SportsRank

A Google PageRank Implementation on Sports Teams

Beth Nakamura

Math 462: Mathematical Modeling
Dr. Adam Stinchcombe
April 21, 2015

Code and datasets: https://github.com/bnak/SportsRank

ABSTRACT

Google's PageRank algorithm allows efficient ranking of webpages by creating a directed graph of the web. By considering each webpage a node and hyperlinks as the arcs, the resulting Markov matrix provides the left steady-state eigenvector; this PageRank vector provides the relative probabilities for each webpage, and thus a ranking. PageRank provides a ranking of a complex network using a single measure: the number and direction of hyperlinks. SportsRank applies this model to athletics, demonstrated with the NBA. Assume each team is a node with arcs representing the sum margin of losses from one team to another. SportsRank ranks the team based on point differential in losses, which can be compared to the standard win-loss method of ranking. Although both ranking methods have significant discrepancies' when compared to actual playoff results, the potential of PageRank is established – if a single representative measure is identified to connect the nodes, PageRank is an efficient, elegant ranking system for large directed graphs.

Table of Contents

Introduction

Just "Google it": the everyday action we all do to effectively use the web. Google's search engine crawls billions of webpages to deliver relevant and useful results to almost any search term in fractions of a second. Its proprietary algorithm, PageRank, was Google's earliest attempt to bring order to the web.

Google's PageRank is an innovative application of Markov chains that provides a framework for ranking other complicated networks. PageRank considers webpages as nodes in a directed graph and arcs as hyperlinks, with the number of outedges can be entered into a Markov matrix. Using Markov chain analysis, PageRank then produces a ranking of all nodes within the graph – the highest ranked nodes, or webpages, appear at the top of the search engine results.

SportsRank applies this model to sports leagues, using teams as nodes and margins of losses as arcs. This paper uses the National Basketball Association (NBA) and Python to implement a PageRank algorithm using past regular season data and the compares it to the performance of the standard rankings based on number of wins.

Google's PageRank

Developed in 1998 by Google founders Sergey Brin and Larry Page, PageRank is an algorithm used to rank search engine results. Faced with the challenge of significant diversity and number of web pages, Brin and Page modeled the Internet as a directed graph and used Markov Chains to measure the relative importance of webpages.

PageRank models the searchable web as a directed graph, with each webpage acting as a node. Every page has some number of forward links (outedges) leading to another webpage and backlinks (inedges), or links directed to that webpage. These links are "citations" with respect to a webpage. Consider an academic paper – inedges would occur when another other research cite the academic paper, while outedges are other papers cited within the paper of interest. A seminal piece of research in a field would be cited often by other works, meaning there would be a high number of inedges to that paper. Additionally, a well written research paper would cite a number of other sources, resulting in a high number of outedges. Google uses directed hyperlinks as a measure to determine which webpages are of high quality, and thus should be ranked higher. PageRank considers the Internet as a directed graph in which webpages are nodes and the arcs are the hyperlinks leading to the next node.

To convert the directed graph to a matrix, we create an $n \times n$ matrix where n is the total number of nodes and k_i is the number of outgoing links for node i. The entries for matrix P are then as follows:

$$P_{ij} = \begin{cases} \frac{1}{k_i} & \text{if } i \text{ has } k > 0 \text{ outgoing links to } j \\ 0 & \text{otherwise} \end{cases}$$

Consider an example Markov matrix below, with websites A, B and C and corresponding indices (row/column number) 0, 1, and 2 respectively. Suppose that A links to 25 other pages, or has 25 outgoing links, one which goes to website B. Thus the entry $P_{0,1} = 1/25$, representing an outedge from A to B and normalized over the total number of outlinks.

Markov Matrix P

	A	B	C	…
A	0	1/25	0	…
B	1/14	0	1/14	…
C	…	…	0	…
…	…	…	…	…

Note that the Markov matrix only carries entries of outgoing lines – this means that incoming links are taken into account as outgoing links of other nodes. However, if a node has no out-edges, it will not have any non-zero row entries, preventing the calculation of an eigenvector. For Google, these are websites that are linked to but have no outgoing links, such as a press release or company product launch; many news outlets or blogs may be linking to this page, but an announcement may not link to any other pages. To account for dangling nodes, PageRank uses a parameter $c \in (0,1)$ to weight most nodes but account for dangling nodes by using the term $(1/n)$(E), with E being a matrix of the same size filled with ones. Matrix P~ accounts for dangling nodes:

$$P^{\sim} = cP + (1 - c)\left(\frac{1}{n}\right)E$$

This fills the entries that would be 0 with extremely small values, ensuring that there are no trivial rows and accounting for dangling nodes when calculating the eigenvector.

With the Markov matrix, the steady-state left eigenvector π is calculated such that $\pi = \pi$P and $\pi = \pi 1$. Recall that entry $\pi_{i,j}$ represents the probability of transitioning to the node corresponding column j. PageRank considers π the PageRank vector, and utilizes the magnitudes of entries to determine a ranking. Recall the three website example above and suppose the resulting PageRank was produced:

$$\pi = \pi P^{\sim} \qquad \pi = \pi 1$$

State/Website:	A	B	C
π =	(.285,	.263,	.452)

In this case, C would be the highest ranked website and show up first in search results. Note that when using Google, search terms help seed the model and provide improved results by potentially identifying which node one is starting at or moving from.

Google's PageRank algorithm has since evolved with the Internet, users, and Google as a company; search engine results now incorporate user history, advertising, and content relevance. Additionally, Google works with significant scale and implements Monte Carlo sampling to produce relevant results rather than processing a full matrix representative of billions of webpages. PageRank is an innovative application of using the steady-state eigenvector of a Markov matrix to rank nodes – it considers the values as probabilities of transitioning to the next state as a ranking mechanism and can thus be used to rank a high volume of nodes in complex and extensive networks.

Applying PageRank to Sports

PageRank uses Markov Chains to rank nodes in a directed graph; this application of Markov Chains and linear algebra allows complicated systems to be analyzed if one identifies a single well-defined measure to compare nodes.

Building off of Albright, Govan, and Meyer's "Generalizing Google's PageRank to Rank National Football League Teams," we consider the measure of summed point differential in losses. Using this measure and PageRank, we will produce a ranking and compare it to standard regular season rankings (based solely on win-loss records, or simply the number of wins in a season) and attempt to determine which measure is a better determinate of playoff success.

Essentially we are comparing the quality of two measures used for ranking, the number of overall wins versus the overall point deficits of a team relative to opposition. By creating a directed graph and considering the point deficits of one team relative to all other teams, a better ranking could result by taking "magnitude" of wins and losses into effect based on margin of loss – that is, a team with few but significant losses would be ranked relatively lower than standard rankings because of drastic point deficits. Theoretically, using a different measure from standard number of wins, one may be able to develop more accurate ranking by using a measure that incorporates nuances such as varying difficulties of schedules, magnitudes of victories, and repeated match-ups.

Model Applied to the National Basketball Association

Each team is a node, with the arc from node i to node j carrying the value of the amount team i lost to team j. The point differentials are summed – that is, if team i lost to team j more than once in the season, the value of the arc is the sum of the margins of loss overall three games.

The NBA has 30 teams that each play 82 regular season games. There has never been an undefeated team (the best single season record is the Chicago Bulls' 1995-1996 season with 72 wins and 10 losses), so the term to account for dangling nodes is not needed. The 30 teams are divided into an East and West conference, although in the regular season every team plays every other team, regardless of conference.

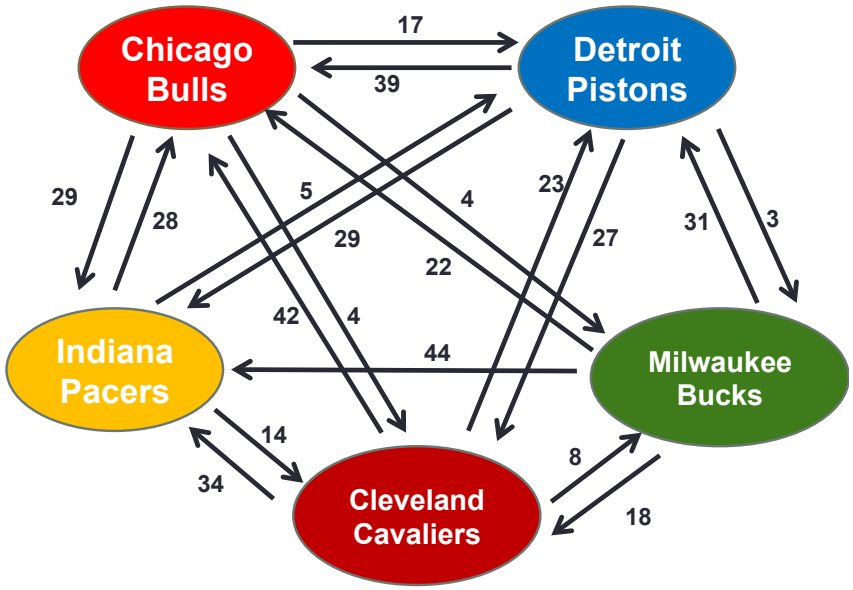
After the regular season, the top 8 teams in each conference are seeded into a best-of-seven elimination tournament. Each match-up plays a best-of-seven series, with the idea that seven games reduces variability produces the best teams (compared to a format such as single-game elimination NCAA tournament). The 2012 playoff bracket is shown below:

With the tournament playoff format, it can be hard to determine a final ranking – we know the top-two overall teams, but we cannot necessarily distinguish between the semifinalists (third and fourth place) because they do not play for third place. These can be ranked by margin of losses; i.e. a team that did not make the finals because they lost the seven game series 3-4 would be ranked higher than the semifinalist who lost to the finalist 0-4. We can compare relative correctness and see if PageRank predicts any upsets of lower seeds beating higher seeds.

Example: NBA Central Division, 2013-2014 Season

As an example, we'll apply this model to a smaller subset of teams, specifically the Central Division consisting of the following five teams: Chicago Bulls, Cleveland Cavaliers, Detroit Pistons, Indiana Pacers, and Milwaukee Bucks. In the 2013-2014 season, these five teams played forty inter-divisional games; the point differentials

of these games create a closed directed graph shown below. Recall that the value of the arcs is the sum total the team lost to next team overall games played – i.e., over all the games between the Milwaukee Bucks and the Indiana Pacers, Milwaukee's point differential in losses totaled 44 points, but Indiana never lost to Milwaukee.



This directed graph is then converted to a point differential matrix; entries are divided by the sum of each row to create a stochastic Markov matrix.

Point Differential Matrix: $P_{i,j}$ is the sum amount team *i* lost to team *j*

	Chicago Bulls	Cleveland Cavaliers	Detroit Pistons	Indiana Pacers	Milwaukee Bucks
Chicago Bulls	0	4	17	29	4
Cleveland Cavaliers	42	0	23	34	8
Detroit Pistons	39	27	0	23	3
Indiana Pacers	28	14	5	0	0
Milwaukee Bucks	22	18	31	44	0

→

Stochastic Markov Matrix: each entry is divided by the sum of the row

	Chicago Bulls	Cleveland Cavaliers	Detroit Pistons	Indiana Pacers	Milwaukee Bucks	Chicago Bulls
Chicago Bulls	0	.074	.314	.537	.074	0
Cleveland Cavaliers	.393	0	.215	.318	.075	.393
Detroit Pistons	.424	.293	0	.250	.033	.424
Indiana Pacers	.595	.298	. 106	0	0	.595
Milwaukee Bucks	.191	.157	.269	.383	0	.191

The PageRank vector, or left eigenvector $\pi = \pi P$, is then calculated, providing the results for the Central Division subset.

	Chicago Bulls	Cleveland Cavaliers	Detroit Pistons	Indiana Pacers	Milwaukee Bucks
$\pi =$	(.645	.338	.360	.577	.085)

Actual Season Rankings (win-loss)	Model Result Rankings	Notes:
Indiana Pacers (56-26)	Chicago Bulls	Chicago and Indiana split series 2-2
Chicago Bulls (48-34)	Indiana Pacers	
Cleveland Cavaliers (33-49)	Detroit Pistons	
Detroit Pistons (29-53)	Cleveland Cavaliers	Cavaliers went 7-9 in division
Milwaukee Bucks (15-67)	Milwaukee Bucks	

There are discrepancies between the two rankings, which is expecting considering the different data sets (Actual season rankings uses all games played, whereas this example only uses the forty inter-divisional games). The limited dataset provides insight on the differences in the rankings. The Chicago Bulls are ranked higher than

the Indiana Pacers in the model because over the four games they played each other, they split the series 2-2 with point differentials of 29 and 28 points, respectively, whereas Indiana won more games overall during the regular season. Similarly, the Cavaliers performed relatively poorly within the division, losing over half the games, resulting in a lower rank when considering only inter-divisional games. Thus, we look forward results in the two ranking systems use the same dataset below.

Python Code and Data

This model was implemented in Python utilizing the built-in linear algebra functionality of SciPy and NumPy libraries. The full Python script is attached in appendix and code and datasets are published on GitHub.

Data

Season data for the NBA was downloaded from public website Basketball-Reference.com that specializes in league datasets. Below is a sample of a resulting .csv file:

```
Source: http://www.basketball-reference.com/leagues/NBA_2014_games.html
Date,,Visitor/Neutral,PTS,Home/Neutral,PTS,,Notes
Tue Oct 29 2013,Box Score,Orlando Magic,87,Indiana Pacers,97,,
Tue Oct 29 2013,Box Score,Los Angeles Clippers,103,Los Angeles Lakers,116,,
Tue Oct 29 2013,Box Score,Chicago Bulls,95,Miami Heat,107,,
Wed Oct 30 2013,Box Score,Brooklyn Nets,94,Cleveland Cavaliers,98,,
Wed Oct 30 2013,Box Score,Atlanta Hawks,109,Dallas Mavericks,118,,
Wed Oct 30 2013,Box Score,Washington Wizards,102,Detroit Pistons,113,,
Wed Oct 30 2013,Box Score,Los Angeles Lakers,94,Golden State Warriors,125,,
Wed Oct 30 2013,Box Score,Charlotte Bobcats,83,Houston Rockets,96,,
Wed Oct 30 2013,Box Score,Orlando Magic,115,Minnesota Timberwolves,120,OT,
Wed Oct 30 2013,Box Score,Indiana Pacers,95,New Orleans Pelicans,90,,
```

Building Nodes

The first part of the Python script parses the file identifies each team as a Node object, updating a dictionary of losing teams and point differentials as each line of data is processed.

```
1.    class Node(object):
2.     def __init__(self, name):
3.        self.name = name
4.        self.losses = {} #Key = other team, value = sum margin of losses (point differential)
5.     def add_loss(self, oTeam, pointDiff):
6.        self.losses[oTeam] = self.losses.get(oTeam, 0) + pointDiff
7.        #Either adds a new loss, or updates the overall pointDiff
8.
9.  def build_graph(games):
10.    '''Processes all games and create each team as a node object
11.    INPUT: games-array of games in format games[i] = [loser, winner, pointDiff]
12.    OUTPUT: nodes - with nodes['team name'] = object node of team '''
13.    nodes = {} #dictionary Team Name : Team Object
14.    for item in games:
15.       loser = item[0]
16.       winner = item[1]
```

```
        pointDiff = item[2]
        nodes[loser]= nodes.get(loser, Node(loser))
        nodes[loser].add_loss(winner, pointDiff)
    return nodes
```

Markov Matrix and Left Eigenvector

With the loss data and corresponding point differential loaded into the nodes, the point differential matrix can quickly built; the team_index dictionary keeps track of which rows/column index belongs to which team, crucial for interpreting the results.

```
def build_matrix(nodes):
    """Builds the point differential matrix from nodes
    INPUT:
        nodes - with nodes['team name'] = object node of team
    OUTPUT:
        team_index - team_index['team name'] = index corresponding to row & column in A
        A: point differential matrix with A[row][column]
            Rows are losers, and entries are the point differential between column team
    '''
    A = [[0 for x in range(len(nodes.keys()))] for x in range(len(nodes.keys()))]
    teams = sorted(nodes.keys()) #array of teams alphabetically
    team_index = {}

    for i in range(len(teams)):
        team_index[teams[i]] = i

    for i in range(len(teams)):
        node = nodes[teams[i]]
        for item in node.losses.keys():
            pointDiff = node.losses[item]
            col_index = team_index[item]
            A[i][col_index] = float(pointDiff)
    return A, team_index
```

Once the point differential matrix is created, one can quickly create the Markov matrix and find the left eigenvector using SciPy's linear algebra functionality.

```
def markovMatrix(matrixA):
    """Creates Markov matrix by dividing each entry in A by the sum of the row
    INPUT: A - point differential matrix with A[row][column]
    Rows are losers, and entries are the point differential between col team
    OUTPUT: H - Markov matrix
    '''
    A = np.array(matrixA)
    H = [[0 for x in range(len(A))] for x in range(len(A))]
    for i in range(len(A)):
        row_sum = np.sum(A[i])
        for j in range(len(A[i])):
            H[i][j] = float(A[i][j])/row_sum
    return H

def pageRank(matrixA):
    """Returns the left eigenvector (the PageRank vector) of the input matrix
    INPUT: H - Markov matrix
    OUTPUT: vl - left eigenvector, or the PageRank vector
    '''
```

```
20.     A = np.array(matrixA)
21.     H = markovMatrix(A)
22.     w, vl, vr = linalg.eig(H, left = True)
23.     vl = np.absolute(vl[:,0].T)
24.     return vl
```

Results

SportsRank was applied to three complete seasons and compared to the playoff results and then used to predict playoffs for the ongoing season. Error is the absolute number of places off from the result (positive or negative), with a maximum of 8 if a ranked team did not even make it into the top 8 of final playoff results. For instance, if the NBA champion (#1 ranking in playoff results) was ranked by SportsRank to be 5th, the error would be 4.

2012-2013 Season

Playoff Results	Actual Season Rankings	Error	SportsRank	Error
Miami Heat	Miami Heat	0	Oklahoma City Thunder	4
San Antonio Spurs	Oklahoma City Thunder	3	Los Angeles Clippers	8
Indiana Pacers	San Antonio Spurs	1	San Antonio Spurs	1
Memphis Grizzlies	Denver Nuggets	8	Denver Nuggets	8
Oklahoma City Thunder	Los Angeles Clippers	8	Miami Heat	4
Golden State Warriors	Memphis Grizzlies	2	New York Knicks	8
Brooklyn Nets	New York Knicks	8	Houston Rockets	1
Houston Rockets	Brooklyn Nets	1	Memphis Grizzlies	4
Sum Error:		**31**		**38**

2013-2014 Season

Playoff Results	Actual Season Rankings	Error	SportsRank	Error
San Antonio Spurs	San Antonio Spurs	0	San Antonio Spurs	0
Miami Heat	Oklahoma City Thunder	1	Oklahoma City Thunder	1
Oklahoma City Thunder	Los Angeles Clippers	2	Los Angeles Clippers	2
Indiana Pacers	Indiana Pacers	0	Houston Rockets	8
Los Angeles Clippers	Houston Rockets	8	Indiana Pacers	1
Washington Wizards	Miami Heat	5	Phoenix Suns	8
Brooklyn Nets	Portland Trailblazers	1	Miami Heat	1
Portland Trailblazers	Golden State Warriors	8	Golden State Warriors	8
Sum Error:		**25**		**29**

For the 2012-2013 and 2013-2014 seasons, we see that both regular season rankings and SportsRanks produce large errors. We may be able to infer that neither system produces a good ranking, which is neither total wins nor a PageRank vector based on point differentials are good measures – however, there are a few systemic errors. The playoff system structured as a tournament with divided conferences significantly limits the performance of overall league rankings. In the 2012-2013 season, both regular season wins and SportsRank ranked San Antonio Spurs, the Oklahoma City Thunder, and the Los Angeles Clippers as the top three teams, respectively; all three of these teams are in the Western conference, and the playoff structure dictates that *only* one Western Conference team can be in the top two. Similarly, if one conference is significantly worse than the other then it's possible for the better conference to weaken its finalist while the other conference finalist enters the championship well rested and uninjured. Additionally, for more extensive analysis the rankings of all 30 teams should be considered with a more rigorous error measure.

Ultimately, it seems the search for an accurate sports prediction continues. Despite mixed results, SportRank's predictions for the current playoff season are below:

2014-2015 Season – Predicted

Actual Season Rankings	SportsRank
Golden State Warriors	Golden State Warriors
Atlanta Hawks	San Antonio Spurs
Houston Rockets	Cleveland Cavaliers
Los Angeles Clippers	Los Angeles Clippers
Memphis Grizzlies	Atlanta Hawks
San Antonio Spurs	Portland Trailblazers
Cleveland Cavaliers	Memphis Grizzlies
Portland Trail Blazers	Chicago Bulls

Class Questions

Major questions discussed in-class:

- **Did you consider for only the last two-thirds of the season? (Sanda Mong)** This is a great modification of the model, given the nature of sports – it completely weights the second two-thirds of the season, taking major injuries, trades, coaching changes, or teams "tanking" into account. For rankings that only considered the second two-thirds of the NBA season, all a games before January 1st of each season were removed (NBA regular season typically runs from the last weekend in October to mid-April). Here are the results using only games after January:

2012-2013 Season: Using Last 2/3rds of Season		
Playoff Results	**SportsRank, last two thirds**	**Error**
Miami Heat	Oklahoma City Thunder	4
San Antonio Spurs	Denver Nuggets	8
Indiana Pacers	Miami Heat	2
Memphis Grizzlies	Indiana Pacers	1
Oklahoma City Thunder	New York Knicks	8
Golden State Warriors	Brooklyn Nets	1
Brooklyn Nets	Los Angeles Clippers	8
Houston Rockets	San Antonio Spurs	6
Sum Error:		**38**

2013-2014 Season: Using Last 2/3rds of Season		
Playoff Results	**SportsRank, last two thirds**	**Error**
San Antonio Spurs	San Antonio Spurs	0
Miami Heat	Oklahoma City Thunder	1
Oklahoma City Thunder	Los Angeles Clippers	2
Indiana Pacers	Miami Heat	2
Los Angeles Clippers	Chicago Bulls	2
Washington Wizards	Golden State Warriors	8
Brooklyn Nets	Portland Trail Blazers	1
Portland Trailblazers	Brooklyn Nets	1
Sum Error:		**17**

Although the sum error remained the same for the 2012-2013 season compared to using full season results, the error dropped significantly from using only the last two-thirds of 2013-2014 season. Further research to determine the most representative set of data (best subset of games) could provide more accurate rankings.

- **Rankings for other sports? (Hockey was suggested as a good comparison to the NBA with similar number of games and playoff structure)**
  To follow-up on this, the Python code can quickly be adapted and applied to the National Hockey League data. Here are some results:

National Hockey League, 2011 -2012 Regular Season				
Playoff Results	**Actual Season Rankings**	**Error**	**SportsRank**	**Error**
L.A. Kings	Vancouver Canucks	8	Pittsburgh Penguins	8
New Jersey Devils	New York Rangers	1	Boston Bruins	8
New York Rangers	St. Louis Blues	4	Detroit Redwings	5

Phoenix Coyotes	Pittsburgh Penguins	8	New York Rangers	1
Washington Capitals	Nashville Predators	1	St. Louis Blues	2
Nashville Predators	Philadelphia Flyers	8	Philadelphia Flyers	8
St. Louis Blues	Boston Bruins	8	Vancouver Canucks	8
Detroit Redwings	Detroit Redwings	8	New Jersey Devils	8
Sum Error:		**48**		**48**

National Hockey League, 2012 -2013 Regular Season				
Playoff Results	**Actual Season Rankings**	**Error**	**SportsRank**	**Error**
Chicago Blackhawks	Chicago Blackhawks	0	Chicago Blackhawks	0
Boston Bruins	Pittsburgh Penguins	2	Anaheim Ducks	8
L.A. Kings	Anaheim Ducks	8	St. Louis Blues	8
Pittsburgh Penguins	Montreal Canadians	8	Detroit Redwings	2
San Jose Sharks	Boston Bruins	3	L.A. Kings	2
Detroit Redwings	St. Louis Blues	8	Vancouver Canucks	8
Ottawa Senators	L.A. Kings	3	San Jose Sharks	3
New York Rangers	Vancouver Canucks	8	Dallas Stars	8
Sum Error:		**40**		**39**

Again, both rankings produce very high errors, even higher than the NBA seasons. A significant difference between the NBA and the NHL is the point differentials, or the arcs within the model and eventually the values in the matrix. In the NBA, the average margin of victory is 10.78 points[1], whereas the average differential in the NHL is 0 points[2] – the NHL, where teams typically score in the low single digits, have much closer games resulting in more unpredictable results.

Professor's Questions: Dr. Adam Stinchcombe

- **Why is NBA team ranking with a Markov chain better or worse than other methods of ranking? (i.e. w/r/t/ record)? Why point differential instead of just win/loss?**
  Markov chains provide insight about the relationships and transitions between nodes. Theoretically, a Markov chain ranking could outperform the standard win/loss ranking in predicting playoff results by using another measure that is more representative of a team's true value. Point differential takes the magnitude of loss into account, penalizing teams that have larger margins of loss and taking who the team lost to into account. Although results were inconclusive, if one could identify a sufficient measure or adjust the model, Markov chain analysis allows more information than just number of wins to be consider, such as who won over whom and margin of victory.

[1] http://www.sportsonearth.com/article/74360454/2014-nba-playoffs-most-game-sevens-quantifying-game-quality
[2] http://www.sportingcharts.com/nhl/stats/team-goal-differential-per-game/2013/

- **Your model assumes a point differential in one game is the same value as another game (against the same team). Is this a good model?**
  This setup aligns SportsRank with the win-loss ranking, in which a single game result is equal to another game result against the same team regardless of when the game is played. However, this method is not necessarily a good model – the suggestion of weighting games played later in the season is a very efficient way to improve the model – one can simply use a reduced dataset or weigh the second half of the season more. Another method that would require significant manual work would be to pinpoint specific games that are outliers and weigh or remove them as needed, such as a single game where a star was injured or where the coach chooses to sit and rest players. Identifying and analyzing outliers in context of broader results such as total win-loss is often done by media's sports analysts and journalists when writing long-form playoff prediction or team evaluation articles.

- **Could you weight the scores? (counting blowouts, i.e. a 40 point blowout would not count as much as forty 1-point wins)**
  Yes, another potentially valuable adjustment would be weighing the scores – one could do this using the average league loss margin and comparing the point differential, and choosing to weigh close games more heavily than blowouts. Another efficient way would be to cap the maximum point differential; all loss margins over 20 points are simply inputted as a 20 point differential, which assumes that the losing team gave up and the points beyond that cap are relatively meaningless to analyze team value.

- **What about absolute scores instead of point differential?**
  Point differential is a better measure than absolute scores because it weighs low-scoring and high-scoring wins equally without penalizing defensive-minded teams. Teams often strategize around the strengths of players, implementing a "high powered offense" that attempts to create a fast-paced game (meaning lots of possessions for each team) and out score opponents – this produces higher absolute scores. Alternatively, other teams emphasize defense, low turnovers, and long possession times, resulting in low absolute scores. Point differential allows a 70-60 win to be equal to a 120-110 point win since both have a point differential of 10. Determining whether point differential or absolute score would be a better measure for different leagues would be dependent on the variations of total score; for the NBA, there is significant variation of high-scoring and low-scoring games.

- **Which leagues best fit your model? (also asked by Irvind Narayan)**
  Unfortunately, it is unclear whether any leagues best fit the model – I opted for the NBA because playoff results most often align with regular season results, compared with other leagues. A significant assumption is the belief that sports playoffs can predicted and modeled based on regular season results; this may not be a valid assumption since all professional sports leagues have significant economic incentive to implement playoff structures that are *not* predictable and

produce "unlikely" winners, such as elimination tournaments. Thus, regular season results are not necessarily fully indicative of post-season success, but instead determine which subset of teams will be participating in playoffs.

Additionally, it's important to choose the measure used for the arcs in context of the league. The NHL example showed that point differential is not a good measure when the majority of games are decided by a single point – perhaps an absolute score would provide a better measure in this case. This model has worked well with ranking single athletes in sports such as tennis and golf; this may be because with a single person, the number of points won or shots under par is very representative of the player's value and produces successful ranking.

- **How do you normalize the rows of the point differential matrix? What model assumptions does that entail?**
  The rows are normalized manually (entry by entry) within the Python script by dividing each entry by the sum of the row. This assumes that each summed loss margin to an opposing team is weighted equally. One oversight in this model is that for teams with very few losses would have high row entries, signifying that if one were to transition out of that node, there would only be few possible next nodes with relatively high probabilities; that is, if Team A only had three losses, the arcs would have high values relative to Team B with thirty losses. If both Team A lost to Team B, then Team A would have a "higher probability" of 1/3 to transition to node C (or losing to Team C) relative to Team B's 1/30 probability.

Questions about Google's PageRank

- **Is there a separate ranking for different types of articles/pages? Does a PageRank algorithm rank # of citations higher than # of figures? (Irvind Narayan, Sanda Mong)**
  Today, Google's web crawlers are extremely advanced and able to use content analysis to discern between different types of nodes such as company webpages, academic papers, or social media, taking figures and hyperlinks into account. Notably, images are often tagged by web-crawlers (for Google's image search) but the specific weighting is not public.

- **Did you pursue how people manipulate their way to the front page? (David Hodgson, Chad Newton, Sanda Mong)**
  Once businesses recognized that being highly ranked within Google's search engine results would significantly increase user traffic, a technique known as "Search Engine Optimization" (SEO) evolved – specific techniques to optimize a page to appear in search engine results.

  Due to Brin and Page's early publication, companies began falsely implanting links to attempt to raise the number of outlinks. Further, knowledge about how webcrawlers work provide insight on additional techniques such as embedding links or content in HTML code that are not actually shown on the page.

Currently, Google monitors these techniques and has removed company websites from search results to punish "black hat" SEO techniques (JC Penny was forced to rework their website in 2011 after notices from Google). Acceptable techniques include maintaining quality and up-to-date content. Because of the incentive to "trick" PageRank, Google updates and protects its proprietary algorithms.

- **Can the ranking be used in other scenarios? (Mengyero Gino , Zicheng (Kevin) Bi)**
  Yes, PageRank has proven useful in analyze complex systems with numerous nodes. Interesting applications include GeneRank and ProteinRank to examine underlying scenarios when multiple genes contribute to the expression of a single trait - PageRank Beyond the Web, by David F. Gleich covers several examples.

- **How much does search history change search results? (Brianna LaBelle Hahn)**
  Google's current PageRank algorithm certainly weights search history, of both the user and traffic as a whole; the early PageRank model could not leverage the significant user data that Google has collected over the past decade. The extent to which it weighs search history is likely to depend on the types of terms being searched – i.e. searching restaurants produces local results relevant to your location, whereas a search such as "politics" is less reliant on search history and more likely to bring up recent news or more general results.

- **How important is the weighting factor? (Brianna LaBelle Hahn, David Hodgson)**
  The weighing factor c is simply a parameter that allows for dangling nodes to be taken into account – otherwise, the Markov matrix produced may not be diagonalizable (since one can get "stuck" at a dangling node with no outward transitions). The parameter can be manipulated depending on the content of a directed graph; for instance, if one entered the search term "Microsoft product launch", c may be reduced weighing webpages with no outlinks more; thus, official Microsoft press releases would be ranked higher. One can also set $c = .50$ to equally weigh dangling nodes with other interconnected nodes.

- **If we have several key words when we do the search on Google, how does Google rank their pages based on these key words? (Siyuan Wie)**
  Keywords help "seed" the algorithm and provide the algorithm a starting point to rank possible relevant results; conceptually, it would help Google determine which node or webpage to start from and the most likely transitions to other webpages. Now that Google has indexed millions of webpages and has significant past data, search words can also return a smaller directed graph that would produce better results. For instance, if someone searched "University of Michigan Mathematics Department", instead of considering billions of webpages on every topic, Google's PageRank could instead only consider the subset of

webpages with an .edu domain.

- **Is it "PageRank" specifically or how does this differ from an ordinary Markov chain analysis? (John Han)**
  PageRank, at the time, was an unusual application of Markov chain analysis. It interprets the probabilities within steady-state eigenvector as a ranking, with the higher probability of transition to a state as a higher ranking; useful because it allows Markov chain analysis to be applied in different settings.

- **Where/how did you find the information about how Google's PageRank works? (Taylor Vender)**
  The earliest (and simplest iteration) PageRank was initially published as an academic endeavor by Google's founders at Stanford that later evolved to a full search engine and web-crawler. As Google's significance increased, becoming an almost universal portal to the internet for all users, more research about how Google's ranking worked and what individuals could do to both get more relevant results (by inputting better search terms) and what websites could do to improve their ranking. The exact algorithm is considered intellectual property and protected by Google to ensure a level of user trust with search engine results, but the mechanism of using directed graphs a Markov analysis to study complex systems is widely researched.

Questions about the Model

- **Did you find the point different manually? (Ruilin Zhang)**
  Point differential was calculated from final game results during the data extraction.

- **How does the number of times teams play each other affect the model? (Ruilin Zhang)**
  The number of times teams play each other in a season is taken into account since the margin of all losses between teams is summed; that is, if team i loses to team j twice, the arc from i to j is the sum of the point differential of both games. An alternative that would not weight the number of games between teams would be to use average the point differential across losses rather than the sum.

- **Have you considered adding other factors like field goal percentage, fouls, etc. to the model & weighing their significance? Home v Away? Intangibles? (David Hodgson, Chad Newton)**
  Although additional measures could be considered, the value of PageRank is the potential of a comprehensive ranking with a single measure based on the idea that the single measure is representative of the other factors. For instance, it assumes better field goal percentage would result in fewer losses and lower point differentials (from more successful scoring) and is thus incorporated within the measure. Similarly, the model assumes that intangibles is represented in margin of losses – that is, good team chemistry and "grit" would

result in more wins and small margin of losses (ability to keep games close).

- **How feasible is this to expand to all teams? (Uziel Mendes, Matthew Stanulet, Psvatik Holla)**
  Since all the nodes are connected and every league has a limited number of teams, considering all teams is straightforward (however, have to be aware of dangling nodes for undefeated teams). Sports leagues have around 30 teams, so the Markov matrix can be quickly constructed and manipulated at that size. However, with matrices and linear algebra, the data and processing requirements grow exponentially with size –extremely large matrices and significant calculations can cause hindered performance. Technical decisions such as programming languages used, distributed computing, and problem structure can be optimized to work with enormous datasets, although performance and hardware constraints will always be part of solution design.

- **Why did you pick this simple model to use? What was most interesting about PageRank? (Susannah Engdahl, Taylor Vender)**
  PageRank interested me because it uses a very simple measure to quickly rank billions of webpages, requiring only one linear algebra operation. Ultimately, the assumption that a single representative measure could the only valued needed to rank multiple nodes allows one to gain significant conclusions about complex networked systems and limited information.

  Sports teams were used due to personal interest and the challenge – sports playoff results are notoriously hard to predict (gambling around these predictions is a billion dollar industry), especially with math because of the infinite number of variables involved with every athlete, game, and play and the incentive for leagues to create unpredictable scenarios.

- **How different would the model be when using sports where individuals are ranked, such as golf? (Matthew Stanulet, Brianna LaBelle Hahn)**
  Other publications have shown individual rankings to be more successful with PageRank – with only a single player and head to head matches, there are fewer variables (such as injuries affecting a team or trades) that allow rankings to better predict playoffs. A more thorough exploration would involve multiple sports over several seasons, with a rigorous error measurement system.

- **How will you quantitatively assess the accuracy of the model? (Brianna LaBelle Hahn)**
  This was also a significant shortcoming of the model – I used an absolute displacement from the top-8 rankings to measure error. A more thorough error measure taking all teams into account is necessary to sufficiently consider the model.

Another valuable adaptation would be to determine a "success" measure in the playoffs, using the results of the playoffs determine a "success" measure or probability (with the winning team having the highest level of "success") instead of a straight ranking – then SportsRank or standard win-loss ranking is used to derive these measures instead of a rigid predicted playoff ranking.

- **What happens if two teams don't play each other? (Taylor Vender)**
  Ranking can still be achieved because the model sees the league as a closed network. Since the teams are nodes, the existing connections between nodes (results of when two teams do play each other) provide insight about unconnected nodes. This model accounts for connects, meaning it can infer from existing data that if Team A lost to Team B and Team B lost to Team C, Team A is likely to lose to Team C (even though they never played) and should be ranked lower.

- **How would a tie be dealt with in the regular season for the sports that do not include tie breakers? (Sam Tuck)**
  A tie would be simply be considered a 0 in the matrix, or if the teams played each other more than once it could be accounted for by reducing the weight of the remaining summed point differential from the other games. Given the rarity of ties games, however, they could be thrown out and considered outliers – notably, the standard ranking system with wins-losses does not account for ties either (simply washout in the standings relative to the other number of wins, with ties counting more than losses when two teams have the same number of wins).

Conclusion

Although SportsRank did not provide significant improvement over the standard win-loss ranking, several interesting explorations were made. The appeal and elegance of PageRank is clear – if one can identify a single measure that is representative of the value of a node (for whatever value is defined to be by the user), then an extensive directed graph with millions of nodes can be efficiently analyzed using linear algebra and Markov chain analysis. This application of Markov chains allows conclusions about complex networks not only due to the analysis based on a single measure, allowing networks to be quickly evaluated by multiple measures by building multiple Markov simulations, but also implemented quickly due to the ability for computers to work with extensively matrices.

Unfortunately, applying to PageRank to sports was not necessarily an appropriate model and did not improve on the standard win-loss ranking. SportsRank uses the measure of margin of losses to take the magnitude of a loss into account and applies PageRank to create a ranking of teams. A superficial error measure showed

significant displacement from SportsRank's results and standard win-loss rankings to actual playoff results. Questions from the class and additional simulations provide some discernment to the model's shortcomings. Currently, both SportsRank and the standard win-loss ranking weigh each game equally – an insightful suggestion to weigh the latter half of the season or later games more heavily showed improvement on the model. Additionally, creating bounds on the point differential could help normalize the data by not weighing points earned in "garbage time" when games were already decided and gameplay would not be representative of playoff performance.

A significant assumption for both ranking methods is that regular season data is predictive of playoff results. Given that all professional sports leagues have economic incentive to create unpredictable playoff structures and the infinite number of variables contributing to each game, a better assumption may be to consider the regular season a "first round" of playoffs and simply a mechanism to reduce the pool of competition before broader playoff analysis. Similarly, rather than comparing the ranking to the playoff ranking, a better method may be to use the left eigenvector to consider the probability of success in playoffs, recognizing that probabilistic predictions account for the variation in sports. Extended exploration would include developing a rigorous error measurement to and playoff "success" measure to better compare simulated results to reality and thus better adapt the model to fit until predictions are more successful. Furthermore, SportsRank should be applied a wide variety of leagues and utilize substantial amounts of past season data to improve the model. Ultimately, SportsRank provided insight on Markov analysis within directed graphs and how conclusions could be drawn from a single measure, but the search for a predictive algorithm for athletics continues.

Sources

Ranking National Football League Teams Using Google's PageRank, by Angela Y. Govan and Carl D. Meyer (Additional slidedeck presented from ACES Workshop)

PageRank Beyond the Web, by David F. Gleich

The predictive power of ranking systems in association football, by Jan Lasek, Zotan Szlavik, and Sandjai Bhulai

Markov Chains, Google's PageRank Algorithm, by Jeff Jauregui, UPenn Math Slidedeck

The PageRank Citation Ranking: Bringing Order to the Web by Larry Page, Sergey Brin, et al.

Appendix: NBApageRank.py script

```
from sys import argv
import re
import random
import pprint
import numpy as np
from scipy import linalg

class Node(object):
    def __init__(self, name):
        self.name = name
        self.losses = {} #Key = other team, value = sum margin of losses (point differential)

    def add_loss(self, oTeam, pointDiff):
        self.losses[oTeam] = self.losses.get(oTeam, 0) + pointDiff
        #Either adds a new loss, or updates the overall pointDiff

def load_data(my_file):
    """Reads input file and creates nodes. Check that indicies align with data;
    filename is read in as an argument from terminal (see main() function)
    INPUT: my_file - raw csv data of games
    OUTPUT: games - array of games in format games[i] = [loser, winner, pointDiff]"""
    f = open(my_file)
    filetext = f.readlines()
    f.close()
    filetext = filetext[2:] #eliminate header lines
    games = []

    for line in filetext:
        gameData = line.split(',')
        if int(gameData[3]) < int(gameData[5]):
            loser = gameData[2]
            winner = gameData[4]
            pointDiff = int(gameData[5]) - int(gameData[3])
        else:
            winner = gameData[2]
            loser = gameData[4]
            pointDiff = int(gameData[3]) - int(gameData[5])
        games.append([loser, winner, pointDiff])
    return games

def build_graph(games):
    '''Processes all games and create each team as a node object
    INPUT: games - array of games in format games[i] = [loser, winner, pointDiff]
    OUTPUT: nodes - with nodes['team name'] = object node of team '''
    nodes = {} #dictionary Team Name : Team Object
    for item in games:
        loser = item[0]
        winner = item[1]
        pointDiff = item[2]
        nodes[loser]= nodes.get(loser, Node(loser))
        nodes[loser].add_loss(winner, pointDiff)
    return nodes

def build_matrix(nodes):
    '''Builds the point differential matrix from nodes
    INPUT:
        nodes - with nodes['team name'] = object node of team
    OUTPUT:
```

```
    team_index - team_index['team name'] = index corresponding to row & column in A
    A: point differential matrix with A[row][column]
      Rows are losers, and entries are the point differential between column team
  '''
  A = [[0 for x in range(len(nodes.keys()))] for x in range(len(nodes.keys()))]
  teams = sorted(nodes.keys()) #array of teams alphabetically
  team_index = {}

  for i in range(len(teams)):
    team_index[teams[i]] = i

  for i in range(len(teams)):
    node = nodes[teams[i]]
    for item in node.losses.keys():
      pointDiff = node.losses[item]
      col_index = team_index[item]
      A[i][col_index] = float(pointDiff)
  return A, team_index

def markovMatrix(matrixA):
  '''Creates Markov matrix by dividing each entry in A by the sum of the row
  INPUT: A - point differential matrix with A[row][column]
      Rows are losers, and entries are the point differential between col team
  OUTPUT: H - Markov matrix
  '''
  A = np.array(matrixA)
  H = [[0 for x in range(len(A))] for x in range(len(A))]
  for i in range(len(A)):
    row_sum = np.sum(A[i])
    for j in range(len(A[i])):
      H[i][j] = float(A[i][j])/row_sum
  return H

def pageRank(matrixA):
  '''Returns the left eigenvector (the PageRank vector) of the input matrix
  INPUT: H - Markov matrix
  OUTPUT: vl - left eigenvector, or the PageRank vector
  '''
  A = np.array(matrixA)
  H = markovMatrix(A)
  w, vl, vr = linalg.eig(H, left = True)
  vl = np.absolute(vl[:,0].T)
  return vl

def printResults(vl, team_index):
  '''Uses the team_index dictionary to interpret the results from the PageRank vector
  INPUT:
    vl - left eigenvector, or the PageRank vector
    team_index - team_index['team name'] = index corresponding to row & column in A
  OUTPUT:
    top10: arrray of top10 teams based on PageRank vector
  '''
  top10= []
  teams = sorted(team_index.keys())
  for i in range(10):
    ind = np.argmax(vl)
    top10.append(teams[ind])
    vl[ind] = 0
  return top10
```

```
def main():
    script, file1 = argv
    nodes = build_graph(load_data(file1))
    A, team_index =  build_matrix(nodes)
    pi = pageRank(A)
    pprint.pprint(pi)
    pprint.pprint(printResults(pi, team_index))

if __name__ == '__main__':
    main()
```