Lecture 12 Random Walks and Markov Chain Monte Carlo (MCMC) Methods	L12.1 6.046 03/23/2017

Today:

Random walks in graphs → Stationary distribution

Examples

- Diffusion
- Card shuffling
- PageRank

Markov chain Monte Carlo algorithms

Random walks in graphs

- Methods for exploring a graph from a particular starting vertex s include breadth-first search (BFS) and depth-first-search (DFS).
- Today, in contrast to these orderly searches, will examine the properties of random searches, which turn out to be powerful for a variety of applications.

Definition For the undirected graph $G=(V,E)$ and the starting vertex s, define a random walk in G of length t as a randomized process in which we start at s and repeat t times the process of choosing randomly among the neighbors v' of the current vertex v we are at, and moving to it.

- If our graph has (non-negative) edge weights, we move to a particular neighbor v' with a probability proportional to the edge weight w_e for $e=(v, v')$.

- For directed graphs we apply a similar definition but traverse arcs (directed edges) outgoing from v (whose tail is at v and whose head is at v').

Representations of random walks

- Trajectory. One way of representing a particular random walk is as a trajectory — essentially a list of the vertices visited in the order in which they were visited.

- Distribution. Generally we are interested in a statistical summary of one (or more) walks that is represented as a distribution. More specifically, today we focus on the probability distribution on vertices induced by walks.

(★ This is the one we care most about)

Let $p_v^t \equiv$ probability that walk visits vertex v at step t of walk.

$$p_v^0 = \begin{cases} 1 & \text{if } v=s \\ 0 & \text{otherwise} \end{cases}$$ because walk starts at s, and $t=0$ is the start

$$p_v^{t+1} = \sum_{e \in E, e=(u,v)} \frac{1}{d(u)} p_u^t \quad \forall v \in V$$ where $d(u)$ = degree of vertex u

Note:

- Generally will represent the set of p_v^t across v as a vector $p^t \in \mathbb{R}^V$ whose v^{th} coordinate is p_v^t.
- For weighted graphs, $d(u) \equiv$ weighted degree of vertex u (← outgoing)

so, $d(u) = \sum_{e \in E, e=(u,v)} w_e$ & $p_v^{t+1} = \sum_{e \in E, e=(u,v)} \frac{w_e}{d(u)} p_u^t \quad \forall v \in V$

and we assume $d(u) > 0 \ \forall u$ (Self loops are allowed)

Lazy random walks

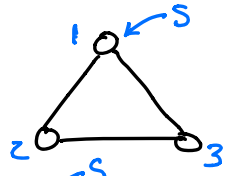
Sometimes it is useful to allow the random walker to remain at the current vertex in addition to the possibility of progressing to a neighboring vertex. A walk of such a type is called a lazy random walk, and we will represent its probabilities with hats, to distinguish from an ordinary random walk. (Essentially adds self loop to each vertex.)

$$\hat{p}_v^{t+1} = p_{lazy}\, \hat{p}_v^t + (1 - p_{lazy}) \sum_{e \in E, e=(u,v)} \frac{1}{d(u)} \hat{p}_u^t \quad \forall v \in V$$

$0 < p_{lazy} < 1$

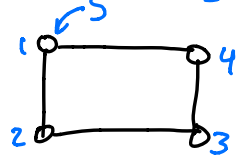
(p_{lazy} ← probability of remaining at current vertex)

Examples



t: 0 → 1 → 2 → 3 → ... converges → ∞

$\begin{pmatrix}1\\0\\0\end{pmatrix} \to \begin{pmatrix}0\\1/2\\1/2\end{pmatrix} \to \begin{pmatrix}1/2\\1/4\\1/4\end{pmatrix} \to \begin{pmatrix}1/4\\3/8\\3/8\end{pmatrix} \to \cdots \to \begin{pmatrix}1/3\\1/3\\1/3\end{pmatrix}$



$\begin{pmatrix}1\\0\\0\\0\end{pmatrix} \to \begin{pmatrix}0\\1/2\\0\\1/2\end{pmatrix} \to \begin{pmatrix}1/2\\0\\1/2\\0\end{pmatrix} \to \begin{pmatrix}0\\1/2\\0\\1/2\end{pmatrix}$

oscillates forever // Lazy walk eliminates oscillation & leads to convergence

Matrix formalism

Given undirected graph $G=(V,E)$ with $n=|V|$, let the adjacency matrix A of G be the $n \times n$ matrix

$$A_{u,v} = \begin{cases} 1 & \text{if } (v,u) \in E \\ 0 & \text{otherwise} \end{cases}$$

and the degree matrix D of G be the $n \times n$ diagonal matrix with $D_{u,u}$ = degree of vertex u

$$D_{u,v} = \begin{cases} d(u) & \text{if } u=v \\ 0 & \text{otherwise} \end{cases}$$

Note: D^{-1} is then

$$D^{-1}_{u,v} = \begin{cases} 1/d(u) & \text{if } u=v \\ 0 & \text{otherwise} \end{cases}$$

Define the walk matrix

$$W \equiv A D^{-1} \quad \rightarrow \quad W_{u,v} = \begin{cases} \frac{1}{d(v)} & \text{if } (v,u) \in E \\ 0 & \text{otherwise} \end{cases}$$

Then $p^{t+1} = W p^t = W^t p^0$ for ordinary random walks

and $\hat{W} \equiv P_{lazy} I + (1 - P_{lazy}) W$

$\vec{p}^{\,t+1} = \hat{W} \hat{p}^t = \hat{W}^t \hat{p}^0$

} for lazy random walks

For weighted graphs, adjacency and degree matrices become

$$A_{u,v} = \begin{cases} w_e \text{ if } e=(v,u) \in E \\ 0 \text{ otherwise} \end{cases} \qquad D_{u,v} = \begin{cases} \sum_{e=(u,v') \in E} w_e \text{ if } u=v \\ 0 \text{ otherwise} \end{cases}$$

Stationary Distribution

L12.5

It turns out that many graphs converge to a stationary distribution independent of the starting vertex. That is, for large t the distribution p^t (or $\hat{p}^t$) approaches a distribution that does not change as additional steps are carried out. This is actually a surprising and amazing result!

We signify this stationary distribution as π and note its invariance with additional steps of the walk with

$$W\pi = \pi \quad \text{and} \quad \hat{W}\pi = \pi$$

for ordinary random walks — for lazy random walks

The stationary state π represents a "steady state".

For undirected graphs, the stationary distribution π is given by $\pi_v = \dfrac{d(v)}{\sum_{u \in V} d(u)}$

That is, the probability of being at v is proportional to its degree.

Proof

$$\pi_v = \frac{d(v)}{\sum_{u \in V} d(u)} \Rightarrow \pi = \begin{pmatrix} d(1) \\ d(2) \\ \vdots \\ d(n) \end{pmatrix} \frac{1}{\sum_{u \in V} d(u)} = D\mathbf{1} \frac{1}{\sum_{u \in V} d(u)}$$

$\mathbf{1}$ = vector of all 1's

Stationary: Need to show $W\pi = \pi$

$$W\pi = \pi$$

$$AD^{-1}D\mathbf{1}\frac{1}{\sum_{u \in V} d(u)} = D\mathbf{1}\frac{1}{\sum_{u \in V} d(u)}$$

$AD^{-1} = I$

Examining def. of adjacency matrix this is also $\begin{pmatrix} d(1) \\ d(n) \end{pmatrix}$ for undirected graphs

$$A\mathbf{1} = D\mathbf{1} = \begin{pmatrix} d(1) & & & \\ & d(2) & & \\ & & \ddots & \\ & & & d(n) \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} d(1) \\ d(2) \\ \vdots \\ d(n) \end{pmatrix} \checkmark$$

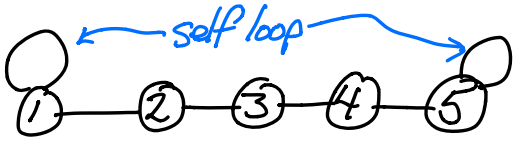
Theorem: Every connected non-bipartite undirected graph has a stationary distribution to which random walks in the graph are guaranteed to converge for $t \to \infty$.

- For lazy random walks, the theorem holds for all connected undirected graphs (they don't need to be non-bipartite).
- For directed graphs the conditions for the theorem to hold are that:
  - (a) The graph is strongly connected ("irreducible")
    Every vertex is reachable from all other vertices
  - (b) The graph is aperiodic
    It is not the case that every cycle has a length that is a multiple of some number larger than one.

Diffusion Example

model of 1D diffusion

Consider the graph →



$$W = \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

diagonal

$$W^2 = \frac{1}{4}\begin{pmatrix} 2 & 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 & 1 \\ 0 & 1 & 0 & 2 & 1 \\ 0 & 0 & 1 & 1 & 2 \end{pmatrix}$$

$$W^3 = \frac{1}{8}\begin{pmatrix} 3 & 3 & 1 & 1 & 0 \\ 3 & 1 & 3 & 0 & 1 \\ 1 & 3 & 0 & 3 & 1 \\ 1 & 0 & 3 & 1 & 3 \\ 0 & 1 & 1 & 3 & 3 \end{pmatrix} \quad \cdots \quad W^\infty = \frac{1}{5}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \Rightarrow \pi = \begin{pmatrix} 1/5 \\ 1/5 \\ 1/5 \\ 1/5 \\ 1/5 \end{pmatrix}$$

Card Shuffling

A deck of 52 playing cards can be in one of 52! (52 factorial) permutations. Conceptualizing each permutation as a vertex (which would be a very large space, indeed), we can examine card shuffling as starting from one vertex in this graph and performing a random walk in the graph.

The point of (fair) shuffling is to arrive at a random vertex in the permutation graph that is independent of the starting vertex. The edge set for the graph is determined by the method of shuffling.

Consider 3 different shuffling methods

① "riffle shuffle" – usual method

② "top-to-random" – top card inserted randomly

③ "cut-the-deck" – not really a method

The top card can end up in one of 52 positions, before or after any other card in the deck. Each vertex is connected to 51 other vertices by an edge as well as to itself via a self-loop. The graph is strongly connected and meets other conditions to have a stationary distribution.

Similar to a "parallel" version of "top-to-random". Each vertex is connected to many more than 51 neighbors. The graph is strongly connected and meets the other conditions for having a stationary distribution.

Cutting the deck leads to a cyclic permutation of the deck, but no real mixing of the cards. The graph is not connected and fails the conditions for having a stationary distribution independent of starting vertex. But, all 52 accessible cyclic permutations are reachable in a single step.

So after a sufficient number of shuffles, both riffle shuffle and top-to-random will lead to a distribution sufficiently close to the stationary distribution (which has a probability of $1/52!$ for each permutation). Intuitively we know riffle shuffling is more efficient and should take fewer steps to approach the limiting distribution.

It is beyond the scope of 6.046 to go into the details, but this number of steps, called the "mixing time", can be studied. For n cards

$$\tau_{mix} \sim \frac{3}{2}\log_2 n \text{ for riffle shuffle} \xrightarrow{n=52} \tau_{mix} \sim 8$$

$$\tau_{mix} \sim n \ln n \text{ for top-to-random} \xrightarrow{n=52} \tau_{mix} \sim 205$$

Ranking web pages

Consider the web as a directed graph $G=(V,E)$

- nodes are web pages
- directed edges (u,v) represent link from page u to page v
- let's ignore content and use pattern of links to rank pages in terms of importance or usefulness

$Rank \in \mathbb{R}^{|V|}$ → vector giving rank of each page

① Count Rank

Rank of page is proportional to number of incoming edges.

$$CountRank_u = \sum_{v \in V} A_{u,v}$$

$$A_{u,v} = \begin{cases} 1 & \text{if } (v,u) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$\text{CountRank} = A\,1$$

↑ vector of all 1s.

CountRank not ideal

Intuitively we know that being on the list of the three best restaurants is better than being on a list of three hundred restaurants, but CountRank considers them equally good.

② WeightRank

Weight recommendation coming from a given page by the inverse of the number of recommendations that page makes.

$$\text{WeightRank}_u = \sum_{v \in V} \frac{1}{d(v)} A_{u,v}$$

$d(v)$ ← outgoing degree of v

$$\text{WeightRank} = AD^{-1}1 = W1$$

W ← random walk matrix

D^{-1} ← outgoing degree matrix

$$D^{-1}_{u,v} = \begin{cases} \frac{1}{d(v)} & \text{if } u = v \\ 0 & \text{otherwise} \end{cases}$$

can add self loop for vertex with no outgoing edges

WeightRank is proportional to probability of ending up at a given page if first pick any page uniformly at random and then take one step of a random walk from there.

WeightRank is not ideal.

A recommendation from a page depends (inversely) on the number of other recommendations it makes, but not on the importance of the recommending page.

③ RecRank

$$\text{RecRank}_u = \sum_{v \in V} A_{u,v} \left[\frac{1}{d(v)}\right] \text{RecRank}_v$$

↑ Here a recommendation is weighted by the strength (rank) of the recommending page.

$$\text{RecRank} = W\,\text{RecRank}$$

↑ This implies that if RecRank is interpreted as a probability distribution, it is a stationary distribution for W

RecRank is not ideal, as the stationary distribution need not be unique.

④ PageRank

$$\text{PageRank} = (1-\alpha)\,W\,\text{PageRank} + \frac{\alpha}{n}\mathbf{1}$$

↑ $0 < \alpha < 1$ is a parameter of choice

↑ $|V| = n$

PageRank corresponds to a stationary distribution for a random process that with probability $(1-\alpha)$ takes a random "surf" step, & with probability α jumps (independently of edges) to a random vertex in the graph. α controls the frequency of random jumps.

Markov Chain Monte Carlo Methods

A Markov chain is a process for which the future state of the system depends (probabilistically) on the current state of the system without any dependence on past states.

The random walks in graphs that we have been discussing are Markov chains with the vertices representing states and the edges (and their weights) relative transition probabilities. Future states of the system can be obtained by operating on the current state (which may be represented as a probability distribution) with the walk matrix W (or $\hat{W}$).

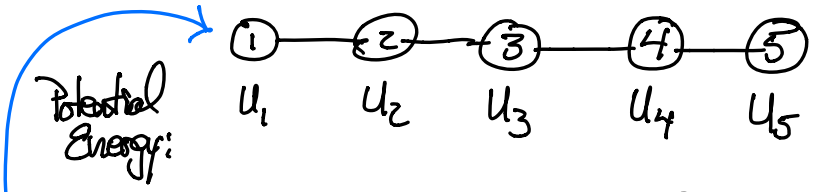
Metropolis-Hastings algorithm

To study certain problems, many of which come from statistical physics, we know the states (vertices) and we know something about the stationary distribution (we know the relative probability of any two states, for instance), but calculating the transition probabilities directly is problematic.

→ The Metropolis-Hastings algorithm provides a way to construct the transition probabilities "on the fly" so that random walks on the graph produce the proper stationary distribution.

→ Note that every random walk on a graph corresponds to a Markov chain, and every Markov chain can be viewed as a random walk in some graph with weights corresponding to transition probabilities.

Example: Diffusion under a potential energy function



Probability of state i: $P_i \propto e^{-\beta U_i}$ ← from statistical physics

$\beta = \frac{1}{k_B T}$

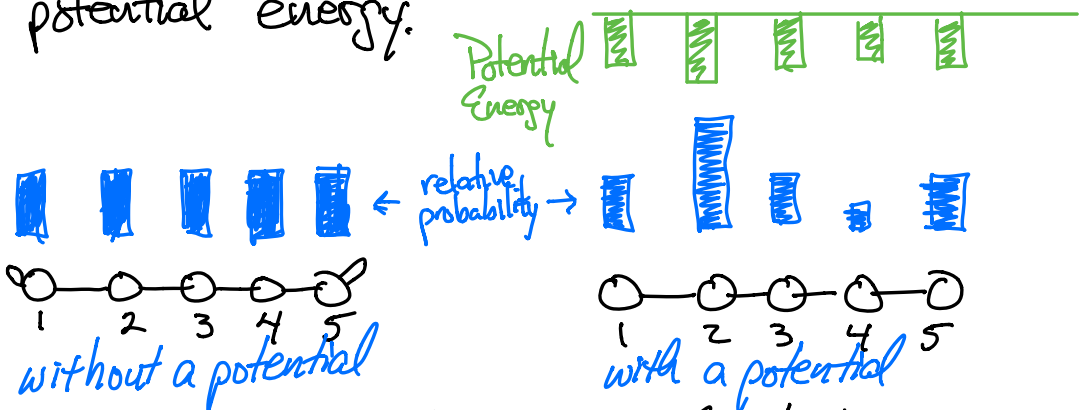
Metropolis-Hastings procedure for random walk:

1. Start at arbitrary vertex $X_t = X_0$
2. Randomly pick a neighbor of X_t as a candidate to transition to. Call it X_{trial}
3. Evaluate $f_{trial} = \frac{e^{-\beta U(X_{trial})}}{e^{-\beta U(X_t)}} = \frac{P(X_{trial})}{P(X_t)}$

Don't need self loops because M-H has them implicitly.

4. If $t_{trial} \geq 1$ (meaning trial vertex is at least as probable in stationary distribution as X_t) the "accept the trial move" and let $X_{t+1} = X_{trial}$, advance t, and repeat from step 2.
5. Otherwise, accept the trial move with probability t_{trial} (by picking a random number). If the trial move is not accepted, reject it by setting $X_{t+1} = X_t$ (effectively walking along a self loop), advance t, and repeat from step 2.

When we modeled diffusion without a potential, the stationary distribution was uniform. Here the particles should cluster at points of lower potential energy.



This simple 1-D example is trivial. The true power of the method is for highly dimensional spaces (eg., 100,000-D) where it is not possible to visualize & traverse the graph.