# tarea-1

August 10, 2022

## 1 Tarea 1 Evaluada

- Optimización estocástica
- MA2004B.101
- Profesor: Jaime Eduardo Martínez Sánchez



| Alumno | Matrícula |
|---|---|
| Verónica Victoria García De la Fuente | A00830383 |
| Emily Rebeca Méndez Cruz | A00830768 |
| Daniel de Zamacona Madero | A01570576 |
| Eugenio Santiesteban Zolezzi | A01720932 |
| Juan Pablo Echeagaray González | A00830646 |



- 11 de agosto del 2022

[1]:
```
# Dependencias básicas
import random
import matplotlib.pyplot as plt
import numpy as np

# Estilo de gráfica y semilla para reproducibilidad
plt.style.use('ggplot')
random.seed(7501)

# Parámetros generales
max_iter = 1e3
```

### 1.1 El Problema de Monty Hall

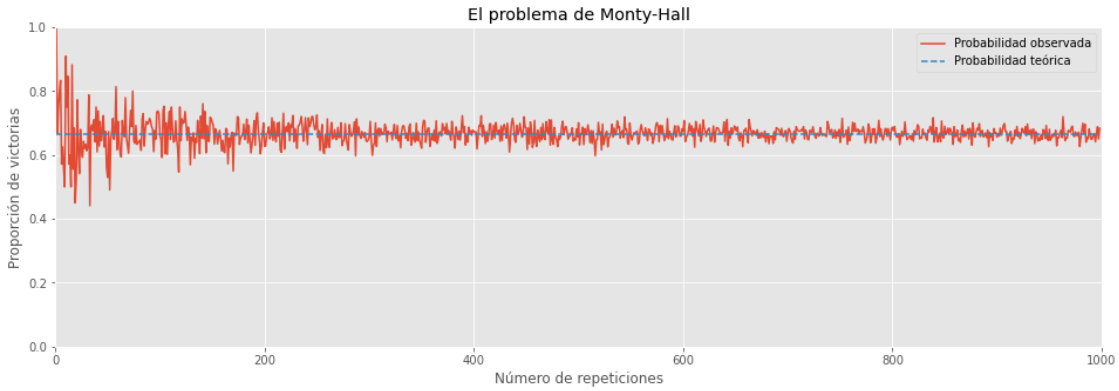
En un concurso se tienen tres puertas, detrás de una de ellas hay un auto y detrás de las otras hay una cabra. El participante debe elegir una de las tres puertas, sin abrirla. Después Monty, el presentador, abre una de las dos puertas restantes en la que hay una cabra. Así quedan dos puertas sin abrir una con auto y otra con cabra. Monty ofrece la posibilidad al concursante de cambiar su puerta o permanecer con su elección. ¿Qué es mejor, cambiar de puerta o no? (Este problema es uno de los más controversiales en probabilidad y se basa en un programa de televisión de los 70's).

[2]:
```
def problema_1():

```

```
    n_doors = 3

    def monty_hall(bin_switch, n_doors, num_tests):

        def get_non_prize_door(host: int, num_doors: int, player_choice: int):
            i = 1
            if (host > num_doors) or (player_choice > num_doors):
                raise ValueError("There aren't enough doors")
            while (i == host or i == player_choice):
                i = (i + 1) % num_doors
            return i


        def switch_function(shown_door, num_doors, player_choice):
            i = 1
            while (i == shown_door) or (i == player_choice):
                i = (i + 1) % num_doors

            return i

        win_switch_cnt = 0
        win_no_switch_cnt = 0
        lose_switch_cnt = 0
        lose_no_switch_cnt = 0
        # doors = [i for i in range(n_doors)]

        for i in range(int(num_tests)):
            door_with_prize = random.randint(0, n_doors - 1)
            host = door_with_prize
            player_choice = random.randint(0, n_doors - 1)
            shown_door = get_non_prize_door(host, n_doors, player_choice)

            if bin_switch:
                player_choice = switch_function(shown_door, n_doors,␣
↪player_choice)

            if player_choice == host and bin_switch == False:
                # print("Player won with no switch")
                win_no_switch_cnt += 1
            elif player_choice == host and bin_switch == True:
                # print("Player won with switch")
                win_switch_cnt += 1
            elif player_choice != host and bin_switch == False:
                # print("Player lost with no switch")
                lose_no_switch_cnt += 1
            elif player_choice != host and bin_switch == True:
                # print("Player lost with switch")
```

```
                lose_switch_cnt += 1
            else:
                print('Something went wrong')

        return win_switch_cnt, win_no_switch_cnt, lose_switch_cnt,␣
↪lose_no_switch_cnt

    win_prop = []

    for i in range(1, int(max_iter)):
        y = monty_hall(True, n_doors, i)
        win_prop.append(y[0] / i)

    plt.figure(figsize=(16, 5))
    plt.plot(np.arange(max_iter - 1), win_prop, '-', label='Probabilidad␣
↪observada')
    plt.plot(np.arange(max_iter - 1), np.ones(int(max_iter) - 1) * (2/3), '--',␣
↪label='Probabilidad teórica')
    plt.xlim(0, max_iter)
    plt.ylim(0, 1)
    plt.xlabel('Número de repeticiones')
    plt.ylabel('Proporción de victorias')
    plt.legend()
    plt.title('El problema de Monty-Hall');

    print(f'Proporción de victorias con el cambio de puerta: {win_prop[-1]:.
↪2f}')

problema_1()
```

```
Proporción de victorias con el cambio de puerta: 0.68
```

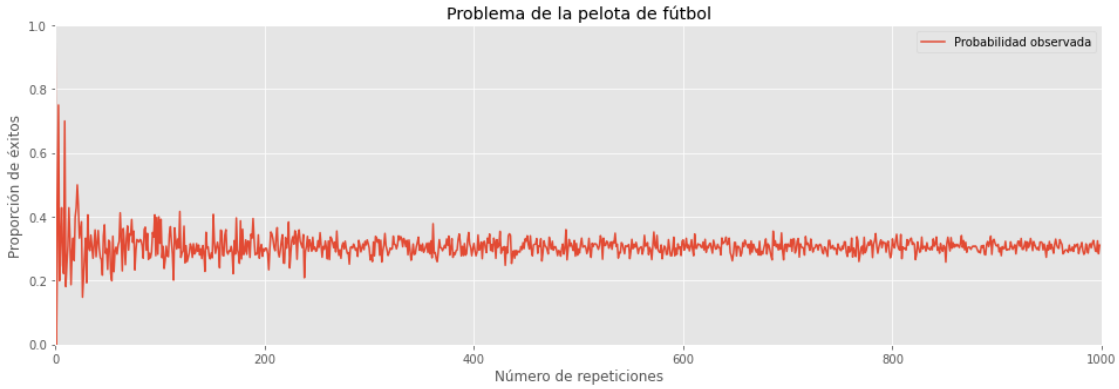


## 1.2 La bola de futbol

En un refresco que compró Juan en la pulpería la MINITA, cercana a su colegio, se ganó una bola de fútbol. Sin embargo, al reclamar su premio en la MINITA, la encargada le indicó que el premio solamente se lo puede dar el camión repartidor y únicamente pasa el martes entre 10am y 11am aleatoriamente, y en la pulpería se queda exactamente 10 minutos. Dado que Juan está en clases ese día, decide elegir al azar un tiempo entre 10:00 am y 11:00 am para fugarse de clases y esperar en la pulpería exactamente diez minutos para ver si logra encontrarse con el camión repartidor. ¿Cuál es la probabilidad de que el martes obtenga su premio?

[3]:
```
def problema_2():

    def futbol_ball(n_hours, wait_time, n_simulations):

        results = []

        for i in range(n_simulations):
            juan_arrival = n_hours * random.random() * 60
            truck_arrival = n_hours * random.random() * 60

            if abs(juan_arrival - truck_arrival) <= wait_time:
                res = 1
            else:
                res = 0

            results.append(res)

        return results

    results = []
    for i in range(1, int(max_iter)):
        res = futbol_ball(1, 10, i)
        win_prop = sum(res) / i
        results.append(win_prop)

    plt.figure(figsize=(16, 5))
    plt.plot(np.arange(max_iter - 1), results, '-', label='Probabilidad␣
 ↪observada')
    plt.xlim(0, max_iter)
    plt.ylim(0, 1)
    plt.xlabel('Número de repeticiones')
    plt.ylabel('Proporción de éxitos')
    plt.legend()
    plt.title('Problema de la pelota de fútbol');

    print(f'Probabilidad relativa: {results[-1]:2f}')

problema_2()
```

```
Probabilidad relativa: 0.311311
```




### 1.3 La Ley de los grandes números

Explore con Excel la siguiente proposición: De acuerdo con la Ley de los Grandes Números, entre más veces se tira una moneda, más cerca se estará el número obtenido de escudos de la mitad del total de los lanzamientos.

La proposición del ejercicio anterior es falsa. La Ley de los grandes Números se refiere a valores relativos no absolutos. Dicha afirmación es una mal-interpretación frecuente de la Ley de los grandes números y se desmiente en el siguiente problema.

[4]:
```
def problema_3():

    def toss_coin(n_simulations):
        results = []

        for i in range(n_simulations):
            res = random.randint(0, 1)
            results.append(res)

        return results

    # Supondremos que un valor de 1 se asigna a "Escudos"
    results = []
    for i in range(1, int(max_iter)):
        res = toss_coin(i)
        win_prop = sum(res) / i
        results.append(win_prop)

    plt.figure(figsize=(16, 5))
    plt.plot(np.arange(max_iter - 1), results, '-', label='Probabilidad␣
↪observada')
```

```
    plt.plot(np.arange(max_iter - 1), np.ones(int(max_iter) - 1) * (1/2), '--',␣
 ↪label='Probabilidad teórica')
    plt.xlim(0, max_iter)
    plt.ylim(0, 1)
    plt.xlabel('Número de repeticiones')
    plt.ylabel('Proporción de escudos')
    plt.legend()
    plt.title('Tiro de una moneda');

    print(f'Probabilidad relativa: {results[-1]:2f}')

problema_3()
```

```
Probabilidad relativa: 0.515516
```


## 1.4 El falso determinismo

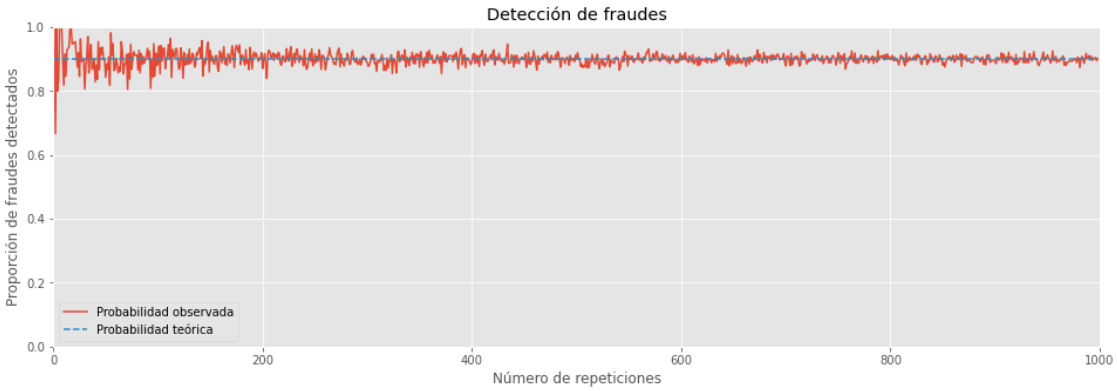
Un software asegura que detecta el 90% de los fraudes bancarios que ocurren en las tarjetas. Ante esto el Banco de Los Sueños decide adquirir el software para detectar los fraudes que les ocurren a sus clientes en las tarjetas. Sin embargo, en el primer momento de uso, el software no detectó un fraude.

El banco decide demandar a la empresa, pero al revisar el software, resulta que los cálculos están bien hechos. ¿Qué está sucediendo entonces?

[5]:
```
def problema_4():
    prob = 0.9
    np.random.seed(7501)

    results = []
    for i in range(1, int(max_iter)):
        test = np.random.choice([0, 1], size=i, p=[1-prob, prob])
        win_prop = sum(test) / i
        results.append(win_prop)
```

```
    plt.figure(figsize=(16, 5))
    plt.plot(np.arange(max_iter - 1), results, '-', label='Probabilidad␣
 ↪observada')
    plt.plot(np.arange(max_iter - 1), np.ones(int(max_iter) - 1) * prob, '--',␣
 ↪label='Probabilidad teórica')
    plt.xlim(0, max_iter)
    plt.ylim(0, 1)
    plt.xlabel('Número de repeticiones')
    plt.ylabel('Proporción de fraudes detectados')
    plt.legend()
    plt.title('Detección de fraudes');

    print(f'Probabilidad relativa: {results[-1]:2f}')

problema_4()
```

```
Probabilidad relativa: 0.901902
```




## 1.5 Resumen: Probabilidad de ruina en el modelo clásico de Cramer-Lundberg

Jiménez Hernández, J. D. C., & Maldonado Santiago, A. D. (2011). Probabilidad de ruina en el modelo clásico de Cramer-Lundberg. REPOSITORIO NACIONAL CONACYT.