LS88: Sports Analytics

Ranking Systems

Why ranking systems?

- ➔ In the NBA and the NHL, each pair of teams play at least twice
- ➔ In MLB, a team is only guaranteed to play teams within its own league
- ➔ In the NFL, a team is only guaranteed to play teams within its division
- ➔ Division 1A college football has 129 teams. They play 12 games each.

Ideally teams would play a round robin. That's not possible.

Given diverse performances & schedules and relative lack of games played, how do we build a ranking under these conditions?

Elo Ratings

- Elo was originally developed for Chess
- In recent years, it’s been heavily adapted for all sorts of sports
- Nate Silver and 538 are some of the most avid users of Elo ratings

Elo Ratings

Two teams with ratings R_A and R_B

Probability of winning:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}} = \frac{Q_A}{Q_A + Q_B}$$

$$E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}} = \frac{Q_B}{Q_A + Q_B} = 1 - E_A$$

$$Q_A = 10^{R_A/400}, \quad Q_B = 10^{R_B/400}$$

Elo Ratings

- ➔ Outcome $S_A = 1 - S_B$ (typically 1 for win, 0 for loss)
- ➔ Adjustment rate *K*, aka the *K*-factor

Updated ratings:

$$R'_A = R_A + K \times (S_A - E_A),$$
$$R'_B = R_B + K \times (S_B - E_B)$$

Principles of Elo Ratings

- Beat opponents better than you, your rating rises a lot
- Lose to opponents worse than you, your rating falls a lot
- Your rating updates every time you play
- Unlike some other methods we'll see, you always have a rating and the update can always be made
- Directly takes into account opponent strength: your rating can't go up if you have a weak schedule

Matrix/Regression Ratings

- ➔ Matrix ratings are very much different from Elo
- ➔ We can either solve a system of equations or devise a regression model
- ➔ The Colley and Keener methods solve a system of equations
- ➔ The Massey and Bradley-Terry methods use a regression model

Massey Method

- ➔ Massey Regression Equation

$$\text{Home Score - Away Score} = \text{Home-Field Advantage} + \sum_{\text{All Teams}} \text{Team } i \text{ Rating} \times \text{Team } i \text{ is at Home}$$
$$- \sum_{\text{All Teams}} \text{Team } i \text{ Rating} \times \text{Team } i \text{ is Away}$$

- ➔ The rating is like the APM or RAPM we computed for the NBA
  - We can and probably should use a penalized regression like with RAPM
- ➔ Unlike plain Elo, takes into account margin of victory

Bradley-Terry Method

- ➔ Bradley-Terry Logistic Regression Equation

$$\text{Log Odds for Home Team} = \text{Home-Field Advantage} + \sum_{\text{All Teams}} \text{Team } i \text{ Rating} \times \text{Team } i \text{ is at Home}$$
$$- \sum_{\text{All Teams}} \text{Team } i \text{ Rating} \times \text{Team } i \text{ is Away}$$

- ➔ The right-hand side is the exact same
- ➔ The left-hand side is strange: we can't observe the log-odds
- ➔ We're trying to predict wins and losses: a good set of ratings does a good job of predicting who won the game (higher the log-odds, higher the probability)
- ➔ The key to logistic regression: find coefficients to make the RHS large/positive when the home team won and small/negative when the home team lost

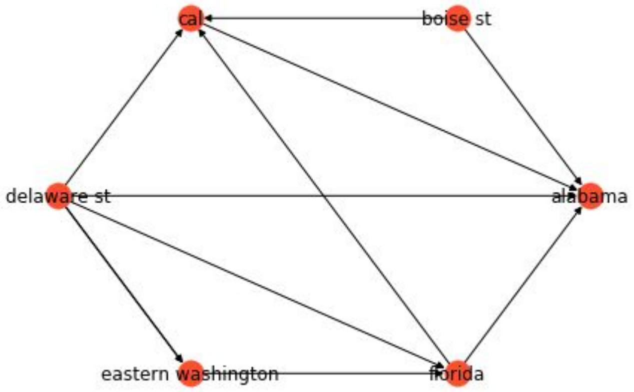
Graph Methods (aka Markov Chain Methods)

A *graph* or *network* is a way to spatially represent relationships

Nodes: entities like teams or players

Edges: relationships like a match/game

- ➔ An edge can be directed, indicating a hierarchy in the relationship
- ➔ An edge can be weighted (like margin of victory)
- ➔ An edge can be repeated (for multiple matchups)



Graph Methods (aka Markov Chain Methods)

Why are they called Markov Chain methods?

We’re going to model the ranking through a random walk process along the graph

A random walker traversing a graph is type of *Markov Chain*

The defining feature of *Markov Chains*: the simulation only depends on the current state and not at all on previous history

PageRank

The original basis for the Google search engine

- ➔ The Random Walker: traverses the network by picking an outbound edge and walking to the next node
- ➔ Random Jump: every so often, the walker picks a random node from the whole network and jumps directly there (regardless of edges)
  - ◆ This ensures the walker can't get trapped. If edges are games, Alabama, Clemson, and ND have no outbound edges so the walker would get trapped without the jump
  - ◆ A typical jump probability is about 15% of the time

PageRank

- ➔ Run to infinity, count the proportion of visits to a node to get the ranking
  The more visits, the more important, the higher the ranking
- ➔ For small graphs, the walker simulation is okay
- ➔ For large graphs (like our CFB example or the internet), we need math
  “Infinity” is not possible through the walker simulation: instead solve a linear algebra problem

PageRank

Purdue beat Ohio State (their only loss)

- ➔ When the walker gets to OSU, if it doesn't jump it'll go directly to Purdue
- ➔ OSU is visited a lot so Purdue is subsequently visited a lot
- ➔ Purdue isn't good but gets a *huge* rating boost

This is a fatal flaw of PageRank for CFB ranking

In general, there are a lot of issues that crop up with PageRank and in its pure, original form, it is woefully inadequate for ranking web pages. It's unclear what Google uses at this point.

MonkeyRank

An alternate approach (from a friend of mine)

- ➔ The random walker (now a monkey) doesn't go to teams that beat the current team, but rather picks an opponent (won or lost) at random and favors that team depending on a coin flip
- ➔ The monkey will switch allegiance to the winner of the matchup if the coin flip comes up heads
- ➔ The probability p of heads is a model choice and values can range from .5 to 1

A linear algebra problem can be solved to produce the MonkeyRank

Ranking Systems

- No shortage of ranking systems
- There can be some really wonky things that show up due to the data
- There are certain desirable features you could want
- You can also *ensemble* the rankings into a composite
- If in doubt: do what 538 does and start with Elo