Probability vectors

- A probability (row) vector $\mathbf{x} = (x_1, \ldots x_n)$ tells us where the walk is at any point.
- E.g., (000…1…000) means we're in state i.
  $1 \quad i \quad n$

More generally, the vector $\mathbf{x} = (x_1, \ldots x_n)$ means the walk is in state i with probability x_i.

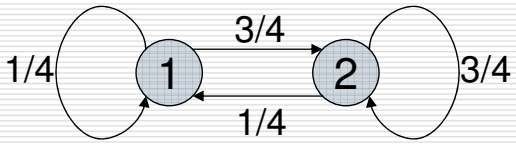
$$\sum_i x_i = 1$$

Change in probability vector

- If the probability vector is $\mathbf{x} = (x_1, \ldots x_n)$ at this step, what is it at the next step?
- Recall that row i of the transition prob. Matrix **P** tells us where we go next from state i.
- So from **x**, our next state is distributed as **xP**.

Steady state example

- The steady state looks like a vector of probabilities $\mathbf{a} = (a_1, \ldots a_n)$:
  - a_i is the probability that we are in state i.



For this example, $a_1=1/4$ and $a_2=3/4$.

How do we compute this vector?

- Let $\mathbf{a} = (a_1, \ldots a_n)$ denote the row vector of steady-state probabilities.
- If our current position is described by $\mathbf{a}$, then the next step is distributed as $\mathbf{aP}$.
- Whenever $\mathbf{a}$ is the steady state, it should be $\mathbf{a}=\mathbf{aP}$.
- Solving this matrix equation gives us $\mathbf{a}$.
  - So $\mathbf{a}$ is the (left) eigenvector for $\mathbf{P}$.
  - (Corresponds to the "principal" eigenvector of $\mathbf{P}$ with the largest eigenvalue.)
  - Transition probability matrices always have larges eigenvalue 1.

One way of computing **a**

- Recall, regardless of where we start, we eventually reach the steady state **a**.
- Start with any distribution (say $\mathbf{x}=(10\ldots0)$).
- After one step, we're at $\mathbf{xP}$;
- After two steps at $\mathbf{xP}^2$, then $\mathbf{xP}^3$ and so on.
- "Eventually" means for "large" k, $\mathbf{xP}^k = \mathbf{a}$.
- Algorithm: multiply **x** by increasing powers of **P** until the product looks stable.
- Strict convergence is not necessary;
  - [Brin&Page98] reports acceptable convergence on 322M nodes in about 50 iterations

Pagerank summary

- Preprocessing:
  - Given graph of links, build matrix **P**.
  - From it compute **a**.
  - The entry a_i is a number between 0 and 1: the pagerank of page i.
- Query processing:
  - Retrieve pages meeting query.
  - Rank them by their pagerank.
  - Order is query-*independent*.

Topic Specific Pagerank [Have02]

- Conceptually, we use a random surfer who teleports, with say 10% probability, using the following rule:
  - Selects a category (say, one of the 16 top level ODP categories) based on a query & user -specific distribution over the categories
  - Teleport to a page uniformly at random within the chosen category
- Sounds hard to implement: can't compute PageRank at query time!

Topic Specific Pagerank [Have02]

- Implementation
  - **offline**:Compute pagerank distributions wrt to *individual* categories

    Query independent model as before

    Each page has multiple pagerank scores – one for each ODP category, with teleportation only to that category
  - **online**: Distribution of weights over categories computed by query context classification

    Generate a dynamic pagerank score for each page - weighted sum of category-specific pageranks

Considerations on PageRank

- The ranking returned by PageRank can be used for doing prioritized crawling
- Without the teleporting factor, PageRank would be un-crackable by spammers

- The (undisclosed) ranking formula used by Google nowadays is a complex recipe (PageRank is the most important ingredient). Other ingredients include:
  - Text in the page
  - Anchor text
  - Query term proximity
  - URL length

HITS (Klimberg98]

- HITS may be seen as a modification of InDegree where a companion notion of the authority value (the hub value) is introduced.

- Authority Value a_i of p_i (how authoritative p_i is, 'seminal papers')
- Hub Value h_i of p_i (how good p_i is helping the user in locating authoritative pages, 'survey papers')

- They are defined in a mutual recursive manner
  - A page is a good hub when it points to many good authoritative pages $h_i = \sum_{j \in F(i)} a_j$
  - A page is a good authority when it is pointed by many good hubs $a_i = \sum_{j \in B(i)} h_j$

Equations

- Recasting equations in a matrix-vector form, we have
  - $h \leftarrow W\, a$
  - $a \leftarrow W^T h$
- Substituting these into one another, we obtain
  - $h = W\, W^T h$
  - $a = W^T W\, a$
- Eigenvectors equations!

Considerations

- The iterative updates, if scaled by an appropriate eigenvalues, are equivalent to the power iteration method for computing the eigenvectors of WW^T and W^TW respectively
- Thus the steady state is determined by the entries in W and hence the structure of the graph
- In computing these eigenvectors entries, we are not restricted to use the power iteration method

Problems

- The problem of HITS is that it is easily spammable: in fact, a spammer wishing to promote a page p_s only needs to set up a page p_t that points to many known authorities and to p_s

A variant: HubAvg

- A problem with HITS is that h_i monotonically grows not only with the authority, but also with the number $|F(i)|$ of the forward neighbors of p_i;
- Thus, the best hub is the one which points to all pages in BS!
- The HubAvg algorithm [Borodin+05] views h_i as the average authority value of the forward neighbors of p_i
  - $h_i = (\sum_{j \in F(i)} a_j)/|F(i)|$
  - $a_i = (\sum_{j \in B(i)} h_j)$

A variant: HubAvg

- It can be seen as a hybrid between HITS and PageRank
  - Authority and hubs to every page
  - Subdivides the hub score of a page amongst its forward neighbors
- Fairly easy to spam, although slightly more difficult than HITS