Proyecto de Simulacio de la Escalada Deportiva

Alejandro Lamelas Delgado

Mauro Eduardo Campver Barrios

¿Qué es la Escalada Deportiva?

La escalada deportiva es una modalidad de escalada que se practica en paredes de roca natural o en muros artificiales. En esta modalidad, los escaladores ascienden rutas predefinidas utilizando anclajes fijos distribuidos a lo largo de la ruta. Estos anclajes, que incluyen chapas y parabolts, están diseñados para proteger al escalador en caso de caída. La escalada deportiva se caracteriza por centrarse en la dificultad técnica y física de la escalada, con un enfoque en la fuerza, la flexibilidad, la técnica y la resistencia del escalador.

Modalidades

- Boulder:

En Boulder, los atletas escalan muros de 4,5m de altura sin cuerdas en un tiempo limitado y en el menor número de intentos posible, las puntuaciones constan de 4 digitos, el primero es la cantidad de rutas o problemas completados por el atleta(topes), el segundo la cantidad de zonas que alcanzó, que son el punto medio de un problema, y los últimos 2 digitos son los los intentos de alcanzar topes y zonas respectivamente.

- Leader:

En la disciplina de Lead, los atletas escalan lo más alto que pueden una pared de más de 15m de altura en seis minutos sin haber visto la ruta con antelación. Las rutas para este evento son cada vez más complejas y desafiantes a lo largo de la prueba, requiriendo todas las habilidades físicas y mentales de los atletas.

- Speed:

La modalidad Speed es una espectacular carrera contrarreloj en rondas eliminatorias de uno contra uno que combinan precisión y explosividad, los puntajes estan dados por el tiempo.

Objetivos del Proyecto

En este proyecto nos propusimos simular las modalidades ***Speed*** y ***Boulder*** para intentar predecir los 8 primeros puestos en dichas disciplinas en las olimpiadas 2024, utilizando para esto los datos oficiales de las puntuaciones registradas en la página https://ifsc-

climbing.org/ de los eventos ocurridos desde 2022 hasta la fecha, obtenidos a través de un exhaustivo scrapeo mayormente hecho a mano puesto que el funcionamiento de la página deja mucho que desear.

Lectura de Datos

Nuestros datos fueron extraidos en formato json, todos se encuentran en la carpeta data y los importamos ya procesados y separados por sexo en la variable `athletes_points` para centrarnos en los detalles de la simulación. Cada atleta posee un array de puntos por cada modalidad, estos puntos los utilizaremos para hallar una funcion de densidad.

In [ ]:
```
import sys
from dataReading import athletes_points
```

Función de Densidad

Una vez con los datos a nuestra disposición nos damos cuenta que no conocemos la distribución que tienen las puntuaciones de los atletas en las distintas disciplinas. Para solucionar esto utilizamos la estimacion de la densidad de Kernel (KDE).

KDE: Es la aplicación del suavizado de kernel para la estimación de la densidad de probabilidad, es decir, un método no paramétrico para estimar la función de densidad de probabilidad de una variable aleatoria basada en kernels como pesos. KDE responde a un problema fundamental de suavizado de datos en el que se hacen inferencias sobre la población, basándose en una muestra finita de datos.

Nos auxiliamos de la biblioteca `sklearn`

Al trabajar con KDE hay que definir 2 elementos clave, la función que se utilizará como kernel y el ancho de banda (bandwidth).

El ancho de banda lo escogimos tanteando valores que nos parecion sensatos en el contexto y como función de Kernel escogimos Tophat que nos arrojó buenos resultados.

In [ ]:
```
import pandas as pd
import calendar, datetime
import math
import numpy as np
from sklearn.neighbors import KernelDensity
from sklearn.model_selection import GridSearchCV



def CalcKde(points,discipline):

    best_bandwidth = 1
    if(discipline=='boulder'):
        best_bandwidth=2

    return KernelDensity(kernel="tophat", bandwidth=best_bandwidth).fit(points)
```

Relevancia de las puntuaciones

Al hallar cada Funcion de densidad modificamos los datos, dandole mayor relevancia a los atletas con mayor cantidad de puntuaciones registradas, esto lo conseguimos multiplicandole a cada puntuación de un atleta en una disciplina un $Pval = (1 - \frac{\alpha}{cantPtos})$. Los α los escogimos tanteando y buscando quitarnos atletas poco confiables que tenian pocas marcas y en nuestras simulaciones aparecian alto en el ranking.

In [ ]:
```
##Funcion que modificara los puntos de los atletas en funcion de la cantidad de

def modify_scores_based_on_cuantity(points,discipline):

    ##Alpha -5 para el Pval de las 2 primeras componentes y 5 para las 2 ultimas
    if discipline=='boulder':
        alpha=5
        pondVal_first_two_comp=(1-alpha/len(points))

        pondVal_last_two_comp=(1+alpha/len(points))
        for vector in points:
            for i in range(2):
                vector[i]= max(vector[i]*pondVal_first_two_comp,0)
                vector[i+2]= vector[i+2]*pondVal_last_two_comp

    ##Alpha 4 para la categoria speed
    else:
        alpha=4
        pondVal=(1+alpha/len(points))
        for time in points:
            time=time*pondVal

##Funcion que devuelve el KDE de los atletas en cada disciplina en caso de que h
##mas de una puntuación registrada
def KDE_for_athlete(athletes_points):
    athletes_kde={}
    for athlete in athletes_points:
        if athlete not in athletes_kde:
            athletes_kde[athlete]={}
        for discipline in athletes_points[athlete]:
            if(len(athletes_points[athlete][discipline])>1 and discipline != 'le

                ##Se modifican los puntos
                modify_scores_based_on_cuantity(athletes_points[athlete][discipl

                if discipline == 'speed':
                    athletes_kde[athlete][discipline]=CalcKde(np.array(athletes_
                    continue;
                athletes_kde[athlete][discipline]=CalcKde(athletes_points[athlet
    return athletes_kde
```

Simulación

Para simular el evento (calificacion y final) usamos las funciones de probabilad de densidad que ya mostramos, esto lo hacemos 1000 veces y nuestro top 8 propuesto como predicción lo formamos basandonos en las posiciones obtenidas (se ordenan por la cantidad de primeros lugares obtenidos, si dos tienen la misma cantidad pues se miran los segundos lugares y asi sucesivamente). Para mayor comodidad esto último lo implementamos por medio de un heap de maximo donde la prioridad esta definida por los rankings como explicamos anteriormente.

In [ ]:
```
from heapImplementation import MaxHeap
from functools import cmp_to_key
from EventsSimulation import simulate_event

max_heap_boulder_men = MaxHeap()

max_heap_boulder_women = MaxHeap()

max_heap_speed_men = MaxHeap()

max_heap_speed_women = MaxHeap()

men_athletes_kde = KDE_for_athlete(athletes_points['men'])

women_athletes_kde = KDE_for_athlete(athletes_points['women'])


def compare_vectors(player1, player2):
    """
    Compares two vectors of length 4 based on the specified rules:
    *Boulder Criteria

    1. Compare first components.
    2. If equal, compare second components.
    3. If still equal, compare third components.
    4. If still equal, compare fourth components.

    Args:
        vec1 (np.ndarray): First vector of length 4.
        vec2 (np.ndarray): Second vector of length 4.

    Returns:
        int: -1 if vec1 is smaller, 1 if vec1 is larger, 0 if equal.
    """
    vec1=player1['score'];
    vec2=player2['score'];

    for i in range(2):
        if vec1[i] < vec2[i]:
            return 1
        elif vec1[i] > vec2[i]:
            return -1
    for i in range(2):
        if vec1[i+2] < vec2[i+2]:
            return -1
        elif vec1[i+2] > vec2[i+2]:
            return 1
    return 0
```

```
key_function = cmp_to_key(compare_vectors)

for i in range(0,1000):
    simulate_event(men_athletes_kde,max_heap_boulder_men,'boulder',key_function)
    simulate_event(women_athletes_kde,max_heap_boulder_women,'boulder',key_funct

    simulate_event(men_athletes_kde,max_heap_speed_men,'speed',lambda x: x['scor
    simulate_event( women_athletes_kde, max_heap_speed_women,'speed',lambda x: x
```

Boulder Hombres

In [ ]:
```
#Top 8 de boulder masculino
for i in range(0,8):
    athlete= max_heap_boulder_men.pop()
    #                   Cantidad de veces en primer lugar
    print(athlete.name,athlete.ranks[0])
```

```
kokoro fujii 174
mejdi schalck 100
yoshiyuki ogata 89
tomoa narasaki 60
yannick flohé 48
antoine girard 45
nicolai uznik 41
sam avezou 41
```

Boulder Mujeres

In [ ]:
```
#Top 8 de boulder femenino
for i in range(0,8):
    athlete= max_heap_boulder_women.pop()
    #                   Cantidad de veces en primer lugar
    print( athlete.name, athlete.ranks[0])
```

```
iziar martínez almendros 90
camilla moroni 66
lisa klem 58
natalia grossman 56
janja garnbret 55
chloe caulier 40
oriane bertone 37
giulia medici 35
```

Speed Hombres

In [ ]:
```
#Top 8 de speed masculino
for i in range(0,8):
    athlete= max_heap_speed_men.pop()
    #                   Cantidad de veces en primer lugar
    print(athlete.name,athlete.ranks[0])
```

```
peng wu 66
xinshang wang 59
jinbao long 50
jianguo long 49
kiromal katibin 47
zainal aripin 42
veddriq leonardo 41
aditya tri syahria 37
```

Speed Mujeres

In [ ]:
```
#Top 8 de speed femenino
for i in range(0,8):
    athlete= max_heap_speed_women.pop()
    #                   Cantidad de veces en primer lugar
    print(athlete.name,athlete.ranks[0])
```

```
aleksandra miroslaw 263
lijuan deng 126
rajiah sallsabillah 76
desak made rita kusuma dewi 75
aleksandra kalucka 68
emma hunt 58
narda mutia amanda 50
di niu 44
```

Posibles Mejoras

Propondre un par de ideas que con un poco más de tiempo y un mejor dataset podrian mejorar la precisión de las predicciones

-Prioridad a Fechas Recientes:

Hacer que las puntuaciones a medida que son mas cercanas a la actualidad aparezcan mas veces, asi la función de densidad que construya el KDE hará mas probable las puntuaciones mas recientes. (Los datos de las fechas no los scrapeamos)

-Funcion de Kde por Fase:

Crear una Función de densidada para cada Fase de las competencia (Calificación,Semifinal,Final) basandose en los datos registrados del atleta en la respectiva fase aumentaría considerablemente la precisión de la simulacion teniendo en cuenta que las fases mas avanzadas suelen ser mas complejas y las puntuaciones suelen ser más bajas, esto haría que la simulación del performance de un atleta se parezca muchísimo más a la realidad. (Separar los datos por fase no iba a ser tan sencillo y surge también el siguiente problema: ¿qué función de densidad le asigno a un atleta en una fase en la que nunca ha clasificado?)