Player Rankings to Inferred Position Probabilities

1 Summary

One way to predict win probability for a group of players is to predict the rank of them. There are ample techniques to achieve this, however, there is no clear way to transform the predicted rank into a valid probability distribution. This work proposes a method which transforms predicted rankings of arbitrary number of players into a inferred doubly stochastic position matrix, where the expected rank of each player equals to the predicted ranks. Python implementation may be found under `https://github.com/xl402/tools`

2 Methods

Denote number of players as N with i denoting the index of a player within a game. X_i being the actual position of a player at the end of a game and $\mathbb{E}[X_i] = r_i$ being the expected position of a player i.e. the rank of a player. Finally $P \in \mathbb{R}^{N\times N}$ denotes the doubly stochastic matrix where P_{ij} denotes the probability of player i finish in position j.
The constraints are therefore:

$$\sum_{j=1}^{N} jP_{ij} = r_i, \quad \forall i \tag{1}$$

$$\sum_{j=1}^{N} P_{ij} = 1, \quad \forall j \tag{2}$$

$$\sum_{i=1} P_{ij} = 1, \quad \forall i \tag{3}$$

which can be formulated into a convex optimisation problem. Denote:

$$\begin{aligned}
M_1 &= I \otimes \begin{bmatrix} 1 & 2 & \dots & N \end{bmatrix} && (4)\\
M_2 &= I \otimes \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} && (5)\\
M_3 &= \begin{bmatrix} I & \dots & I \end{bmatrix} && (6)\\
\mathbf{t} &= \begin{bmatrix} r_1 & \dots & r_N & 1 & \dots & 1 \end{bmatrix}^T && (7)\\
X &= \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} && (8)
\end{aligned}$$

where $I \in \mathbb{R}^{N\times N}$, $\otimes$ denotes the Kronecker product. Hence $M_1, M_2, M_3 \in \mathbb{R}^{N\times N^2}$ and $X \in \mathbb{R}^{3N\times N^2}$. Problem formulation becomes as follows:

$$\begin{aligned}
\min_{P} \quad & XP - \mathbf{t} && (9)\\
\text{s.j.} \quad & 0 \leq P_{ij} \leq 0 \quad \forall i, j && (10)
\end{aligned}$$

which can be solved using any off-the-shelf convex optimisation package.