Champion Recommendation Models

Daniel McFalls

May 4, 2019

This paper on League of Legends presents various formal data-driven models for recommending a champion to pick during champion-select based on a player's match history and the existing picks on the two teams.

1 Preliminaries

Let us define the following:

- Let N be an ordered collection of all champions in League of Legends.
- An N_i denotes the champion at index i in N
- Define the number of champions as $n = |N|$

Assume the following data is available for some summoner S:

- Let M be a vector containing the match history for S
- Let each M_i be a game G
- Each G contains a list of five champions $G[S]$ including a champion played by S, and a list of five champions $G[O]$ on the opposing team
- Each G also contains G_w, defined to be `true` if S won the match, and `false` otherwise

Now we define the following:

- Let F be an $n \times n$ matrix where F_{ij} denotes the number of games S has played as champion N_i against champion N_j
- Let T be an $n \times n$ matrix where T_{ij} denotes the number of games S has played as champion N_i on a team with N_j
- Let W be an $n \times n$ matrix where W_{ij} denotes the number of wins that S has on champion N_i against champion N_j
- Let V be an $n \times n$ matrix where V_{ij} denotes the number of wins that S has on champion N_i on a team with N_j

2 Win-rate Models

Here we propose a few models of varying complexity for answering the question, "If I'm picking my champion and some champions are already picked to be on my team and on the opposing team, what should I pick to ensure my highest chances of victory?"

2.1 Simple Independent Win-rates

A simple intuitive approach to recommending a pick at an arbitrary point along the pick-ban phase is to take the pick that synergizes best with existing team mates' picks and performs best against the opposing team's picks.

We assume, for this first model, that win-rates between champion pairs can be treated independently of which other champions appeared in the games.

Furthermore, this first model will only consider relationships between pairs of champions.

Finally, we incorporate only data from the match history of S and how S has performed with and against other champions on a given champion. This restricts recommendations to champions that S has played.

Now, we define a $\texttt{Score}_i$ for S on champion N_i as follows:

$$\texttt{Score}_i = \prod_{j \in G[S]} \frac{V_{ij}}{T_{ij}} \times \prod_{j \in G[O]} \frac{W_{ij}}{F_{ij}} \tag{1}$$

This number is simply the product for S of win-rates with champions on the allied team and of win-rates against champions on the opposing team.
Now it is easy to define a `Recommendation` as follows

$$\texttt{Recommendation} = \underset{i}{\operatorname{argmax}}(\texttt{Score}_i) \tag{2}$$

Any k recommendations can be generated by taking each of the $k^{\texttt{th}}$ order statistics from the collection of available `Scores`.

2.2 Independent Win-rates

Interactions between champions extend beyond 1:1 relationships. For example, it would be nice if our model could take into account synergy with Malphite *and* Orianna or advantage against Xayah *and* Rakan.

A simple way to accomplish this is to populate additional structures V, T, W, and F which is queried by, instead of a pair of champion-indices, a set of champion-indices.

For example, consider generalized V whose superscript denotes $|C| - 1$. We define V^3 to be an $n \times n \times n$ matrix where each V^3_{iC} denotes the number of wins that S has on champion N_i playing on a team with the champions in C.

For V we can now populate the generalized structures V^2 up to V^5. These structures are space-intensive, but not prohibitively so. They can all be generated efficiently by iterating through the match history of S.

It is now possible to extend our model for $\texttt{Score}_i^k$ to include the generalized win-rate structures defined above:

$$\texttt{Score}_i^k = \prod_{|C|=k, C \in G[S]} \frac{V_{iC}}{T_{iC}} \times \prod_{|C|=k, C \in G[O]} \frac{W_{iC}}{F_{iC}} \tag{3}$$

The adjusted formula for $\texttt{Recommendation}_k$ follows:

$$\texttt{Recommendation}_k = \operatorname*{argmax}_i(\texttt{Score}_i^k) \tag{4}$$

3 Complex Models

TODO: develop more complex models and add additional sections describing them

4 Analysis

TODO: analyze time−complexity and memory−complexity for the models discussed