Modelagem do Tênis

Igor Patrício Michels

21 de dezembro de 2020

1 Introdução

O presente documento visa relatar o desenvolvimento da modelagem do tênis, buscando analisar a linearidade do mesmo, isto é, se um jogador A ganha de um jogador B, o qual ganha de C, podemos afirmar que A ganha de C?

Os dados utilizados podem ser encontrados em [2], enquanto os códigos utilizados podem ser vistos em [1].

2 Metodologia

Em primeiro lugar, fiz um tratamento de dados, criando algumas funções que auxiliam na captação dos dados, de um arquivo `.csv` para um DataFrame do pandas, as quais podem ser encontradas no arquivo `data_functions.py`. Nesse arquivo temos as seguintes funções:

- `catch_players`: recebe o arquivo `.csv` com o ranking e um valor n representando quantos jogadores queremos e retorna uma lista com esse top n jogadores do ranking;
- `catch_games`: recebe um arquivo com os jogos de um ano, a lista de jogadores retornada pela função anterior, a superfície[1] desejada e a quantidade de sets desejada, retornando um DataFrame com todos os jogos entre os jogadores da lista de entrada que satisfazem as restrições de superfície e de sets;
- `catch_all_games`: generalização da função anterior, recebendo uma lista de arquivos ao invés de um arquivo só;
- `split_games`: divide um DataFrame de jogos em dois, possibilitando dividir os jogos entre jogos para fi do modelo e jogos para testar o modelo;
- `catch_data`: recebe o DataFrame com todos os jogos e um booleano para ver se iremos retornar dados para fit (preparado para otimização) ou para ver a eficácia do modelo (retornando apenas os resultados e os jogos).

Tendo feito o tratamento dos dados, podemos fazer a modelagem. Para tanto, defini, no arquivo `model_functions.py`, as seguintes funções:

- `find_probability`: calcula a probabilidade de um jogador A, com parâmetros (a_1, a_2) ganhar de um jogador B, com parâmetros (b_1, b_2) através da expressão

$$P\left(A \text{ vencer } B\right) = \frac{\exp a_2 \cdot b_1}{\exp a_2 \cdot b_1 + \exp b_2 \cdot a_1};$$

- `find_parameter`: recebe a probabilidade de um jogador A ganhar de um jogador B e retorna um dos possíveis conjunto de parâmetros;[2]

[1] Tipo de quadra.

[2] Essa função acabou não sendo utilizada.

- `likelihood`: recebe uma lista de jogadores (cada elemento dessa lista é um vetor com os parâmetros do jogador) e os resultados dos jogos, retornando a log-verossimilhança negativa dos dados observados com os parâmetros dados.

Feito isso, podemos utilizar a biblioteca `scipy` para achar os parâmetros que minimizam a log-verossimilhança negativa nos dados de fitagem. Ao realizar o cálculo da verossimilhança com os dados de teste obtemos um valor inferior ao resultante no fit, o que nos leva a inferir que o modelo está no caminho certo.

Por fim, realizei um teste de linearidade, pegando todos os possíveis trios de atletas e buscando por trios A, B e C de modo que A ganhe de B, B ganhe de C e C ganhe de A ou que B ganhe de A, A ganhe de C e C ganhe de B. Fazendo isso, percebi que aproximadamente 95% dos trios apresentaram linearidade.

Referências

[1] Igor Patrício Michels. *Modelagem*. URL: `https://github.com/IgorMichels/Modelagem`.

[2] Jeff Sackmann. *Tennis ATP*. URL: `https://github.com/JeffSackmann/tennis_atp`.