Universidad de La Laguna

Sección de Matemáticas | Facultad de Ciencias

Zebensuí Hernández Díaz

Modelos probabilísticos para el estudio de la lucha canaria

Estimación de un ranking

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Julio de 2017

Dirigido por

Carlos González Alcón

Carlos González Alcón
Departamento de Matemática, Estadística e Investigación Operativa
Universidad de La Laguna
38271 La Laguna, Tenerife

Agradecimientos

A las primeras personas que se lo quiero agradecer son a mis padres, por estar ahí proporcionando la mejor educación y aprendizajes de la vida.

En especial a mi madre, por enseñarme que con esfuerzo, trabajo y constancia todo se consigue, y que en esta vida nadie te regala nada.

En especial a mi padre, por cada día hacer la vida más diferente, por ser un amigo más y confiar en mis decisiones.

Agradecer también a mí tutor Carlos González Alcón, que sin su ayuda y conocimientos no hubiese sido posible la realización de este trabajo.

A mis compañeros de clase, con los que tengo bonitos recuerdos y grandes momentos. Aquí en especial guardo un espacio para mí querido compañero y amigo Alberto García Díaz, donde he debatido razonamientos de mi trabajo en momentos de bloqueo y además hemos sido grandes compañeros de camino durante la carrera y fuera de la Facultad.

Resumen · Abstract

Resumen

Este trabajo trata de modelizar y resolver el problema de estimar un ranking de luchadores en el deporte de la Lucha Canaria con el objetivo de (1) clasificar a los luchadores por categorías y (2) servir de ayuda al entrenador para organizar los luchadores de su equipo. Para ello se modelizó el enfrentamiento entre dos luchadores como una cadena de Markov. Se diseñaron algoritmos iterativos para asignar una valoración a cada luchador ajustado a datos de los encuentros.

Palabras clave: *Cadena de Markov, algoritmo PageRank, matriz de transición, estado transitorio, estado absorbente, probabilidad de transición, lucha canaria, modelo probabilístico, función del modelo matemático, distribución trinomial, método iterativo, vector de valoraciones.*

Abstract

This paper tries to model and solve the problem of making an estimation in order to rank the wrestlers in the sport called "Lucha Canaria" with the objective of (1) classifying wrestlers by categories, and (2) to help the coach to organize the wrestlers of his team. For this, the confrontation between two wrestlers was modeled as a Markov chain. Iterative algorithms were designed to assign a valoration to each wrestler adjusted to match data.

Keywords: *Markov chain, PageRank algorithm, transition matrix, transient state, absorbing state, transition probability, Canarian wrestling, probabilistic model, function of the mathematical model, trinomial distribution, iterative method, vector assessments.*

Contenido

Agradecimientos III

Resumen/Abstract V

Introducción IX

1. Motivación y objetivos 1
1.1. Motivación 1
1.2. Objetivos 2

2. Fundamentos teóricos 3
2.1. Matriz de transición 4
2.2. Probabilidad de llegar a un estado en n pasos 4
2.3. Probabilidad de transición en n pasos 5
2.4. Cadena homogénea absorbente y forma canónica 6

3. Descripción del problema 7

4. Enfrentamiento de dos luchadores 9
4.1. Construcción de la cadena de Markov 9
4.2. Funciones de probabilidad en un enfrentamiento 11

5. Modelo para la estimación de un ranking 13

6. Resolución del modelo del problema principal 17
6.1. Primer Algoritmo 18
6.2. Segundo Algoritmo 21
6.3. Mejoras del algoritmo respecto a uso de los datos 22

7. Conclusiones 23

8. Apéndice I 25
8.1. Algoritmo primero 25
8.2. Algoritmo segundo 26
8.3. Funciones usadas para los dos algoritmos 27
8.4. Simulación de datos respecto con el tiempo 32

9. Apéndice II 37

Bibliografía 39

Poster 41

Introducción

En este trabajo se trata de construir modelos probabilísticos en el ámbito del deporte de la lucha Canaria. A continuación vamos a dar una breve explicación sobre la estructura del documento.

En el *capítulo* 1 se presenta la motivación y objetivos del trabajo, además se comenta el uso diverso de las *cadenas de Markov* en muchas áreas de las ciencias. Después en el *capítulo* 2 profundizamos en las herramientas matemáticas que se emplearán en este trabajo. En el siguiente *capítulo* 3 realizamos una descripción detallada del problema real que se va abordar. Se trata de construir un algoritmo que a partir de una base de datos nos permita estimar un ranking de luchadores. Pero antes necesitamos estudiar el enfrentamiento de dos luchadores cualesquiera y eso se verá en el *capítulo* 4, donde primero fijamos el sistema de lucha y establecemos los distintos sucesos mediante un esquema gráfico, luego a partir de dicho esquema podemos aglutinar todas las probabilidades de cambio de situaciones o estados en una matriz.

A continuación en el *capítulo* 5, ya podemos adentrarnos en el problema principal que es la construcción de un algoritmo para estimar un ranking, donde se necesita un registro de datos, valoraciones iniciales, probabilidades por agarrada y probabilidades de cada enfrentamiento (que dependen de las probabilidades por agarrada).

A partir de aquí de forma natural nos podemos preguntar cuánto es de probable que se dé el registro si tenemos unas determinadas probabilidades de enfrentamiento. Con esto llegamos a construir una función del modelo probabilístico de estimación de un ranking, donde la función se encarga de medir cuánto de verosímil son las probabilidades de que se de cierto registro de núme-

ros de enfrentamientos.

Ahora en el *capítulo* 6 se busca un método iterativo que resuelva el modelo probabilísitico de estimación de un ranking, ya que resolverlo como un problema de análisis matemático es muy complicado por la complejidad respecto a la cantidad de variables.

En el *capítulo* 7 se comenta dos posibles ampliaciones de este trabajo y se dan conclusiones del uso de esté estudio realizado. Al final de este documento, como anexos se ha puesto todos los programas que se han implementado y las pautas de una simulación para generar los registros de datos.

Recursos utilizados:

- *Geogebra 5.0* para construir los dibujos o representaciones gráficas.
- *Gimp 2.9* para manipular algunas imagenes.
- *Python 2.7* para implementar funciones, gráficas y testear algoritmos construidos que resuelve los modelos.
- *Latex* para redactar la investigación que se ha realizado.

1

Motivación y objetivos

1.1. Motivación

La decisión de haber elegido esta área de las matemáticas para realizar el trabajo fin de grado surgió durante la carrera cuando cursé las asignaturas de *Probabilidades* I en donde se resolvían problemas de procesos de ensayos independientes y en *Probabilidades* II se trabajó con algunos modelos estocásticos donde se vio la dependencia de experimentos anteriores en las predicciones. La herramienta matemática que resuelve esta cuestión se llama *cadenas de Markov* y está atribuida a *Andrei Andreyevich Markov* que la publicó en el año 1907.

Por otra parte, me llamo la atención cómo en la actualidad las cadenas de Markov se considera una herramienta esencial para modelizar problemas reales en disciplinas como en:

- **Física:** Se usa en muchos problemas de termodinámica y la física estadística. Ejemplos importantes se pueden encontrar en la *cadena de Ehrenfest* o *el modelo de difusión de Laplace*.
- **Meteorología:** Si consideramos el clima de una región a través de distintos días, es claro que el estado actual solo depende del último estado y no de toda la historia, de modo que se pueden usar cadenas para formular *modelos climatológicos básicos*.
- **Internet:** El *pagerank* de una página web (usado por Google en sus motores de búsqueda) se define a través de una cadena, donde la posición que tendrá una página en el buscador será determinada por su peso en la distribución estacionaria de la cadena.
- **Juegos de azar:** Por ejemplo el *modelo de la ruina del jugador*, que establece la probabilidad de que una persona apuesta en un juego de azar finalmente termine sin dinero.
- **Economía y Finanzas:** Se utiliza en *modelos simples de valuación de opciones* para determinar cuándo existe oportunidad de arbitraje, así como en

el *modelo de colapsos de una bolsa de valores* o para determinar la *volatilidad de precios*. En los negocios se han utilizado para analizar los patrones de compra de los deudores morosos, para planear las necesidades de personal y para analizar el reemplazo de equipo.

1.2. Objetivos

Los objetivos planteados de este trabajo sobre *Modelos probabilísticos para el estudio de la Lucha Canaria* son:

- Primero construir un modelo que represente el problema sobre el enfrentamiento de dos luchadores cualesquiera y de ahí sacar las probabilidades de enfrentamiento en función de las probabilidades por agarrada.
- En segundo lugar modelizar el problema de estimación de un ranking y buscar la forma de resolverlo, donde en el proceso de construcción del modelo se hará uso del problema anterior modelizado.

La finalidades de este trabajo son:

- Ayudar a la Federación en clasificar a los luchadores de cada categoría.
- Mejorar las tomas de decisiones del *mandador*[*] para organizar sus luchadores teniendo en cuenta no solo las reglas de enfrentamientos por equipos, sino también la habilidad de los luchadores.

[*] *Mandador*: Preparador, entrenador de un equipo de lucha canaria. Casi todos los mandadores han sido luchadores en su tiempo.

2

Fundamentos teóricos

Un **proceso estocástico** es una familia arbitraria de variables aleatorias $\{X_t \in S :\ t \in T\}$, donde:

- S es el espacio muestral, el conjunto de valores posibles que puede tomar $\{X_t\}$.
- t es el índice que hace referencia a la posición dentro de la secuencia, y toma valores en T.

Es usual interpretar t como el tiempo transcurrido desde el *instante inicial* ($t = 0$) y que en cada instante $t \in T$ hay un experimento donde el resultado queda determinado por X_t.

Principalmente, interesa saber con qué probabilidad cada X_t asume valores en ciertos subconjuntos de S, y también si esta probabilidad está influenciada de alguna manera por los valores observados en los instantes de tiempo anteriores (la historia del proceso). Por otro lado interesa aquí el caso en que $T = \mathbb{N} = \{0, 1, \dots\}$ y S es un conjunto a lo sumo numerable.

Definición 2.0.1. Sea S un conjunto a lo sumo numerable (de *estados*) y $\{X_n\}_{n\in\mathbb{N}}$ un proceso estocástico a valores en S. Se dice que el proceso satisface la **propiedad markoviana** si para todo entero $n \geqslant 0$ y $s_j, s_{i_n}, \dots, s_{i_0} \in S$, se cumple que:

$$P\left(X_{n+1} = s_j | X_n = s_{i_n}, \dots, X_0 = s_{i_0}\right) = P\left(X_{n+1} = s_j | X_n = s_{i_n}\right). \qquad (2.1)$$

Una **cadena de Markov** es un proceso estocástico que satisface la condición markoviana. Es decir, la propiedad establece que la probabilidad de que en el instante $n + 1$ la cadena pase al estado s_j sólo depende del estado s_{i_n} en que se encuentra en el instante n, y no de la historia completa $s_{i_0}, \dots, s_{i_n}$.

Sea $S = \{s_1, \dots, s_r\}$ el conjunto de estados. Si el proceso comienza en uno de los estados s_i con una probabilidad inicial $p_i^{(0)}$, (es decir, es la probabilidad de

que el sistema ocupe inicialmente el estado s_i), entonces la distribución inicial es $p^{(0)} = \left(p_1^{(0)}, \ldots, p_r^{(0)}\right)$.

2.1. Matriz de transición

En general la probabilidad $P\left(X_{n+1} = s_j | X_n = s_i\right)$ podría depender no sólo de los estados s_i y de s_j, sino también de n. Aquellas cadenas en las que depende sólo de s_i y de s_j constituyen una importante clase.

Definición 2.1.1. Una cadena se dice **homogénea** si para todo $n \geqslant 0$ y $s_i, s_j \in S$, es $P\left(X_{n+1} = s_j | \ X_n = s_i\right) = P\left(X_1 = s_j | X_0 = s_i\right)$. En este caso, el número $P\left(X_{n+1} = s_j | X_n = s_i\right)$ es denotado mediante p_{ij}, y se denomina **probabilidad de transición**.

Si $S = \{s_1, \ldots, s_r\}$ es el conjunto de estados, entonces todos los valores p_{ij} se combinan formando la **matriz de transición** M de tamaño $r \times r$, así:

$$M = [p_{ij}] = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ p_{r1} & p_{r2} & \cdots & p_{rr} \end{pmatrix}.$$

2.2. Probabilidad de llegar a un estado en n pasos

La probabilidad de llegar a s_j después de n pasos se denota con $p_j^{(n)}$. Es decir, $P\left(X_n = s_j\right) = p_j^{(n)}$. Para calcular $p_j^{(1)}$ aplico el teorema de la probabilidad total:

$$p_j^{(1)} = \sum_{i=1}^{r} p_i^{(0)} p_{ij}.$$

Esto se puede expresar de forma vectorial. Sean los vectores fila de probabilidad $p^{(0)} = (p_1^{(0)}, \ldots, p_r^{(0)})$ y $p^{(1)} = (p_1^{(1)}, \ldots, p_r^{(1)})$, donde $p^{(0)}$ es la distribución inicial y $p^{(1)}$ es la probabilidad de que se alcance uno de los estados $s_1, \ldots, s_r$ después de un paso. Con esta notación, se puede expresar:

$$p^{(1)} = [p_j^{(1)}] = \left[\sum_{i=1}^{r} p_i^{(0)} p_{ij}\right] = p^{(0)} M.$$

Del mismo modo,

$$p^{(2)} = p^{(1)} M = p^{(0)} M^2,$$

y en n pasos,

$$p^{(n)} = p^{(n-1)}M = p^{(0)}M^n,$$

donde $p^{(n)} = \left(p_1^{(n)}, \ldots, p_r^{(n)}\right)$.

Al igual que $p^{(0)}$ es la distribución de probabilidad inicial, $p^{(n)}$ es la distribución de probabilidad en el instante n.

2.3. Probabilidad de transición en n pasos

Dados $i, j \in S$ se define $p_{ij}^{(n)}$ como la probabilidad de que la cadena esté en el estado j después de n pasos, dado que la cadena empezó en el estado i. Es decir:

$$p_{ij}^{(n)} = P\left(X_n = j | X_0 = i\right).$$

Por otro lado como la cadena debe haber pasado por uno de los r posibles estados en la etapa $n-1$, se tiene que:

$$p_{ij}^{(n)} = \sum_{k=1}^{r} P\left(X_n = j, X_{n-1} = k | X_0 = i\right), \qquad \forall n \geq 2.$$

Observación 2.3.1. Se tiene la siguiente igualdad, para tres posibles sucesos A, B y C:

$$P\left(A \cap B \mid C\right) = P\left(A \mid B \cap C\right) P\left(B \mid C\right). \tag{2.2}$$

Se sustituye:

$$A \to (X_n = j), \quad B \to (X_{n-1} = k) \quad \text{y} \quad C \to (X_0 = i),$$

entonces:

$$\begin{aligned}
p_{ij}^{(n)} &= \sum_{k=1}^{r} P\left(X_n = j, X_{n-1} = k \mid X_0 = i\right) \\
&\underset{Obs:(2.3.1)}{=} \sum_{k=1}^{r} P\left(X_n = j \mid X_{n-1} = k, X_0 = i\right) P\left(X_{n-1} = k \mid X_0 = i\right) \\
&\underset{Def:(2.0.1)}{=} \sum_{k=1}^{r} P\left(X_n = j \mid X_{n-1} = k\right) P\left(X_{n-1} = k \mid X_0 = i\right) \\
&= \sum_{k=1}^{r} p_{kj}^{(1)} p_{ik}^{(n-1)} = \sum_{k=1}^{r} p_{ik}^{(n-1)} p_{kj}^{(1)},
\end{aligned}$$

usando la *propiedad markoviana* 2.0.1. La expresión anterior se denomina **Ecuaciones de Chapman-Kolmogorov**.

Haciendo n igual a $2, 3, \ldots$ se obtiene que las matrices con esos elementos son:

$$\left[p_{ij}^{(2)}\right] = \left[p_{ik}^{(1)} p_{kj}^{(1)}\right] = M^2,$$
$$\left[p_{ij}^{(3)}\right] = \left[p_{ik}^{(2)} p_{kj}^{(1)}\right] = M^3,$$

ya que $p_{ik}^{(2)}$ son los elementos de M^2 y sucesivamente se tiene que $\left[p_{ij}^{(n)}\right] = M^n$.

2.4. Cadena homogénea absorbente y forma canónica

En el modelo probabilístico que vamos a trabajar se distinguen dos tipos de estados:

- **Estado absorbente**: Es aquel que una vez que se entra en él no se puede salir del mismo.
- **Estado transitorio**: Es aquel estado en que se puede pasar a otro.

Las cadenas que no dependen del tiempo y tienen estados transitorios y absorbentes se denominan **cadenas homogéneas absorbentes**.

3

Descripción del problema

La *Lucha Canaria* es un deporte de las Islas Canarias (España). Se caracteriza por la habilidad para aprovechar la fuerza del contrario y no herirlo. Para quebrar la estabilidad del rival no se permite la lucha en el suelo, como ocurre con otras modalidades, ni ninguna clase de llaves.

Este es un deporte de oposición que se fundamenta en el enfrentamiento de dos adversarios, los cuales desde una posición inicial de bipedestación y agarre a la boca-manga del pantalón, procuran, durante un tiempo no superior al minuto y medio, sin salirse de un terreno circular, desequilibrar para hacer tocar al contrario con cualquier parte del cuerpo que no sea la planta de los pies y utilizan para ello una serie de *mañas* o técnicas, tanto de manos como de pies o combinadas y en las que está prohibido el golpeo.

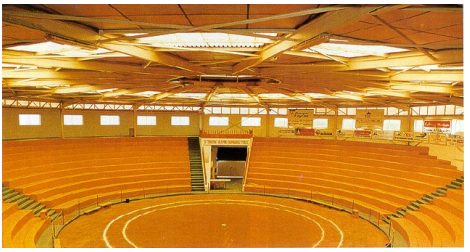


Figura 3.1. Terrero de lucha

La lucha se desarrolla dentro de un círculo de tierra con dos círculos centrales concéntricos de un diámetro de 15 y 17 metros cada uno, llamado *terrero*[*].

Existen varios sistemas de lucha: por equipos, ya sea *a tres agarradas*, *lucha corrida*, o *todos contra todos*; enfrentamientos individuales, por pesos, categorías o desafíos. La forma más habitual se desarrolla entre dos equipos, por lo general de doce luchadores cada bando, enfrentándose individualmente.

El encuentro por equipos se desarrolla como vemos en el siguiente ejemplo. En la primera gráfica se enfrentan primero con primero y segundo con segundo, en la siguiente gráfica vemos que gana el primero del equipo A y pasa a enfren-

[*] Terrero: es un campo de arena circular donde se práctica la lucha canaria.

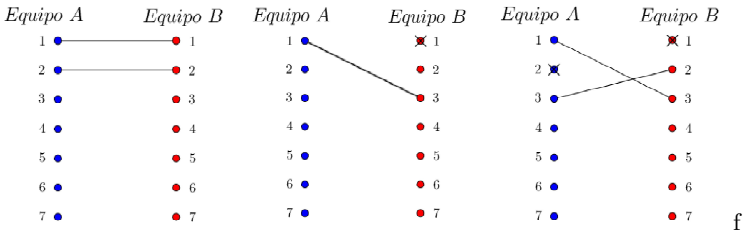


Figura 3.2. Enfrentamientos por equipo:

tarse con el tercer del equipo B y, en la última gráfica el segundo del equipo B gana y pasa a enfrentarse con el tercero del equipo contrario.

Nuestro objetivo primero es estudiar el enfrentamiento de dos luchadores cualesquiera y luego el problema principal que es construir una estimación de un ranking de luchadores a partir de los resultados de diversos enfrentamientos entre algunos luchadores.

4

Enfrentamiento de dos luchadores

Previamente vamos aclarar dos conceptos:

Figura 4.1. Enfrentamiento

- *Agarrada*: es cada uno de los tiempos fijados reglamentariamente de los que consta un encuentro entre dos luchadores. En cada agarrada puede ganar uno de ellos o quedar empate (*separada*).
- *Sistema 3 las 2 mejores*: este tipo de enfrentamiento consta de tres agarradas, cada una de ellas tiene una duración de un minuto y medio, y en caso de empate hay una cuarta agarrada con duración de un minuto. Aquí los resultados que pueden darse al final del enfrentamiento es que gane uno de los dos luchadores o que salgan eliminados ambos.

Aquí empezamos a estudiar y modelizar el enfrentamiento de dos luchadores cualesquiera.

4.1. Construcción de la cadena de Markov

Cada agarrada entre el luchador A y otro B puede acabar con la victoria de uno o con empate. Denotemos por:

- p_A: probabilidad de ganar el luchador A.
- p_E: probabilidad de empate.
- p_B: probabilidad de ganar el luchador B.

Para identificar los posibles estados y refinar más el modelo, pues usaremos las siguientes notaciones:

- $E_i = \boxed{\text{a}}\boxed{\text{s}}\boxed{\text{b}}$ son los estados transitorios, $\forall i \in \{0, \ldots, 9\}$, donde a es el número de agarradas ganadas por el luchador A, s es el número de empatadas y b es el número de ganadas por B, con a, $b \in \{0, 1\}$ y $s \in \{0, 1, 2\}$.

- Los estados absorbentes son:
  - $E_{10} = \boxed{\text{A}}$: gana el enfrentamiento el luchador A.
  - $E_{11} = \boxed{\text{X}}$: eliminados ambos luchadores.
  - $E_{12} = \boxed{\text{B}}$: gana el enfrentamiento el luchador B.

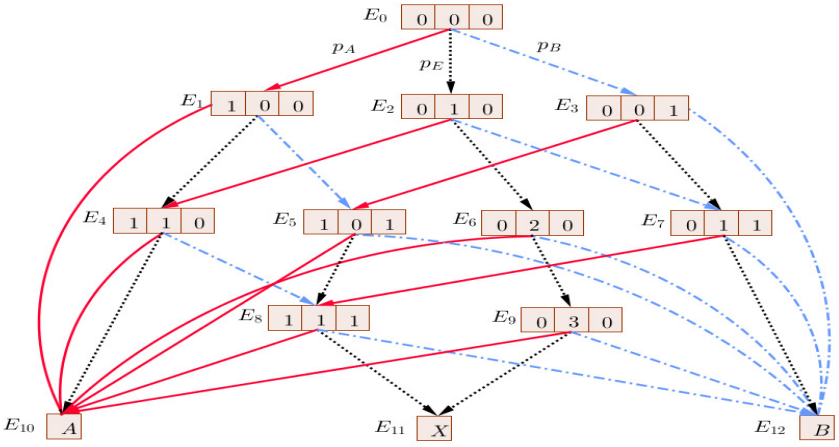


Figura 4.2. Enfrentamiento de dos luchadores en el sistema tres las dos mejores.

A partir de la figura (4.2) podemos construir la matriz de transición M para el sistema tres las dos mejores de la siguiente forma:

$$
\begin{array}{c}
 \\
E_0 \\ E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \\ E_6 \\ E_7 \\ E_8 \\ E_9 \\ E_{10} \\ E_{11} \\ E_{12}
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccccc}
E_0 & E_1 & E_2 & E_3 & E_4 & E_5 & E_6 & E_7 & E_8 & E_9 & E_{10} & E_{11} & E_{12}
\end{array} \\
\left(\begin{array}{cccccccccccccc}
0 & p_A & p_E & p_B & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & p_E & p_B & 0 & 0 & 0 & 0 & p_A & 0 & 0 \\
0 & 0 & 0 & 0 & p_A & 0 & p_E & p_B & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & p_A & 0 & p_E & 0 & 0 & 0 & 0 & p_B \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_B & 0 & p_A + p_E & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_E & 0 & p_A & 0 & p_B \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_E & p_A & 0 & p_B \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_A & 0 & 0 & 0 & p_E + p_B \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_A & p_E & p_B \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_A & p_E & p_B \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)
\end{array}
\tag{4.1}
$$

4.2. Funciones de probabilidad en un enfrentamiento

A partir de las probabilidades p_A y p_B de ganar en una agarradas queremos calcular las probabilidades en el enfrentamiento y que se denotarán así:

- P_A : probabilidad de que el luchador A gane el enfrentamiento.
- P_E : probabilidad de ser eliminados ambos.
- P_B : probabilidad de que el luchador B gane el enfrentamiento.

Vamos a dibujar las tres gráficas de las funciones de probabilidad de enfrentamientos. Primero construimos las funciones enfrentamientos suponiendo que la probabilidad de empate $p_E = 0$, luego en la segunda gráfica fijamos la probabilidad de empate, por ejemplo $p_E = 0.2$.

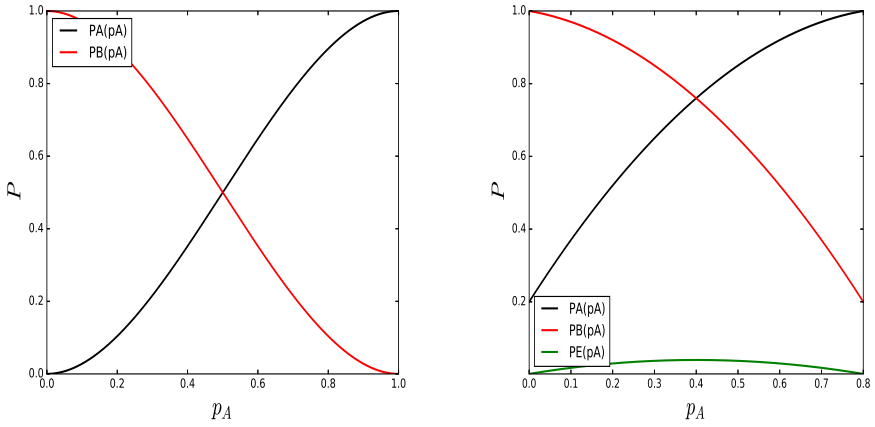
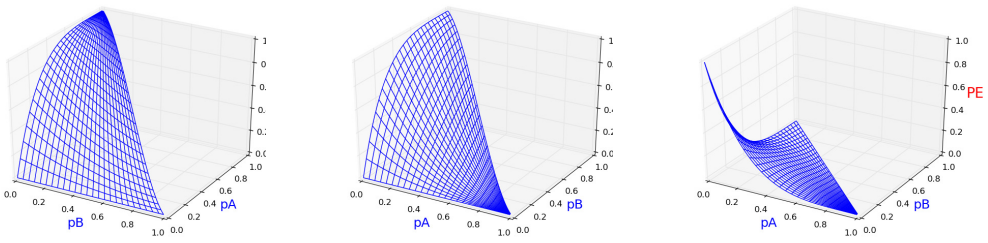


Figura 4.3. Gráficas de probabilidad sin empate(izquierda) y con empate fijo(derecha)

Después en la tercera figura hacemos variar p_E obteniendo las funciones de enfrentamientos. Observamos que el enfrentamiento acaba a lo sumo tras cuatro agarradas ($n = 4$), (es decir, cae en alguno de los tres estados absorbentes). Para obtener las probabilidades que nos interesan calculamos la matriz de transición M elevada a 4. Por lo tanto de la matriz $M^4 = H = [H_{ij}]$ nos interesa los elementos $H_{0,10}$, $H_{0,11}$ y $H_{0,12}$ que son respectivamente las probabilidades en el enfrentamiento P_A, P_E y P_B.

Se observa que las gráficas $P_A = P_A(p_A, p_B)$ y $P_B = P_B(p_A, p_B)$ son simétricas.



Función P_A *Función P_B* *Función P_E*

Figura 4.4. Funciones probabilidad de enfrentamiento

5

Modelo para la estimación de un ranking

En esta sección vamos a modelizar el problema de la estimación de un ranking ajustado a datos. Sea D un registro de resultados de los enfrentamientos previos de los luchadores, entonces queremos encontrar un vector de valoraciones $(v_1, \ldots, v_n)$ tal que se ajuste a los datos. Para ello vamos construir y maximizar una *función* que se encarga de medir cuánto de verosímil son las probabilidades de que se den estos datos dadas las probabilidades por agarradas en función de las valoraciones. Dicha función depende de:

- Registros de datos obtenidos.
- Valoraciones de cada jugador.
- Probabilidades de cada agarrada.
- Probabilidades de cada enfrentamiento.

Empezamos con el enfrentamiento de dos luchadores A y B cualesquiera, donde v_A y v_B son las respectivas valoraciones. Entonces a partir de las valoraciones se asignan las probabilidades por agarrada p_A y p_B, y luego las probabilidades por enfrentamiento P_A, P_E y P_B.

$$\left.\begin{matrix} v_A \\ v_B \end{matrix}\right\} \xrightarrow{\text{Paso 1}} (p_A, p_B) \xrightarrow{\text{Paso 2}} (P_A, P_E, P_B).$$

Paso 1: Como sólo tenemos valoraciones de dos luchadores cualesquiera, entonces solo podemos calcular las probabilidades en función sólo de las valoraciones sin tener probabilidad de empate, esto es:

$$p_A = \frac{v_A}{v_A + v_B} \qquad p_B = \frac{v_B}{v_A + v_B}.$$

Luego necesitamos establecer un *criterio* para calcular la probabilidad de empate p_E. Dadas las valoraciones v_A y v_B de los luchadores A y B, respectivamente, proponemos asignar p_E de la siguiente manera:

$$p_E = \frac{l - |v_A - v_B|}{l},$$

que crece a medida que las valoraciones están muy próximas, donde l es la longitud del rango de valoraciones. Entonces podemos calcular las probabilidades de ganar uno u otro en función de las valoraciones y de p_E:

$$\begin{cases} p_A = \frac{v_A}{v_A+v_B} p_E \\ \\ p_B = \frac{v_B}{v_A+v_B} p_E \end{cases}.$$

Paso 2: Calculamos las probabilidades de enfrentamiento P_A, P_E y P_B usando la matriz M (4.1) y tomando los elementos adecuados de M^4.

En la construcción de la función hay que tener en cuenta todas las valoraciones, es decir, si cambia en alguno de ellos la valoración entonces repercutirá en el resto.

Sea $D_{ij} = (w_{ij}, t_{ij}, w_{ji})$ el registro de dos luchadores, donde:

- ω_{ij} es el número de enfrentamientos ganados del luchador i.
- t_{ij} es el número de eliminados ambos.
- ω_{ji} es el número de ganados del luchador j.

Nótese que, si por ejemplo, tenemos las probabilidades de enfrentamiento $P_A = 0.1$, $P_E = 0.3$ *y* $P_B = 0.6$ y, por otro lado tenemos que el registro es $\omega_{AB} = 8$, $t_{AB} = 2$ y $\omega_{BA} = 1$, entonces es natural preguntarse con estas probabilidades de enfrentamientos cuánto es de probable de que se dé este registro. Para calcular la probabilidad necesitamos usar una distribución multinomial $M(n_{AB} = 11; P_A = 0.1, P_E = 0.3, P_B = 0.6)$[*].

Luego, en general, teniendo las probabilidades por enfrentamientos $P_A(v_i, v_j)$, $P_E(v_i, v_j)$ y $P_B(v_i, v_j)$, entonces la probabilidad de darse el registro D_{ij} entre los luchadores i y j es $P(D_{ij}, P_A, P_E, P_B)$ (para simplificar la notación la denotaremos así $P(D_{ij}, v_i, v_j)$). Por lo tanto, podemos definir una función que dependa del registro D y de todas las valoraciones como:

$$f(D, v_1, \ldots, v_n) = \sum_{\substack{i,j=1 \\ i<j}}^{n} P(D_{ij}, v_i, v_j)(w_{ij} + t_{ij} + w_{ji}). \tag{5.1}$$

Cada factor $P(D_{ij}, v_i, v_j)$ en los sumandos de(5.1) se obtiene a partir de una distribución multinomial con parámetros n_{ij}, P_A, P_E y P_B. Entonces consideremos la variable aleatoria tridimensional:

$$(W_{ij}, T_{ij}, W_{ji}) \backsim M(n_{ij}; P_A, P_E, P_B),$$

[*] Distribución multinomial: Consideremos un fenómeno aleatorio que puede presentarse bajo las alternativas disjuntas $A_1, \ldots, A_k$, de manera que $P(A_j) = p_j$, siendo $\sum_{j=1}^{k} p_j = 1$.

Sea ahora el vector aleatorio $(\xi_1, \ldots, \xi_k)$ que describe el suceso en el que de n observaciones, o repeticiones, se obtenga x_1 veces A_1, x_2 veces A_2, y así hasta x_k veces A_k, de manera que $x_1 + \cdots + x_k = n$.

La *función de probabilidad* conjunta de este vector aleatorio, considerando ya todas las ordenaciones posibles en que puede obtenerse este suceso conjunto, será:

$P(\xi_1 = x_1, \ldots, \xi_k = x_k) = \frac{n!}{x_1! \ldots x_k!} p_1^{x_1} \ldots p_k^{x_k}$.

que son los resultados entre los luchadores i y j tal que W_{ij} es el número de enfrentamientos ganados por i al j con P_A probabilidad de ganar cada enfrentamiento, T_{ij} es el número de enfrentamientos eliminados ambos con P_E probabilidad de estar eliminados ambos en cada enfrentamiento y W_{ji} es el número de enfrentamientos perdidos por i con j con P_B probabilidad de perder cada enfrentamiento. Luego:

$$\begin{aligned} P\left(D_{ij}, v_i, v_j\right) &= P\left(D_{ij}, P_A(v_i, v_j), P_E(v_i, v_j), P_B(v_i, v_j)\right) \\ &= P\left(W_{ij} = w_{ij}, T_{ij} = t_{ij}, W_{ji} = w_{ji}\right) \\ &= \frac{n_{ij}!}{w_{ij}! + t_{ij}! + w_{ji}!} P_A^{w_{ij}} P_E^{t_{ij}} P_B^{w_{ji}}, \end{aligned}$$

donde $n_{ij} = w_{ij} + t_{ij} + w_{ji}$.

Por otro lado, estamos multiplicando por n_{ij} para ponderar y así se tendrá más en cuenta los luchadores con mayor número de enfrentamientos.

Con lo cual la fórmula (5.1) queda de la siguiente forma:

$$f\left(D, v_1, \ldots, v_n\right) = \sum_{i<j} \left(\frac{n_{ij}!}{w_{ij}! + t_{ij}! + w_{ji}!} P_A^{w_{ij}} P_E^{t_{ij}} P_B^{w_{ji}} \right) n_{ij}. \tag{5.2}$$

El objetivo es conseguir un vector de valoraciones $(v_1, \ldots, v_n)$ que maximice la función f.

Analizando la construcción de la función vemos que:

- Cubre los casos de tener valoraciones nulas, ya que en el problema real puede suceder esta situación.
- Se observa que se tiene en cuenta la ponderación del número de enfrentamientos de cada par de luchadores, n_{ij}, frente a los restantes. Pero el factor de ponderación no es el adecuado, ya que todos los registros no tienen el mismo peso, (es decir, cuanto mayor es el n_{ij}, implica un mayor reparto de las probabilidades y con esto tenemos probabilidades más pequeñas).
  Por lo tanto debemos encontrar un *factor de ponderación* adecuado que produzca que todos los números de enfrentamientos de cada par de luchadores tengan el mismo peso. Para ello vamos a empezar esta búsqueda con un sencillo ejemplo:

  Ejemplo 5.0.1.
  Tenemos tres clases de bolas A, B y C, donde las cantidades de cada tipo son N_A, N_B y N_C que son desconocidas y el experimento consiste en coger una bola y después devolverla a la urna. (Supongamos $N_A > N_B > N_C$). Ahora en cada uno de los tres casos vamos a coger un par de clases de bolas y depositarlas en una urna y vamos a realizar dicho experimento:
  1. (A, B) : En la urna colocamos todas las bolas de tipo A y B, realizamos el experimento $n_{AB} = 5$ veces y obtenemos 3 bolas de tipo A y 2 de tipo B:

$$\begin{aligned} (n_A, n_B) = (3,2) &\Rightarrow P\left((3,2) | N_A, N_B, N_C\right) \\ &= \left(\frac{N_A}{N_A + N_B}\right)^3 \left(\frac{N_B}{N_A + N_B}\right)^2 \binom{5}{3}. \end{aligned}$$

2. (B, C) : En la urna colocamos todas las bolas B y C, realizamos el experimento $n_{BC} = 1$ vez y los posibles resultados que podemos obtener son:

$$(n_B, n_C) = (1, 0) \Rightarrow P\left((1, 0)|N_A, N_B, N_C\right) = \frac{N_B}{N_B + N_C}.$$
$$(n_B, n_C) = (0, 1) \Rightarrow P\left((0, 1)|N_A, N_B, N_C\right) = \frac{N_C}{N_B + N_C}.$$

Sabemos que hay más cantidad de bolas de tipo B que de C, entonces es más probable de obtener $n_B = 1$ y $n_C = 0$.
Por otro lado, vemos que en la situación (B, C) hay un menor número de casos respecto a la situación anterior (A, B), entonces en (B, C) habrá un menor número de reparto de probabilidades. Con lo cual, tendremos probabilidades más elevadas en (B, C) que en (A, B).

3. (A, C) : En la urna colocamos todas las bolas A y C, realizamos el experimento $n_{AC} = 2$ veces y obtenemos bolas tipo A en las dos ocasiones:

$$(n_A, n_C) = (2, 0) \Rightarrow P\left((2, 0)|N_A, N_B, N_C\right) = \left(\frac{N_A}{N_A + N_C}\right)^2.$$

Por otro lado, vemos que es más probable que salga $n_A = 2$ y $n_C = 0$.

A partir de este ejemplo podemos construir un nuevo factor de ponderación. Sabemos que cada número de enfrentamientos de cada par de luchadores se compone de tres datos $(\omega_{ij}, t_{ij}, \omega_{ji})$, donde $n_{ij} = \omega_{ij} + t_{ij} + \omega_{ji}$. Vemos que:

- No entran todos los registros sino los dos luchadores que se enfrentan.
- No importa el orden de los resultados de los enfrentamientos.
- Se repiten los resultados.

Entonces, las formas de colocar los tres tipos de resultados son:

$$CR_3^{n_{ij}} = \binom{n_{ij} + 3 - 1}{3 - 1} = \binom{n_{ij} + 2}{2} = C_2^{n_{ij}+2}.$$

La función nos queda así:

$$\boxed{f\left(D, v_1, \ldots, v_n\right) = \sum_{i<j} \left(\frac{n_{ij}!}{w_{ij}! + t_{ij}! + w_{ji}!} P_A^{w_{ij}} P_E^{t_{ij}} P_B^{w_{ji}}\right) \binom{n_{ij} + 2}{2}}. \tag{5.3}$$

6

Resolución del modelo del problema principal

En la resolución del modelo probabilístico para la estimación de un ranking se observa que es un problema casi imposible de resolver desde el punto de vista del análisis matemático, ya que contiene muchas variables y además hay variables que dependen de otras, y por ello recurrimos a los métodos iterativos para resolverlo.

Para la construcción de un método iterativo previamente nos basamos en el *algoritmo PageRank* que *Google* usa para dar valoraciones a las páginas web. Este algoritmo utiliza el amplio sistema de enlaces/hipervínculos como un indicador de la trascendencia de una página web en concreto. Google interpreta un link de una página A a una página B como un voto para la B; pero también tiene en cuenta como factor cuál es en sí la página que emite el voto, es decir, los votos emitidos por las páginas "*importantes*"(PageRank alto), valen más y, por tanto, ayudan a hacer destacar en ella a otras páginas. En el problema de estimación de un ranking pues en vez de páginas web son luchadores y por otro lado usando los mismos argumentos en la explicación del funcionamiento de Google nos planteamos de forma natural la siguiente cuestión ¿cómo podemos asignar valoraciones a los luchadores?

Primero vamos almacenar los registros de los luchadores en dos matrices, donde G son los enfrentamientos ganados y T son los eliminados ambos:

$$W = \begin{pmatrix} 0 & \omega_{12} & \cdots & \omega_{1n} \\ \omega_{21} & 0 & \cdots & \omega_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{n1} & \omega_{n2} & \cdots & 0 \end{pmatrix} \quad y \quad T = \begin{pmatrix} 0 & t_{12} & \cdots & t_{1n} \\ t_{21} & 0 & \cdots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \cdots & 0 \end{pmatrix}.$$

Es decir, cuando se enfrentaron los luchadores 1 y 2 entre sí, 1 ganó ω_{12} encuentros, 2 venció en ω_{21} ocasiones y en $t_{12} = t_{21}$ quedaron eliminados.

Asignar valoraciones a los luchadores nos va a permitir ajustar las probabilidades P_A, P_E y P_B en cada enfrentamiento entre dos ciertos luchadores. Y dicho ajuste lo realizamos mediante un algoritmo recursivo hasta obtener las valoraciones óptimas de la siguiente forma:

$$\begin{cases} V^{(0)} = \left(v_1^{(0)}, \ldots, v_n^{(0)}\right) \\ h\left(D, V^{(m)}\right) = V^{(m+1)}, \quad \forall m \in \mathbb{N} \end{cases} \tag{6.1}$$

donde $V^{(0)}$ es el vector de valoraciones inicial, D son los datos o registros, $V^{(m)}$ vector valoraciones en el paso m y $V^{(m+1)}$ vector valoraciones en el paso $m+1$.

Observación 6.0.1. *Cuando un algoritmo converge tenemos que para un registro de datos el algoritmo convergerá a un vector de valoraciones óptimo. Si el vector de valoraciones inicial está más cerca del vector óptimo la convergencia será más rápida.*

6.1. Primer Algoritmo

A partir del registro de datos $D = (W, T)$ nos interesa sólo los enfrentamientos ganados de cada luchador con el resto, (esto es, la matriz W). Con lo cual, podemos construir el algoritmo recursivo de la siguiente forma:

$$\begin{aligned} g\colon \mathbb{M}_{n\times n}(\mathbb{R}) &\longrightarrow \mathbb{M}_{n\times n}(\mathbb{R}) \\ W^{(m)} &\longmapsto g\left(W^{(m)}\right) = W^{(m+1)}, \quad \forall m \in \mathbb{N}, \end{aligned}$$

$$\begin{aligned} h\colon \mathbb{M}_{n\times n}(\mathbb{R}) &\longrightarrow \mathbb{R}^n \\ W^{(m+1)} &\longmapsto h\left(W^{(m+1)}\right) = V^{(m+1)}, \quad \forall m \in \mathbb{N}, \end{aligned}$$

donde usamos la siguiente notación:

$$\overline{W}^{(m)} = W^{(m-1)}W^{(m-1)}, \quad \forall m \geq 1.$$

En la primera iteración $m = 0$ es un caso particular:

$$\left.\begin{array}{l} W^{(0)} = W \\ s\left(W^{(0)}\right) = \displaystyle\sum_{i,j=1}^{n} \omega_{ij}^{(0)} \end{array}\right\} \Rightarrow g\left(W^{(0)}\right) = \frac{W^{(0)}}{s\left(W^{(0)}\right)} = W^{(1)} \Rightarrow$$

$$\Rightarrow h\left(W^{(1)}\right) = \left(\sum_{j=1}^{n} \omega_{1j}^{(1)}, \ldots, \sum_{j=1}^{n} \omega_{nj}^{(1)}\right) = V^{(1)}.$$

A partir de la iteración $m \geq 1$ realizamos las siguientes operaciones:

$$\left.\begin{array}{l} \overline{W}^{(m)} = W^{(m-1)}W^{(m-1)} \\ s\left(\overline{W}^{(m)}\right) = \displaystyle\sum_{i,j=1}^{n} \overline{\omega_{ij}}^{(m)} \end{array}\right\} \Rightarrow g\left(W^{(m-1)}\right) = \frac{\overline{W}^{(m)}}{s\left(\overline{W}^{(m)}\right)} = W^{(m)} \Rightarrow$$

$$\Rightarrow h\left(W^{(m)}\right) = \left(\sum_{j=1}^{n} \omega_{1j}^{(m)}, \ldots, \sum_{j=1}^{n} \omega_{nj}^{(m)}\right) = V^{(m+1)}.$$

y en cada iteración vamos comprobando esta *condición* o *criterio de parada*:

$$\left|\left|V^{(m+1)} - V^{(m)}\right|\right| < \varepsilon, \quad \forall \varepsilon > 0.$$

Se llamará V^{∞} al resultado del algoritmo si llega alcanzar la condición de parada.

Observación 6.1.1.

- Este método tiene en cuenta los diferentes resultados de enfrentamientos de cada par de luchadores.
  Se puede observar cómo varían las valoraciones si aplicamos el algoritmo en dos casos distintos. En el primer caso el número de enfrentamientos entre dos luchadores cualesquiera A y B es por ejemplo 4, donde $\omega_{AB} = 1$ y $\omega_{BA} = 3$ y en otra situación el número de enfrentamientos es 8, donde $\omega_{AB} = 2$ y $\omega_{BA} = 6$. Obtenemos los siguientes resultados:

Caso: 1	**A**	**B**	C
A	-	**1**	3
B	**3**	-	3
C	2	0	-

$\longrightarrow$ Vector de valoraciones: $v_1^{\infty} = (0.325, 0.477, 0.198)$.

Tabla 6.1.

Caso: 2	**A**	**B**	C
A	-	**2**	3
B	**6**	-	3
C	2	0	-

$\longrightarrow$ Vector de valoraciones: $v_2^{\infty} = (0.328, 0.528, 0.144)$.

Tabla 6.2.

Por lo tanto, teniendo la misma proporción de enfrentamientos ganados y perdidos entre A y B se observa que no se mantiene la diferencia de valoraciones entre A y B, porque se está considerando el número de enfrentamientos ganados de todos los luchadores. Además al mantener el mismo número de enfrentamientos el luchador C baja su valoración, porque en cada iteración se está dividiendo por la suma de todos los elementos de la matriz $W^{(m)}$.

- Notemos que hemos obviado los enfrentamientos eliminados T para simplificar el algoritmo. Pero en el caso de que se quiera incluirlos proponemos medio punto al jugador por cada eliminación que tenga. Entonces tenemos una nueva matriz de datos G, donde:

$$G = W + \frac{1}{2}T.$$

Por lo tanto al usar la matriz G en el algoritmo, vamos a tener un nuevo vector de valoraciones mejor y más ajustado a la tabla de datos D.

Ejemplo 6.1.1.
Tenemos la tabla de datos de tres luchadores:
Almacenamos los datos en las matrices W y T de enfrentamientos ganados y eliminados:

Datos	A	B	C
A	-	(1,1,3)	(3,0,2)
B	-	-	(3,5,0)
C	-	-	-

Tabla 6.3.

$$W = \begin{pmatrix} 0 & 1 & 3 \\ 3 & 0 & 3 \\ 2 & 0 & 0 \end{pmatrix} \qquad \text{y} \qquad T = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 5 \\ 0 & 5 & 0 \end{pmatrix}.$$

A continuación aplicamos el algoritmo:

1. Con la matriz W y nos sale el siguiente vector de valoraciones $v_1^\infty = (0.325, 0.477, 0.198)$. Aquí el luchador mejor valorado es el B y el peor es el C.
2. Con la matriz $G = W + {}^T/_2$ tenemos $v_2^\infty = (0.268, 0.443, 0.288)$. En este caso sigue siendo el luchador mejor valorado el B, pero ahora el peor valorado es el A.

Ejemplo 6.1.2.
Tenemos ahora cinco luchadores, con el registro de datos de la tabla 6.4:

Datos	A	B	C	D	E
A	-	(0,18,0)	(0,6,1)	(0,8,2)	(1,1,0)
B		-	(0,0,5)	(1,0,7)	(1,2,3)
C	-	-	-	(2,1,7)	(2,6,5)
D	-	-	-	-	(7,0,6)
E	-	-	-	-	-

Tabla 6.4.

A continuación aplicamos el algoritmo:

1) Con la matriz W y nos sale el siguiente vector de valoraciones:

$$v_1^\infty = (0.181, 0.131, 0.184, 0.284, 0.219).$$

Aquí por orden decreciente:

$$v_D, v_E, v_C, v_B, v_A.$$

2) Con la matriz $G = W + {}^T/_2$ tenemos:

$$v_1^\infty = (0.032, 0.069, 0.175, 0.393, 0.331).$$

En este caso:

$$v_D, v_E, v_A, v_C, v_B.$$

Observamos que en el caso 2) tenemos una estimación de un ranking diferente al caso 1). Obviamente la estimación del ranking que se ajusta más al registro de datos es el caso 2), ya que a parte de los enfrentamientos ganados también tenemos en cuenta los eliminados ambos.

6.2. Segundo Algoritmo

A partir de la observación vista en el algoritmo anterior donde se tiene en cuenta el número de enfrentamientos total de cada par de luchadores, entonces surge esta siguiente alternativa de construir otro algoritmo que sea inherente al número de enfrentamientos de cada par de luchadores.

Sea $V^{(0)} = \left(v_1^{(0)}, \ldots, v_n^{(0)}\right)$ un vector de valoraciones inicial y en cada iteración n realizamos las dos siguientes operaciones para calcular el vector de valoraciones $V^{(n+1)}$:

1) Normalizamos el vector $V^{(n)}$ y nos queda otro vector:

$$\overline{V}^{(n)} = \frac{V^{(n)}}{s\left(V^{(n)}\right)}, \text{ con } s\left(V^{(n)}\right) = \sum_{i=1}^{n} v_i^{(n)}.$$

2) Ahora calculamos el vector de valoraciones de la iteración $t+1$:

$$v_j^{(n+1)} = \sum_{i \neq j}^{n} \omega_{ji} \frac{\overline{v_i}^{(n)} + \overline{v_j}^{(n)}}{\omega_{ji} + \omega_{ij}}, \quad \forall j \in \{1, \ldots, n\}, \tag{6.2}$$

donde ω_{ji} es el número de enfrentamientos que perdió i frente a j y $v_i^{(n)}$ es la valoración del luchador i en la iteración n.

Otra forma más compacta de escribir la fórmula (6.2) es poner los coeficientes $\frac{\omega_{ji}}{\omega_{ji} + \omega_{ij}}$ en una matriz:

$$\left[\frac{\omega_{ji}}{\omega_{ji} + \omega_{ij}}\right] = \frac{W^t}{W^t + W} = A.$$

Entonces matricialmente nos queda el algoritmo así:

$$\begin{cases} \overline{V}^{(n)} = \dfrac{V^{(n)}}{s\left(V^{(n)}\right)} \\ V^{(n+1)} = A\overline{V}^{(n)} \end{cases}, \forall t = 0, 1, \cdots .$$

Observación 6.2.1.

- Comparamos el primer y segundo algoritmo con este ejemplo de 3 luchadores:

Caso: 1	A	B	C
A	-	1	3
B	3	-	3
C	2	0	-

Tabla 6.5.

En el *Algoritmo 1* nos centramos en las partidas ganadas W y tenemos el siguiente vector de valoraciones:

$$v_1^\infty = (0.325, 0.477, 0.198)\,.$$

En el *Algoritmo 2* nos centramos en las partidas perdidas W^t y tenemos el siguiente vector de valoraciones:

$$v_2^\infty = (0.190, 0.762, 0.048)\,.$$

Concluimos que parece que el vector v_2^∞ se ajusta más a los datos y también parece mejor estimación de un ranking.

- Ahora si doblamos proporcionalmente el número de enfrentamiento y aplicamos este algoritmo para esta nueva situación. Vemos que obtenemos el mismo vector de valoraciones:

Caso: 2	**A**	**B**	C
A	-	**2**	3
B	**6**	-	3
C	2	0	-

$\longrightarrow$ Vector de valoraciones: $v_3^\infty = (0.190, 0.762, 0.048)\,.$

Tabla 6.6.

6.3. Mejoras del algoritmo respecto a uso de los datos

Para refinar más este método podríamos tener en cuenta el aspecto temporal de los datos y las victorias recientes. Para ello se podría aplicar un descuento $d \in [0, 1)$ por cada mes que ha pasado. Con lo cual, nos quedaría una matriz S, con cada:

$$s_{ij} = \sum_{l=0}^{m} (1-d)^{m-l} \omega_{ij}^{(l)} \in \mathbb{R}^+,$$

donde m es el número de meses transcurridos desde el dato hasta este momento.

Por lo tanto al aplicar descuento sobre los enfrentamientos ganados a medida que transcurre el tiempo tenemos otra estimación de ranking de luchadores.

7

Conclusiones

Este trabajo trata sobre la Lucha Canaria, donde primero se modelizó el problema del enfrentamiento entre dos luchadores y, para ello se diferenciaron los tipos de sucesos, se realizó una representación gráfica para construir la cadena de Markov y se obtuvieron las funciones de probabilidades de los enfrentamientos en función de las probabilidades por agarrada.

Después se modelizó el problema de asignación de valoraciones a los luchadores, donde se trató de encontrar un vector de valoraciones tal que se ajustara a los datos. A partir de aquí construimos una función que se encarga de medir la verosimilitud de las probabilidades de que se tenga un cierto registro. Ahora, para maximizar esta función del modelo se acudió a los métodos iterativos, donde se propusieron dos algoritmos recursivos. Se observó que en los registros de datos que se habían usado en los ejemplos, los algoritmos tendían a un cierto vector de valoraciones, que hemos asignado como vector límite.

Por otro lado, sería interesante ampliar este trabajo considerando, más que una valoración unidimensional, un rango de valoraciones que sea multidimensional, por ejemplo, aspectos de resistencia, habilidad,... Podría tratarse también en un futuro trabajo el problema del enfrentamiento de dos equipos, teniendo en cuenta todos los luchadores a la vez.

Finalmente, queremos destacar la experiencia de haber realizado un trabajo aplicando algunos conocimientos adquiridos en la carrera para abordar un problema real. Hemos comprobado la gran utilidad de una herramienta matemática (las cadenas de Markov) en diversas áreas como física, economía, meteorología,... y además de tener otro enfoque de trabajo muy diferente a como se estudia en una asignatura común durante la carrera.

8

Apéndice I

El lenguaje de programación que hemos usado es Python 2.7. Además es necesario cargar los siguientes módulos para poder usar estas funciones.

```
from numpy import *
from sympy import *
from scipy import array
from math import sqrt
from pylab import *
from mpl_toolkits.mplot3d.axes3d import Axes3D
import matplotlib.pyplot as plt
import random
```

8.1. Algoritmo primero

```
#############################################################################
#
# AUTORES Zebensuí Hernández Díaz
#
# FECHA 17 febrero 2016
#
# DESCRIPCION Primera resolución del modelo de estimación de un ranking de
# luchadores
#
#############################################################################

def alg_ranking1(M,n):
   A = M
   #Sumamos todos los elementos de la matriz para dividir
   suma = suma_elemento(A,n)
```

```
   C = A/suma
   #Sumamos todos los elementos
   suma_ini = suma_fila(C,n)
   #Ponemos un chivato para avisar cuando se para el algoritmo
   diferencia_grande = True
   i = 0
   Error = 10**-9

   while diferencia_grande == True:
      i = i+1
      #Multiplicamos dos matrices
      B = np.dot(A,M)
      #Sumamos todos los elementos
      suma = suma_elemento(B,n)
      C = B/suma
      suma_fil = suma_fila(C,n)
      #Comprobamos si el vector de dos iteraciones seguidas son iguales:
      if np.linalg.norm(suma_ini-suma_fil) < Error:
         #Si el chivato cambia a True se para el algoritmo
         diferencia_grande = False
      suma_ini = suma_fil
      #Guardamos sobre la matriz A la nueva matriz producto B:
      A = B

   print "Vector de valoraciones óptimo en la iteración %i:"%(i)
   print suma_fil
   #Reescalamos las valoraciones ponderadas óptimas de cada luchador con
   #el resto en la iteración %i son:" %(i)
   vect_optimo = reescalado(suma_fil,n)
   print "La estimación de un ranking de los luchadores es:"
   #Imprimimos por pantalla el vector de valoraciones óptimo reescalado en
   #[0,100]:
   presentacion(vect_optimo,n)
   return suma_fil
```

8.2. Algoritmo segundo

```
 ##############################################################################
 #
# AUTORES Zebensuí Hernández Díaz
#
# FECHA 17 febrero 2016
#
# DESCRIPCION Segunda resolución del modelo de estimación de un ranking de
# luchadores
```

```
 #
 ##########################################################################

#Generamos el vector inicial de las matrices Ganados, Eliminados y Perdidos:
V_inic = vector_inicial(Gan,Per,Emp,n)

def alg_ranking2(M,V_inic,n):
   V_new = np.zeros((n,1))
   N = modificacion_matriz(M,n)
   #Ponemos un chivato para avisar cuando se para el algoritmo:
   diferencia_grande = True
   i = 0
   Error = 10**-9

   while diferencia_grande == True:
      i = i+1
      V_new = prod_mat_vector(N,V_inic,n)
      m = suma_vector(V_new,n)
      V_new = V_new/m
      #Comprobamos si el vector de dos iteraciones consecutivas son iguales:
      if np.linalg.norm(V_inic-V_new) < Error:
         #Si el chivato cambia a True se para el algoritmo
         diferencia_grande = False
      V_inic = V_new

   print "Iteración %i" %(i)
   print "Vector de valoraciones óptimo en la iteración %i:" %(i)
   print V_new
   #Reescalamos las valoraciones ponderadas óptimas de cada luchador con el
   #resto en la iteración %i son:" %(i)
   vect_optimo = reescalado(V_new,n)
   print "La estimación de un ranking de los luchadores es:"
   #Imprimimos por pantalla el vector de valoraciones óptimo reescalado en
   #[0,100]:
   valoraciones = presentacion(vect_optimo,n)
   valoraciones_ordenadas = ordenar_ranking(valoraciones,n)
   return valoraciones_ordenadas
```

8.3. Funciones usadas para los dos algoritmos

```
 ##########################################################################
 #
# AUTORES Zebensuí Hernández Díaz
#
# FECHA 17 febrero 2016
```

```
#
# DESCRIPCION Funciones usadas en los dos algoritmos anteriores
 #
 ##############################################################################
%*****************************************************************
Funciones para el primer algoritmo:
%*****************************************************************
#Sumar todos los elementos de una matriz:
def suma_elemento(M,n):
   suma = 0
   for i in range(n):
      for j in range(n):
         suma = suma + M[i][j]
   return suma

#Sumar componentes por fila
def suma_fila(M,n):
   suma_fil = np.zeros((n,1))
   for i in range(n):
      suma = 0
      for j in range(n):
         suma = suma + M[i][j]
      suma_fil[i] = suma
   return suma_fil

#Esta función es para reescalar en porcentaje las valoraciones
#para visualizar mejor las diferencias:
def reescalado(vector,n):
   maximo = max(vector)
   vector = (vector*100)/maximo
   return vector

#Para etiquetar los luchadores:
def presentacion(vector,n):
   for i in range(n):
      a = i+1
      print "Luchador %i tiene de valoración: %2.5f" %(a,vector[i])
   return vector

%*****************************************************************
Funciones para el segundo algoritmo:
%*****************************************************************
#Para sacar la matriz del algoritmo segundo:
def modificacion_matriz(M,n):
   N = np.zeros((n,n))
   for i in range(n):
```

```
      for j in range(n):
         if (j != i) and (M[i][j] != 0 or M[j][i] != 0):
            N[i][j] = M[j][i]/(M[i][j]+M[j][i])
   return N
#Es para hacer el producto de una matriz con un vector:
def prod_mat_vector(N,V_inic,n):
   V_new = np.zeros(n)
   for i in range(n):
      suma = 0
      for j in range(n):
         suma =  suma + N[i][j]*V_inic[i]+N[i][j]*V_inic[j]
      V_new[i] = suma
   return V_new

#Sumar los elementos de un vector:
def suma_vector(vector,n):
   suma=0
   for i in range(n):
      suma = suma + vector[i]
   return suma

#Generar un vector unicial de valoraciones:
def vector_inicial(A,B,C,n):
   vector_ganado = suma_fila(A,n)
   vector_perdido = suma_fila(B,n)
   vector_eliminado = suma_fila(C,n)
   total = vector_ganado + vector_perdido + vector_eliminado
   vector = np.zeros(n)
   for i in range(n):
      vector[i] = vector_ganado[i]/total[i]
   return vector

########################################################################
Parte donde se tiene en cuenta enfrentamientos ganados y eliminados:
########################################################################

#Construcción de la matriz de enfrentamientos ganados y eliminados:
def matriz_ganado_perdido_eliminado(A,C,n):
   M = np.zeros((n,n))
   for i in range(n):
      for j in range(n):
         aux = int(C[i][j])
         suma = 0
         for k in range(aux):
            if j!=i:
               suma = suma + 0.5
```

```
            M[i][j] = A[i][j] + suma
            M[j][i] = A[j][i] + suma
    return M

#Ordenar de forma decreciente las valoraciones:
def ordenar_ranking(vector,n):
    vector_indices = np.arange(n)
    for i in range(n):
        for j in range(n):
            if (j>i) and (vector[j] > vector[i]):
                #Cambio de valoraciones:
                aux1 = vector[i]
                aux2 = vector[j]
                vector[i] = aux2
                vector[j] = aux1
                #Cambio de indices:
                indice1 = vector_indices[i]
                indice2 = vector_indices[j]
                vector_indices[i] = indice2
                vector_indices[j] = indice1
    print
    print "*******************************************************"
    print "Estimación de un ranking de los luchadores ordenado es:"
    print "*******************************************************"

    print "Pasamos los datos de estimación de un ranking en un fichero \n"
    nombre = raw_input("Introduce el nombre del archivo: ")
    f = open(nombre, "w")

    f.write("*******************************************************\n")
    f.write("Estimación de un ranking de los luchadores ordenado es:
    (en porcentaje)\n")
    f.write("*******************************************************\n\n")

    #Contar número de luchadores por categoría:
    cont_PA=0
    cont_PB=0
    cont_PC=0
    cont_DA=0
    cont_DB=0
    cont_DC=0
    cont_NC=0
    for i in range(n):
        a = vector_indices[i]+1
        #Definimos la categoría de cada luchador:(Fijamos por cada
        subintervalo de [0,100] a una categoría)
```

```
    if (vector[i]<=100) and (vector[i]>66):
       s = "PA"
       cont_PA = cont_PA + 1
    elif (vector[i]<=66) and (vector[i]>50):
       s = "PB"
       cont_PB = cont_PB + 1
    elif (vector[i]<=50) and (vector[i]>37):
       s = "PC"
       cont_PC = cont_PC + 1
    elif (vector[i]<=37) and (vector[i]>33):
       s = "DA"
       cont_DA = cont_DA + 1
    elif (vector[i]<=33) and (vector[i]>29):
       s = "DB"
       cont_DB = cont_DB + 1
    elif (vector[i]<=29) and (vector[i]>25):
       s = "DC"
       cont_DC = cont_DC + 1
    elif (vector[i]<=25) and (vector[i]>0):
       s = "NC"
       cont_NC = cont_NC + 1
    print "    Posición %i --> Luchador %i --> Valoración: %2.5f
    --> Categoría: %s" %(i+1,a,vector[i],s)
    print "   ------------------------------------------------------"

    aux1 = str(i+1)
    aux2 = str(a)
    aux3 = str(vector[i])
    f.write("    Posición %i --> Luchador %i --> Valoración: %2.5f
    --> Categoría: %s \n" %(i+1,a,vector[i],s))
    f.write("   ------------------------------------------------\n")

print "*****************************************************"
print "Número de luchadores PUNTAL A son: %i" %(cont_PA)
print "Número de luchadores PUNTAL B son: %i" %(cont_PB)
print "Número de luchadores PUNTAL C son: %i" %(cont_PC)
print "Número de luchadores DESTACADOS A son: %i" %(cont_DA)
print "Número de luchadores DESTACADOS B son: %i" %(cont_DB)
print "Número de luchadores DESTACADOS C son: %i" %(cont_DC)
print "Número de luchadores NO CLASIFICADOS son: %i" %(cont_NC)
print "*****************************************************"

n_PA = str(cont_PA)
n_PB = str(cont_PB)
n_PC = str(cont_PC)
n_DA = str(cont_DA)
```

```
   n_DB = str(cont_DB)
   n_DC = str(cont_DC)
   n_NC = str(cont_NC)
   f.write("*****************************************************\n")
   f.write("Número de luchadores PUNTAL A son: %s \n" %(n_PA))
   f.write("Número de luchadores PUNTAL B son: %s \n" %(n_PB))
   f.write("Número de luchadores PUNTAL C son: %s \n" %(n_PC))
   f.write("Número de luchadores DESTACADOS A son: %s \n" %(n_DA))
   f.write("Número de luchadores DESTACADOS B son: %s \n" %(n_DB))
   f.write("Número de luchadores DESTACADOS C son: %s \n" %(n_DC))
   f.write("Número de luchadores NO CLASIFICADOS son: %s \n" %(n_NC))
   f.write("*****************************************************\n")
   f.close()

   print "Pasamos los datos de estimación de un ranking a un fichero. \n"

   return vector
```

8.4. Simulación de datos respecto con el tiempo

```
#############################################################################
#
# AUTORES Zebensuí Hernández Díaz
#
# FECHA 21 marzo 2016
#
# DESCRIPCION Mejoras del algoritmo respecto a uso de los datos respecto
#al tiempo, ( por ejemplo, en meses).
#
#############################################################################
#Simulación de datos por meses:
def generar_datos_enfren_por_mes(n,V,Long):
   """
   Esta función sirve para simular o generar los resultados que tiene
   cada luchador con el
   resto tal que contenga en cada enfrentamiento:
1ª) Enfrentamientos ganados.
2ª) Enfrentamientos empatados o eliminados ambos.
3ª) Enfrentamientos perdidos.
   Tener en cuenta que el vector V contiene las valoraciones de los lucha-
   dores y Long es la máxima valoración
   """

   "Creamos una lista vacía (para los n luchadores), donde cada elemento
   tiene los resultados de cada luchador con el resto:"
```

```
    L=[]

    "Creamos una lista donde se contine la cantidad de agarradas de cada
    luchador con el resto:"
    C = []
    maximo = 20 #Máximo de agarradas de cada luchador con otro

    for i in range(0,n):
        M = []
        L.append(M)
        "Número de agarradas de cierto luchador con el resto:"
        CL = []
        C.append(CL)
        for j in range(0,n):
            if j<i:
                #Probabilidades del luchador i con el luchador:
                vA = float(V[i])
                vB = float(V[j])
                denominador = ((sqrt(2))/2) * Long - (sqrt(2)/2) * abs(vA-vB)
                numerador = sqrt(2)*Long
                pE = denominador/numerador
     pA = (vA/(vA+vB)) * (1 - pE)
                pB = (vB/(vA+vB)) * (1 - pE)
                #Probabilidades enfrentamiento:
                Enf = Luchada_3_al_2_mejor(pA, pB)
                #Contadores a cero:
     a = 0
                b = 0
                c = 0
                #Cantidad de agarradas de cada luchador i con el luchador j:
     can = randint(maximo)
                if can!=0:
                    for m in range(0,can):
                        p = random.random()
                        if p<Enf[0]:
                            a = a+1
                        elif (p>=Enf[0] and p<Enf[0]+Enf[2]):
                            b = b+1
                        else:
                            c = c+1
                N = [a, b, c]
                M.append(N)
    return L

def simulacion_por_meses(n,m,V,Long):
    H=[]
```

```
   for i in range(m):
      L=generar_datos_enfren_por_mes(n,V,Long)
      H.append(L)
   return H

def almacenar_datos_lista_matrices_ganados(n,m,H):
   Lista=[]
   for i in range(m):
      M = matriz_ganados(H[i],n)
      Lista.append(M)
   return Lista

def almacenar_datos_lista_matrices_perdidos(n,m,H):
   Lista=[]
   for i in range(m):
      M = matriz_perdidos(H[i],n)
      Lista.append(M)
   return Lista

def almacenar_datos_lista_matrices_eliminados(n,m,H):
   Lista=[]
   for i in range(m):
      M = matriz_empatados(H[i],n)
      Lista.append(M)
   return Lista

def suma_lista_matrices(n,m,lista_1,lista_2):
   Totales=[]
   for i in range(m):
      Total = matriz_ganado_perdido_eliminado(lista_1[i],lista_2[i],n)
      Totales.append(Total)
   return Totales

def suma_descuento_matriz_datos(n,m,descuento,lista):
   suma = np.zeros((n,n))
   for i in range(m):
      suma = suma + ((1-descuento)**(m-i))*lista[i]
   Datos = suma
   return Datos

#Vector inicial:#############################################################
def Vector_descuento_inicial(n,A,B,C):
   Total_Enfrentamientos = A + B + C
   Enfrentamientos_global_luchador = suma_fila(Total_Enfrentamientos,n)
   Enfrentamientos_global_ganado_luchador = suma_fila(A,n)
   vector = np.zeros(n)
```

```
   for i in range(n):
      vector[i] = Enfrentamientos_global_ganado_luchador[i]/Enfrentamientos_
      global_luchador[i]
   return vector
#########################################################################

#Ahora para usar estas funciones para generar el registro de datos teniendo
#en cuenta la temporalidad, (por ejemplo por meses) es con las siguientes
#llamadas a las funciones:

#Número de meses:
m=8

#Simulación de registro de datos por meses:
H = simulacion_por_meses(n,m,V,Long)

#Almacenar enfrentamientos ganados por meses:
Ganados = almacenar_datos_lista_matrices_ganados(n,m,H)

#Almacenar enfrentamientos perdidos por meses:
Perdidos = almacenar_datos_lista_matrices_perdidos(n,m,H)

#Almacenar enfrentamientos eliminados por meses:
Eliminados = almacenar_datos_lista_matrices_eliminados(n,m,H)

#Registro total por cada mes:(Lista)
Totales = suma_lista_matrices(n,m,Perdidos,Eliminados)

#Aplicamos DESCUENTO###########################################################
#Descuento:
descuento = 0.4

#Sumar matrices totales de meses descuento:
Datos = suma_descuento_matriz_datos(n,m,descuento,Totales)

#Sumar matrices ganados de meses descuento:
Total_Ganados = suma_descuento_matriz_datos(n,m,descuento,Ganados)

#Sumar matrices perdidos de meses descuento:
Total_Perdidos = suma_descuento_matriz_datos(n,m,descuento,Perdidos)

#Sumar matrices eliminados de meses descuento:
Total_Eliminados = suma_descuento_matriz_datos(n,m,descuento,Eliminados)

#Vector inicial:###############################################################
Vector_inicial = Vector_descuento_inicial(n,Total_Ganados,Total_Perdidos,
```

```
Total_Eliminados)
################################################################################

#Aplicamos el algoritmo:
alg_ranking2(Datos,Vector_inicial,n)
```

9

Apéndice II

Para calcular las valoraciones necesitamos los registros de cada luchador, donde contengan el número de encuentros ganados, empatados y pérdidos con cada contrincante.

Para poder tener un registro de datos hemos realizado una simulación teniendo en cuenta que las probabilidades por agarradas dependen de un par de valoraciones de los luchadores que se enfrentan y por otro lado necesitamos establecer un criterio para sacar la probabilidad de empate (usamos el mismo criterio que en la construcción del modelo de estimación de un ranking).

Esto son los pasos que hemos realizado en la simulación:

1. Inicializamos el vector de valoraciones $V = (v_1, \ldots, v_n)$.

2. Fijamos el rango de valoración, por ejemplo $(0, 10]$, entonces el rango tiene longitud $l = 10$.

4. Previamente se calcula la probabilidad de empate $p_{E_{ij}}$ así:

$$p_{E_{ij}} = \frac{l - |v_i - v_j|}{l}, \tag{9.1}$$

para todo $j \in \{i+1, \ldots, n\}$ e i fijo, donde l es la longitud del rango de valoraciones.

5. Ahora podemos calcular $p_{A_{ij}}$ y $p_{B_{ij}}$:

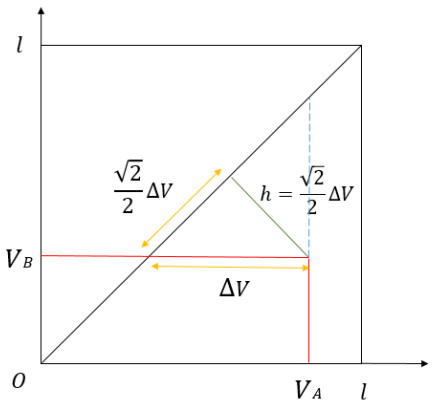


Figura 9.1. Criterio de probabilidad de empate P_E

$$p_{A_{ij}} = \frac{v_i}{v_i + v_j}\, p_{E_{ij}} \qquad (9.2)$$

$$p_{B_{ij}} = \frac{v_j}{v_i + v_j}\, p_{E_{ij}} \qquad (9.3)$$

6. Hacemos una partición del intervalo $I = [0, 1]$ de la siguiente forma:

$$\mathcal{P} = \{I_0, I_1, I_2\} = \left\{ \left[0, p_{A_{ij}}\right), \left[p_{A_{ij}}, p_{A_{ij}} + p_{B_{ij}}\right], \left(p_{A_{ij}} + p_{B_{ij}}, 1\right] \right\}.$$

7. Ahora generamos números aleatorios entre el intervalo $[0, 1]$ y verificamos dónde cae cada número p tal que si:

- $p \in I_0 \Rightarrow$ Gana el luchador i.
- $p \in I_1 \Rightarrow$ Gana el luchador j.
- $p \in I_2 \Rightarrow$ Empata i y j.

8. Así conseguimos, respectivamente:

- $n_{A_{ij}}$: Nº de agarradas ganadas por el luchador i al j.
- $n_{B_{ij}}$: Nº de agarradas ganadas por el luchador j al i.
- $n_{E_{ij}}$: Nº de agarradas empatadas entre los luchadores i y j.

Para calcular las valoraciones necesitamos los registros de cada luchador, donde contengan el número de agarradas ganadas, empatadas y pérdidas.

Observación 9.0.1.
Observando la gráfica (9.1) podemos calcular el área del triángulo equilátero de dos formas:

$$\left.\begin{aligned} A_\Delta &= \frac{1}{2}(|\Delta v|\sqrt{2})h \\ A_\Delta &= \frac{(|\Delta v|)^2}{2} \end{aligned}\right\}$$

Luego de aquí podemos calcular la altura h:

$$|\Delta v| = \sqrt{2}h \Rightarrow h = \frac{|\Delta v|}{\sqrt{2}} \Rightarrow h = \frac{\sqrt{2}}{2}|\Delta v|.$$

Observación 9.0.2.
Podemos tomar también como criterio para calcular el empate así:

$$p_E = 1 - \frac{|v_A - v_B|}{v_A + v_B},$$

donde cuánto más se parece las valoraciones de los luchadores A y B, entonces la probabilidad de empate se acerca a 1.

Bibliografía

[1] Charles M. Grintead and J. Laurie Snell, *Introduction to Probability*, American Mathematical Society, 2012, 2nd edition, 4 July 2006.

[2] Luis Rincón, *Introducción a los PROCESOS ESTOCÁSTICOS* , 2011, `http://www.matematicas.unam.mx/lars`.

[3] Pablo Fernández, *El secreto de Google y el Álgebra lineal*, 2004, Boletín de la Sociedad Española de Matemática Aplicada, `https://sctmates.webs.ull.es/modulo1lp/8/pfernandez.pdf`.

[4] Federación de Lucha Canaria, *Reglamento*.

[5] Libros web: Python para principiantes, `https://librosweb.es/libro/python/`.

[6] Pablo Guido van Rossum, *El tutorial de Python*, 2009, Editor original: Fred L. Drake, Jr, `http://www.python.org.ar/wiki/Tutorial`. http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf

Probabilistic models to study the Canarian Wrestling

Zebensuí Hernández Díaz

Facultad de Ciencias · Sección de Matemáticas
Universidad de La Laguna
alu0100280241@ull.edu.es

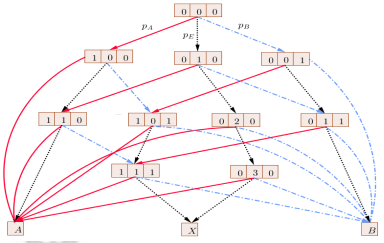
Abstract

This paper tries to model and solve the problem of making an estimation in order to rank the wrestlers in the sport called "Lucha Canaria" with the objective of (1) classifying wrestlers by categories, and (2) to help the coach to organize the wrestlers of his team.

1. Model for the confrontation between two wrestlings

Every clash (*agarrada*) between two wrestlers, A and B, can finish with the victory of one of them or a tie. Let p_A, p_B, p_E the probabilities of victory of A, victory of B and tie (*separada*). In order to identify the possible states, we will use this notation:

- a s b for the transitory states, where a is the number of clashes which wrestler A won, b is the number of clashes which wrestler B won, and s is the number of ties they had, with $a, b \in \{0,1\}$ and $s \in \{0,1,2\}$.
- A, B and X are the absorbing states for the victory of A, the victory of B and both to be pushed-out.



We can build the transition matrix M as:

$$M = \begin{array}{c} \\ 000 \\ 100 \\ 010 \\ 001 \\ 110 \\ 101 \\ 020 \\ 011 \\ 111 \\ 030 \\ A \\ X \\ B \end{array} \begin{array}{c} \begin{array}{ccccccccccccc} 000 & 100 & 010 & 001 & 110 & 101 & 020 & 011 & 111 & 030 & A & X & B \end{array} \\ \begin{pmatrix} 0 & p_A & p_E & p_B & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p_E & p_B & 0 & 0 & 0 & 0 & p_A & 0 & 0 \\ 0 & 0 & 0 & 0 & p_A & 0 & p_E & p_B & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & p_A & 0 & p_E & 0 & 0 & 0 & 0 & p_B \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_B & 0 & p_A + p_E & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_E & 0 & p_A & 0 & p_B \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_E & p_A & 0 & p_B \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_A & 0 & 0 & 0 & p_E + p_B \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_A & p_E & p_B \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_A & p_E & p_B \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \quad (1)$$

Therefore, from the matrix $M^4 = H = [H_{ij}]$ we are interested in elements $H_{0,10}, H_{0,11}, H_{0,12}$ which are the probabilities of match (*enfrentamiento*) P_A, P_E, P_B, respectively.

2. Model for the estimation of a ranking

Let D be a result record of previous match, then we want to find a vector of valuations $(v_1, ..., v_n)$ such that it fits the data. For this we are going to contruct and maximize a function that is in charge of measuring how much of plausible are the probabilities that these data are given the probabilities by clash in function of the valuations:

$$f(D, v_1, \ldots, v_n) = \sum_{i<j} \left(\frac{n_{ij}!}{w_{ij}! + t_{ij}! + w_{ji}!} P_A^{w_{ij}} P_E^{t_{ij}} P_B^{w_{ji}} \right) \binom{n_{ij}+2}{2}, \quad (2)$$

where:

$$\left.\begin{matrix} v_A \\ v_B \end{matrix}\right\} \xrightarrow{\text{Stage 1}} (p_A, p_B) \xrightarrow{\text{Stage 2}} (P_A, P_E, P_B).$$

1: We fix $p_E = \dfrac{l - |v_A - v_B|}{l}$, where l is the size of the range of valorations, so $p_A = \dfrac{v_A}{v_A + v_B} p_E$ and $p_B = \dfrac{v_B}{v_A + v_B} p_E$.

2: We compute the probabilities of match P_A, P_E, P_B using the model for the clash between two wrestlers.

Now the objective function is built considering all valorations, i.e. if the valorations of any wrestler changes, it will change the valorations of the rest of wrestlers.

$$\begin{aligned} P(D_{ij}, v_i, v_j) &= P\left(W_{ij} = w_{ij}, T_{ij} = t_{ij}, W_{ji} = w_{ji}\right) \\ &= \frac{n_{ij}!}{w_{ij}! + t_{ij}! + w_{ji}!} P_A^{w_{ij}} P_E^{t_{ij}} P_B^{w_{ji}}, \end{aligned}$$

where $(W_{ij}, T_{ij}, W_{ji}) \sim M(n_{ij}; P_A, P_E, P_B)$ is a multinomial distribution, P_A, P_B, P_E are the probabilities of the victory of wrestler i in the match, the victory of wrestler j, and both wrestlers to be pushed-out.

3. Resolution of the model for the estimation

Let W be the matrix of matches won, with ω_{ij} the number of maches won by the wrestler i against j. So we can contruct the recursive algorithm as follows:

$$\begin{array}{ccc} g\colon & \mathbb{M}_{n\times n}(\mathbb{R}) & \longrightarrow \mathbb{M}_{n\times n}(\mathbb{R}) \\ & W^{(m)} & \longmapsto W^{(m+1)} \end{array}, \qquad \begin{array}{ccc} h\colon & \mathbb{M}_{n\times n}(\mathbb{R}) & \longrightarrow \mathbb{R}^n \\ & W^{(m+1)} & \longmapsto V^{(m+1)} \end{array}, \quad \forall m \in \mathbb{N},$$

where we notate $\overline{W}^{(m)} = W^{(m-1)} W^{(m-1)}$, $\forall m \geq 1$.

In the first iteration $m = 0$ is a particular case:

$$\left.\begin{aligned} &W^{(0)} = W \\ &s\left(W^{(0)}\right) = \sum_{i,j=1}^{n} \omega_{ij}^{(0)} \end{aligned}\right\} \Rightarrow g\left(W^{(0)}\right) = \frac{W^{(0)}}{s\left(W^{(0)}\right)} = W^{(1)}$$

$$h\left(W^{(1)}\right) = \left(\sum_{j=1}^{n} \omega_{1j}^{(1)}, \ldots, \sum_{j=1}^{n} \omega_{nj}^{(1)} \right) = V^{(1)}.$$

From the iteration $m \geq 1$ we perform the following operations:

$$\left.\begin{aligned} &\overline{W}^{(m)} = W^{(m-1)} W^{(m-1)} \\ &s\left(\overline{W}^{(m+1)}\right) = \sum_{i,j=1}^{n} \overline{\omega_{ij}}^{(m+1)} \end{aligned}\right\} \Rightarrow g\left(W^{(m-1)}\right) = \frac{\overline{W}^{(m)}}{s\left(\overline{W}^{(m)}\right)} = W^{(m)}$$

$$h\left(W^{(m)}\right) = \left(\sum_{j=1}^{n} \omega_{1j}^{(m)}, \ldots, \sum_{j=1}^{n} \omega_{nj}^{(m)} \right) = V^{(m+1)}.$$

The result of the algorithm will be called V^{∞} if it reaches the stop condition $\left(\left|\left|V^{(m)} - V^{(m-1)}\right|\right| < \varepsilon, \quad \forall \varepsilon > 0\right)$.

References

[1] Charles M. Grintead and J. Laurie Snell *Introduction to Probability.* 2006. 2nd edition.

[2] Federación de Lucha Canaria. *Reglamento.*