulting P is

$$[P]_{ij} = \begin{cases} \frac{1-\alpha}{|\mathcal{O}(j)|} + \frac{\alpha}{n} & \text{if } i \in \mathcal{O}(j) \\ \frac{\alpha}{n} & \text{otherwise} \end{cases}$$

- The MC with this transition matrix has a single recurrent class

Modified PageRank (cont.)

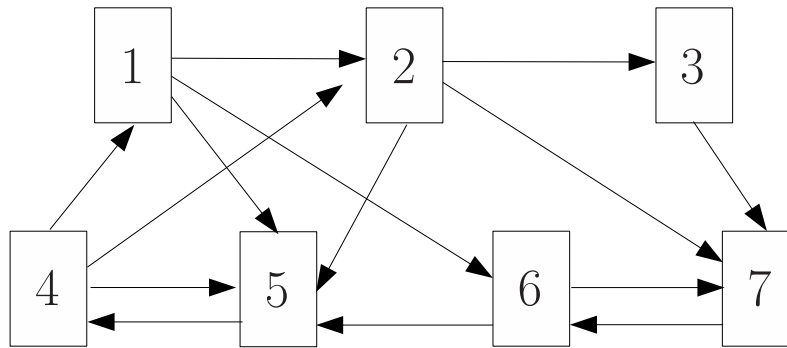
- Now $P\pi = \pi$ has a unique solution, up to a scaling
- This modified definition gives page i rank

$$[\pi]_i = (1 - \alpha) \sum_{j \in \mathcal{I}(i)} \frac{[\pi]_j}{|\mathcal{O}(j)|} + \frac{\alpha}{n}$$

Modified PageRank (cont.)

- Also, $P^T\beta = \beta$ has a unique solution, up to a scaling
- $P^T\beta = \beta$ implies $\beta = \lambda\mathbf{1}$, so weak law of large numbers holds
- We can estimate $[\pi]_i$ by ‘random surfing’
- Randomly click on links, count the frequency of visits to each page

PageRank Example



- Here we use $\alpha = 0.1$ as the 'reset' probability
- The following scoring of pages is obtained:

$$\pi = [0.065 \ 0.085 \ 0.061 \ 0.163 \ 0.248 \ 0.173 \ 0.205]^T$$

- Now 5 is the highest ranking page

The Viterbi algorithm

- Viterbi algorithm used in nearly every cell phone, digital TV, etc.
- Named after Andrew Viterbi, founder of Qualcomm
- **Idea:** A stream of bits is sent from a transmitter to a receiver:
 - For example, say 00011000001111111 is sent
 - Some bits may be corrupted by noise
 - For example, say we receive 00011001001110101 instead
 - We have statistical models of the data source and noise
 - Can we guess the sent bits given the model and received bits?

The Viterbi algorithm (cont.)

- We assume the sent bits are generated by a Markov chain
- Let X_0, X_1, \dots, X_n be RVs giving the values of sent bits
- X_0, X_1, \dots, X_n are generated by a MC with transition probabilities

$$p(x | z) = \mathbf{P}(X_k = x | X_{k-1} = z)$$

- For example, use transition matrix

$$P = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}$$

The Viterbi algorithm (cont.)

- Let Y_0, Y_1, \dots, Y_n be RVs giving the values of received bits
- Each bit is corrupted independently of others, given X_0, X_1, \dots
- Given the value of X_k , received bit Y_k is generated by

$$\begin{aligned} q(y | x) &= \mathbf{P}(Y_k = y | X_k = x) \\ &= \mathbf{P}(Y_k = y | X_0, \dots, X_n, Y_0, \dots, Y_{k-1}) \end{aligned}$$

- For example, could represent by the matrix

$$Q = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

The Viterbi algorithm (cont.)

- Given y_0, \dots, y_n , the goal is to find x_0, \dots, x_n maximizing

$$\mathbf{P}(X_0 = x_0, \dots, X_n = x_n \mid Y_0 = y_0, \dots, Y_n = y_n)$$

- This conditional probability is given by

$$\frac{\rho(x_0)p(x_1 \mid x_0) \cdots p(x_n \mid x_{n-1})q(y_0 \mid x_0) \cdots q(y_n \mid x_n)}{\mathbf{P}(Y_0 = y_0, \dots, Y_n = y_n)}$$

- Since denominator is the same for all x_0, \dots, x_n , can just maximize

$$\rho(x_0)p(x_1 \mid x_0) \cdots p(x_n \mid x_{n-1})q(y_0 \mid x_0) \cdots q(y_n \mid x_n)$$

The Viterbi algorithm (cont.)

- In principle, just search over all x_0, \dots, x_n to maximize

$$\rho(x_0)p(x_1 \mid x_0) \cdots p(x_n \mid x_{n-1})q(y_0 \mid x_0) \cdots q(y_n \mid x_n)$$

- Number of combinations of x_0, \dots, x_n grows exponentially
- When $n = 63$, there are 18,446,744,073,709,551,616 possibilities!
- Fortunately, we can exploit the recursive structure of Markov chains

The Viterbi algorithm (cont.)

- Define V_n by

$$V_n(x_n) = \max_{z_0, \dots, z_{n-1}} \{ \rho(z_0) \cdots p(x_n | z_{n-1}) q(y_0 | z_0) \cdots q(y_n | x_n) \}$$

- The optimal x_n given y_0, \dots, y_n maximizes $V_n(x_n)$
- The V_k can be computed recursively as

$$\begin{aligned} V_n(x_n) &= \max_{z_0, \dots, z_{n-1}} \{ p(z_0) \cdots p(x_n | z_{n-1}) q(y_0 | z_0) \cdots q(y_n | x_n) \} \\ &= \max_{z_{n-1}} \{ q(y_n | x_n) p(x_n | z_{n-1}) V_{n-1}(z_{n-1}) \} \end{aligned}$$

Online Viterbi algorithm

- This algorithm computes optimal x_k given y_0, \dots, y_k :

1. Let $V_0(x_0) = q(y_0 | x_0) \rho(x_0)$
2. The optimal x_0 is the choice that maximizes $V_0(x_0)$
3. Given V_{k-1} and y_k , let

$$V_k(x_k) = q(y_k | x_k) \cdot \max_{z_{k-1}} \{ p(x_k | z_{k-1}) V_{k-1}(z_{k-1}) \}$$

4. The optimal x_k is the choice that maximizes $V_k(x_k)$
- We say this is 'online' because x_k is estimated only from y_0, \dots, y_k
 - What if we want to guess x_k given y_0, \dots, y_n , where $k < n$?

The Viterbi algorithm (cont.)

- Given y_0, \dots, y_n , consider the x_0, \dots, x_n maximizing

$$\mathbf{P}(X_0 = x_0, \dots, X_n = x_n \mid Y_0 = y_0, \dots, Y_n = y_n)$$

- Suppose we already know x_{k+1}^*, \dots, x_n^*

- We want to choose x_0, \dots, x_k to maximize

$$\rho(x_0) \cdots p(x_{k+1}^* \mid x_k) \cdots p(x_n^* \mid x_{n-1}^*) q(y_0 \mid x_0) \cdots q(y_n \mid x_n^*)$$

The Viterbi algorithm (cont.)

- We can factor out the constant term

$$p(x_{k+2}^* \mid x_{k+1}^*) \cdots p(x_n^* \mid x_{n-1}^*) q(y_{k+1} \mid x_{k+1}^*) \cdots q(y_n \mid x_n^*)$$

- So, we want to maximize

$$p(x_{k+1}^* \mid x_k) \left(\rho(x_0) \cdots p(x_k \mid x_{k-1}) q(y_0 \mid x_0) \cdots q(y_k \mid x_k) \right)$$

- The optimal x_k is obtained by maximizing

$$p(x_{k+1}^* \mid x_k) V_k(x_k)$$

The Viterbi algorithm (cont.)

- We can estimate previous bits via a backwards recursion
- Suppose we have already applied the online algorithm
- This algorithm computes optimal x_0, \dots, x_n given y_0, \dots, y_n :
 1. Start with V_0, \dots, V_n and the optimal x_n
 2. Given the optimal x_{k+1} , the optimal x_k maximizes

$$p(x_{k+1}^* | x_k) V_k(x_k)$$

Viterbi algorithm example

- Suppose we have the source and channel matrices

$$P = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} \quad \text{and} \quad Q = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

- The true bit sequence is 0011111010011111
- The observed bit sequence is 1010111000011111
- The online algorithm estimates 0010111000011111
- With all observations, we estimate 0011111000011111

Markov Chain Monte Carlo

- Suppose we have a collection of random variables Y_1, \dots, Y_m
- Suppose each Y_i has $\mathbf{Im}(Y_i) = \{0, 1\}$
- We know full joint PMF

$$\pi(y_1, \dots, y_m) = \mathbf{P}(Y_1 = y_1, \dots, Y_m = y_m)$$

- For some function $r : \{0, 1\}^m \rightarrow \mathbb{R}$, we want to find

$$\mathbf{E}[r(Y_1, \dots, Y_m)] = \sum_{y_1=0}^1 \cdots \sum_{y_m=0}^1 r(y_1, \dots, y_m) \pi(y_1, \dots, y_m)$$

Markov Chain Monte Carlo (cont.)

- Several reasons why this is difficult
- **Example:** Suppose $m = 64$...
- ...need to sum over 18, 446, 744, 073, 709, 551, 616 combinations!
- Difficult to perform sum numerically...
- ...even if we can sum 10^{12} terms/sec, still takes almost a year
- In principle could sample Y_1, \dots, Y_m many times and use LLN...
- ...if RV's are not independent, need to find conditionals to sample

Markov Chain Monte Carlo (cont.)

- Other difficulties arise as well...
- Sometimes we know π up to a normalizing coefficient...
- ...that is, for some known q , we know π is of the form

$$\pi(y_1, \dots, y_m) = Cq(y_1, \dots, y_m)$$

- Constant C is unknown ; need to sum q over all values to find it
- For example, occurs when considering conditionals derived from π

Markov Chain Monte Carlo (cont.)

- It turns out that we can sample Y_1, \dots, Y_m using Markov chains
- **Idea:** We generate a MC with $P\pi = \pi$...
- ...design transition matrix so MC is easy to simulate...
- ...can simulate MC and use LLN to estimate $\mathbf{E}[r(Y_1, \dots, Y_m)]$
- How do we design such a transition matrix?

Markov Chain Monte Carlo (cont.)

- Let each possible value of Y_1, \dots, Y_m be a state of our MC...
- ...that is, $\mathcal{X} = \{(y_1, \dots, y_m) \mid y_i \in \{0, 1\}, i = 1, \dots, m\}$
- We can enumerate all the states so that $\mathcal{X} = \{1, \dots, 2^m\}$
- Suppose (y_1, \dots, y_m) is the k th state
 - Define vector r with $r(y_1, \dots, y_m)$ as its k th element
 - Define vector π with $\pi(y_1, \dots, y_m)$ as its k th element
- In this notation, $r^T \pi = E[r(Y_1, \dots, Y_m)]$

Markov Chain Monte Carlo (cont.)

- Let $N = |\mathcal{X}| = 2^m$
- If we are in state $X_t = x$ at time t , we transition as follows:

$$X_{t+1} = \begin{cases} z \neq x & \text{with probability } p(z | x) = \frac{1}{N} \min \left\{ 1, \frac{q(z)}{q(x)} \right\} \\ x & \text{with probability } 1 - \sum_{z \neq x} p(z | x) \end{cases}$$

- Easy to simulate:
 - Uniformly sample $z \in \mathcal{X}$; sample each component independently
 - Move to z with probability $\min \left\{ 1, \frac{q(z)}{q(x)} \right\}$; otherwise stay at x

Markov Chain Monte Carlo (cont.)

- Note that for all $x, z \in \mathcal{X}$,

$$\frac{q(z)}{q(x)} = \frac{\pi(z)}{\pi(x)}$$

- That is, we don't need normalizing coefficient to simulate
- $x \leftrightarrow z$ for all $x, z \in \mathcal{X}$; transition matrix has $\lambda(P) = 1$ unique
- It turns out that π is the invariant probability for this MC...

Markov Chain Monte Carlo (cont.)

- For all $x, z \in \mathcal{X}$,

$$\begin{aligned} p(z | x)\pi(x) &= \frac{1}{N} \min \left\{ 1, \frac{\pi(z)}{\pi(x)} \right\} \pi(x) \\ &= \frac{1}{N} \min \{ \pi(x), \pi(z) \} \end{aligned}$$

- This gives

$$\begin{aligned} &\sum_{z \neq x} p(x | z)\pi(z) + \left(1 - \sum_{z \neq x} p(z | x)\right)\pi(x) \\ &= \frac{1}{N} \sum_{z \neq x} \min\{\pi(z), \pi(x)\} + \left(\pi(x) - \frac{1}{N} \sum_{z \neq x} \min\{\pi(x), \pi(z)\}\right) \\ &= \pi(x) \end{aligned}$$

Markov Chain Monte Carlo (cont.)

- $\lambda(P) = 1$ is unique and $P\pi = \pi$
- Therefore, Poisson's equation has solution with $\beta = (r^T \pi)\mathbf{1}$
- Weak Law of Large Numbers holds
- We can estimate $\mathbf{E}[r(Y_1, \dots, Y_m)] = r^T \pi$ by simulating Markov chain
- That is, for any initial state and large n ,

$$\frac{1}{n} \sum_{k=0}^{n-1} r(X_k) \approx \mathbf{E}[r(Y_1, \dots, Y_m)]$$

Queueing models

- Consider the following data transmission system:
 - Time is slotted ; For each time slot t ...
 - ...a packet arrives with probability λ
 - ...if we have a packet to send, we attempt to send it...
 - ...an attempted transmission is successful with probability μ
 - ...if packets arrive faster than we can send them, a line forms
- We would like to answer questions such as:
 - On average, how many packets are in the queue?
 - What is the average delay seen by each packet?

Queueing models (cont.)

- Can model as a countable-state Markov chain
- The state gives the number of packets in the queue
- The state-space is $\mathcal{X} = \{0, 1, 2, \dots\}$
- If $X_t = x > 0$, then

$$X_{t+1} = \begin{cases} x + 1 & \text{with probability } \lambda(1 - \mu) \\ x - 1 & \text{with probability } (1 - \lambda)\mu \\ x & \text{with probability } 1 - \lambda - \mu + 2\lambda\mu \end{cases}$$

- If $X_t = 0$, then

$$X_{t+1} = \begin{cases} x + 1 & \text{with probability } \lambda \\ x & \text{with probability } 1 - \lambda \end{cases}$$

Queueing models (cont.)

- To measure steady-state average backlog, use $r(x) = x\dots$
- Steady-state average number of packets in the queue is

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=0}^{n-1} \mathbf{E}[r(X_k) \mid X_0 = x]$$

- We can compute this quantity via Poisson's equation

Queueing models (cont.)

- Let

$$\beta(x) = \frac{(1 - \lambda)\lambda}{\mu - \lambda} \quad \text{for all } x \in \mathcal{X}$$

$$h(x) = \frac{1}{2(\mu - \lambda)}(x^2 + (1 - 2\lambda)x)$$

- We first want to show that for all $x \in \mathcal{X}$,

$$r(x) + \mathbf{E}[h(X_{t+1}) \mid X_t = x] - h(x) = \beta(x)$$

Queueing models (cont.)

- Let A_t be independent random variables with
 - $A_t = 1$ with probability λ
 - $A_t = 0$ with probability $1 - \lambda$
- Similarly, D_t are independent random variables with
 - $D_t = 1$ with probability μ
 - $D_t = 0$ with probability $1 - \mu$
- Let $I : \mathcal{X} \rightarrow \{0, 1\}$ be defined as

$$I(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

Queueing models (cont.)

- For all t ,

$$X_{t+1} = X_t + A_t - I(X_t)D_t$$

- Now we have,

$$\mathbf{E}[X_{t+1} | X_t = x] = x + \lambda - \mu I(x)$$

- Also

$$\begin{aligned} \mathbf{E}[X_{t+1}^2 | X_t = x] &= \mathbf{E}[(X_t + A_t - I(X_t)D_t)^2 | X_t = x] \\ &= x^2 + 2(\lambda - \mu I(x))x + \lambda + \mu I(x) - 2\lambda\mu I(x) \\ &= x^2 + 2(\lambda - \mu I(x))x + \lambda + (1 - 2\lambda)\mu I(x) \end{aligned}$$

Queueing models (cont.)

- Putting everything together gives

$$r(x) + \mathbf{E}[h(X_{t+1}) | X_t = x] - h(x) = \frac{(1 - \lambda)\lambda}{\mu - \lambda}$$

- We also need to show that $\lim_{n \rightarrow \infty} \frac{1}{n} \mathbf{E}[h(X_n) | X_0 = x] = 0$
- This is achieved by showing

$$\mathbf{E}[h(X_{t+1})^2 | X_t = x] - h(x)^2 < \infty \quad (\text{in your HW})$$

- Therefore, the steady-state average backlog is

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=0}^{n-1} \mathbf{E}[X_k | X_0 = x] = \frac{(1 - \lambda)\lambda}{\mu - \lambda}$$

