Relatório Semestral

Gustavo Silva Garone, Eduardio Y. G. Ishihara, Elisabeti Kira

PASSEIO ALEATÓRIO: MODELOS, APLICAÇÕES E SIMULAÇÃO

Introdução

Blaise Pascal propôs no século XVII o problema da Ruína do Jogador que, desde então, tem sido amplamente estudado por diversos pesquisadores. Em nossos estudos, visitamos uma revisão dos principais resultados conhecidos sobre o problema clássico e introduzimos uma nova abordagem. Consideramos um jogo entre dois jogadores, A e B, que começam com a e b reais, respectivamente. Definimos uma variável aleatória discreta X, que determina as regras de ganho e perda de dinheiro dentro do jogo. O jogo termina quando um dos jogadores atinge um saldo de zero reais ou não pode honrar a quantia devida ao oponente.

Neste estudo, propomos um estimador para a esperança da duração total do jogo, considerando dois cenários distintos: um em que X é fixo (caso homogêneo) e outro em que X varia conforme a fortuna atual dos jogadores (caso não homogêneo). Para validar os resultados teóricos obtidos com o estimador, utilizamos algoritmos de simulações de Monte Carlo.

Nesse relatório, apresentamos nosso progresso com o estudo do caso fixo.

Discussão

Inicialmente, realizamos uma revisão da literatura existente para identificar os principais resultados sobre o problema da Ruína do Jogador e suas variações. Encontramos publicações abordando até o quinto momento (centrado e não centrado) da duração do jogo, assim como variações que incluem possibilidade de empate, probabilidades de vitória assimétricas entre os jogadores, aumento no número de participantes e probabilidades de vitória não uniformes ao longo do jogo, um meio não uniforme.

Com base nesses resultados, utilizamos equações de diferenças finitas não homogêneas para reproduzir as formulações teóricas. Paralelamente, desenvolvemos algoritmos de simulação baseados no método de Monte Carlo. A primeira versão do algoritmo, implementada em Python, não obteve resultados satisfatórios em um tempo razoável. Para solucionar essa limitação, migramos para a linguagem Julia, especializada na manipulação de *arrays*, o que

resultou em um aumento significativo na eficiência computacional e, por conseguinte, na qualidade dos dados por nos permitir rodar mais iterações.

Para demonstrar essa melhoria, comparamos o mesmo (ingênuo) algoritmo em Julia e Python*, respectivamente:

```
using StatsBase, LaTeXStrings, Random

function main()
  Random.seed!(1)
  duracoes = []
  M = 1000000
  p = 0.4
  teto = 100
  for i in 1:M
  saldo = 30
    duracao = 0
    while saldo > 0 && saldo < teto
      if p > rand()
        saldo += 1
      else
        saldo -= 1
      end
      duracao += 1
    end
    append!(duracoes, duracao)
  end
  display(LaTeXString("Média da duração de $M passeios:
                       \$ $(round(mean(duracoes), digits = 2))\$"))
  display(LaTeXString("Duração da execução:"))
end

@elapsed main()
```

Média da duração de 1000000 passeios: 150.08

Duração da execução:

```
1.074141064
```

```
using PyCall

py"""
def simulacao_py():
  import random
  import time


  def main():
      random.seed(1)
      duracoes = []
      M = 1000000
      p = 0.4
      teto = 100
      for i in range(M):
          saldo = 30
          duracao = 0
          while saldo > 0 and saldo < teto:
              if p > random.random():
                  saldo += 1
              else:
                  saldo -= 1
              duracao += 1
          duracoes.append(duracao)
      media = sum(duracoes) / len(duracoes)
      print(f"Média da duração de {M} passeios: {media:.2f}")


  start_time = time.time()
  main()
  print(f"Duração da execução: {time.time() - start_time} segundos")
"""
py"simulacao_py"()
```

```
Média da duração de 1000000 passeios: 149.92
Duração da execução: 10.788424015045166 segundos
```

**Nota*: Foi utilizado a biblioteca PyCall que nos permite criar um ambiente Python no Julia - possibilitando rodar código Python nessa linguagem. A performance pode ter sido afetada minimamente nos tempos de chamada ao Kernel de execução do Python.

Desconsiderando o tempo de compilação (aproximadamente 1.2 segundos do tempo total do código em Julia, que só precisa ser feito uma vez), temos que essa linguagem costuma ser consideravelmente mais rápida do que o Python, o que foi crucial nessas simulações.

Nesse teste simples podemos comparar nosso estimador sob estudo com fórmulas conhecidas

da probabilidade e as simulações.

Das simulações, temos a média de duração (que se aproxima da esperança de duração) como 150.

Usando um resultado de probabilidade, temos

$$\mu_a = E(T_a) = \begin{cases} a(N-a) & \textbf{se } p = q = 1/2, \\ \frac{a}{q-p} - \frac{N}{q-p}\left(\frac{1-(q/p)^a}{1-(q/p)^N}\right) & \textbf{se } p \neq q. \end{cases}$$

Onde, nesse caso, a chance de ganhar $p = 0.4 \Rightarrow q = 1 - p$, o início $a = 30$ e final $N = 100$. Da fórmula, temos então que

$$\mu_a = 150$$

O que confere com nossas simulações.

Usando nossa fórmula de aproximação para $q \neq p \neq \frac{1}{2}$,

$$\widehat{E}(T_a) = \begin{cases} \frac{N-a}{p-q} & \textbf{se } p > q \\ \frac{a}{q-p} & \textbf{se } p < q \end{cases}$$

Ou seja, $\frac{30}{0.2} = 150$. O que também condiz com os resultados.

Durante o desenvolvimento dos algoritmos, introduzimos novas regras ao jogo. Nas primeiras simulações, definimos que o jogador A ganhava 1 real com probabilidade $\frac{1}{3}$, ganhava 3 reais com probabilidade $\frac{1}{3}$ e perdia 1 real com probabilidade $\frac{1}{3}$. O novo estimador apresentou um resultado teórico muito próximo ao obtido nas simulações.

Adaptando o estimador acima, podemos trocar o denominador por uma esperança calculada com base nas regras do jogo. Seja X a variável aleatória que contém as regras do jogo:

$$X = \begin{cases} \text{ganho, com probabilidade } p \\ \text{perda, com probabilidade } q = 1 - p \end{cases}$$

Num caso de o jogador ganhar 2 reais com probabilidade $p = 0.3$ e perder 1 real com probabilidade $q = 0.7$, temos a variável de regra X

$$X = \begin{cases} 2, \text{ com probabilidade } 0.3 \\ -1, \text{ com probabilidade } q = 1 - p \end{cases}$$

Com esse recurso, podemos expandir o estimador proposto acima para funcionar com jogos mais complexos:

$$\widehat{E}(T_a) = \begin{cases} \frac{N-a}{E(X)} & \textbf{se } E(X) > 0 \\ \left|\frac{a}{E(X)}\right| & \textbf{se } E(X) < 0 \end{cases}$$

Testamos o estimador sob diferentes regras e, na maioria dos casos, mostrou forte concordância com os valores simulados. Isso destaca a vantagem do estimador: conseguimos estimar a duração de um jogo com regras diversas, o que, até onde encontramos na literatura, não foi encotrado apenas com teoria das probabilidades.

A fraqueza do estimador atualmente reside nas regras que levam a $E(X)$ próximo de 0, uma vez que, conforme $E(X) \to 0$, sua presença no denominador provoca divergências quando comparado com o valor simulado.

Conclusão e Passos Seguintes

Os resultados obtidos indicam que o estimador proposto possui grande flexibilidade para diferentes configurações do jogo e mostra-se especialmente útil, pois as técnicas atuais não são capazes de fornecer um resultado teórico exato para as configurações de jogos propostas. Ademais, o estimador fornece estimativas mais precisas quando há maior circulação de dinheiro entre os jogadores. No entanto, observamos que o estimador tende a divergir rapidamente dos resultados simulados quando a esperança da variável aleatória que rege as regras do jogo aproxima-se de zero devido ao denominador.

Outro desafio identificado foi a complexidade computacional envolvida na simulação de jogos com muitas regras, isto é, quando a variável aleatória pode assumir um grande número de valores. Essa limitação impõe a necessidade de soluções mais eficientes para lidar com cenários de alta complexidade, o que será um obstáculo a ser superado no estudo do caso não homogêneo.

Um interessante caminho de abordagem que buscaremos explorar é a conexão entre passeios aleatórios e redes elétricas como descrito no livro de Snell. Queremos descobrir se existem conexões entre o estimador proposto e fórmulas da física, além de outras abordagens para estudarmos passeios aleatórios, como fizemos com o emprego de equações de diferenças finitas.

Bibliografia

ANDĚL, J.; HUDECOVÁ, Š. Variance of the game duration in the gambler's ruin problem. **Statistics & Probability Letters**, v. 82, n. 9, p. 1750–1754, 31 maio 2012.

EDWARDS, A. W. F. Pascal's Problem: The "Gambler's Ruin". **International Statistical Review / Revue Internationale de Statistique**, v. 51, n. 1, p. 73–79, 1983.

DOYLE, P. G.; J. LAURIE SNELL. Random Walks and Electric Networks. [s.l.] **American Mathematical Soc.**, 1984.