

05. Upper Confidence Bound

Publicado el 19 mayo, 2020 por DaneriCoding

Aprendizaje Por Refuerzo

Comenzamos una nueva sección para estudiar dos técnicas de uno de las ramas más prolíficas del Machine Learning. Se trata del Aprendizaje Por Refuerzo (Reinforcement Learning), denominado también como Online Learning.

El Aprendizaje Por Refuerzo (Reinforcement Learning) generalmente se utiliza para resolver problemas donde exista una interacción entre la máquina y un determinado suceso. Los datos observados hasta un determinado momento T se utilizarán para decidir una determinada acción de respuesta a implementar en el momento $T + 1$.

De esta manera seremos capaces de entrenar máquinas que puedan desempeñar tareas como Caminar o Conducir.

Utilizaremos las teorías del Condicionamiento Operante de tal manera que ofrecemos una *recompensa* a nuestra IA cuando proporciona un resultado esperado, e impartimos un *castigo* a nuestra IA cuando proporciona un resultado no esperado.

De esta manera nuestras Máquinas aprenden a través de Ensayo-Error, o más bien aprenden a través de las tuplas Ensayo-Recompensa, Ensayo-Castigo. Estudiaremos ahora las dos técnicas de Aprendizaje por Refuerzo (Reinforcement Learning) con mayor predicamento:

- Upper Confidence Bound (*Límite de Confianza Superior*).
- Muestreo Thompson.

El Problema Del Bandido Multi-Brazo

Como problema canónico del mundo del Machine Learning abordaremos El Problema del Bandido Multi-Brazo mediante un ejemplo. Resultará un ejemplo muy genérico en relación a lo que el Aprendizaje Por Refuerzo puede conseguir, si bien no es el único tipo de problema que podemos abordar mediante Reinforcement Learning.

Gracias al Aprendizaje Por Refuerzo un perro robot puede aprender a caminar adecuadamente. Como ya hemos dicho aplicaremos los principios del Condicionamiento Operante bien estudiado en Psicología.

Esencialmente para enseñar a un perro robot a caminar le voy dando las acciones que este puede tomar y posteriormente le enseñaré cuando una determina acción resulta positiva (recompensa) y conviene que se repita, y cuando una determinada acción resulta negativa (castigo) y conviene que no se repita.

Generalmente una recompensa se identifica con un $+1$. Nuestro Algoritmo intentará maximizar un determinado conteo. Los castigos resultan abstraídos mediante un 0 o un -1 . Los Algoritmos de Aprendizaje Por Refuerzo suponen una aproximación muy interesante a lo que consideramos como Inteligencia.

El Aprendizaje Por Refuerzo es una técnica de Inteligencia Artificial.

Sirva como imagen sugerente la del siguiente perro robot, para imaginarnos las potencialidades que presentarán nuestro Algoritmos de Aprendizaje Por Refuerzo:



Podemos imaginar nuestro Problema Del Bandido Multi-Brazo como un ladrón que cuenta con una banda organizada en el que cada miembro se puede entender como uno de sus brazos. O de manera más sencilla, podemos conceptualizar nuestro Problema Del Bandido Multi-Brazo como una Tragaperras.

A las Tragaperras históricamente se les llamaban Bandidos Multi-Brazo.

Antiguamente resultaba necesario tirar de una palanca para accionar una Tragaperras. Conviene señalar que una Tragaperras es el juego que más rápidamente arruina a los jugadores de un Casino. En pro del imaginario incluimos a continuación una de las Tragaperras de iconografía histórica:

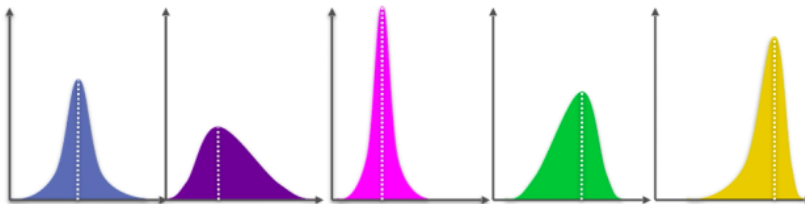


Se les llamaba Bandido por esa práctica seguridad que presentaban de que perdieras tu dinero. Se les llamaba Multi-Brazo porque en los Casinos se solían presentar múltiples máquinas dispuestas de manera contigua, cada una de las cuales presentaban una Probabilidad distinta de ganar:



Abordar el Problema del Bandido Multi-Brazo permitía escoger la mejor tragaperras de un determinado Casino en función de las distintas Probabilidades que podían presentar cada una de ellas de ganar o perder. El Problema consiste esencialmente en descubrir la Distribución de Premios que presenta cada máquina.

Cada una de estas máquinas presentará una determinada Distribución de Probabilidad en cuanto a dar premios, que podemos suponer de la siguiente forma:



Conociendo estas Distribuciones directamente elegiríamos jugar en la quinta máquina, que es la que presenta una Distribución de Probabilidad para los premios de mayor cuantía (más a la derecha).

Para abordar esta problemática debemos buscar los Límites de Confianza del Jugador. Exploraremos las distintas máquinas intentando no sobrepasar dicho Límite de Confianza: recordad que la Distribución de Probabilidad que vemos más arriba no la conocemos a priori.

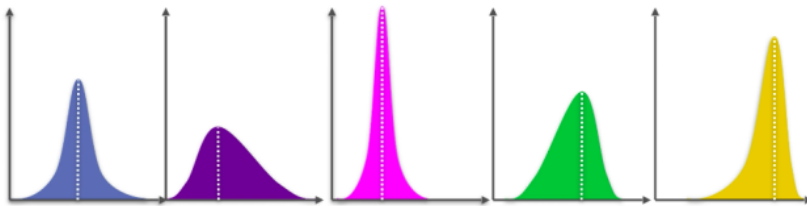
Tenemos como objetivo encontrar la mejor Distribución pasando el menor tiempo posible jugando a cada una de las máquinas. En este sentido el factor Tiempo resultará relevante para nuestro Algoritmo.

Veamos ahora un ejemplo real del mundo del Marketing que encapsula El Problema Del Bandido Multi-Brazo. Supongamos que Coca-Cola desea lanzar una campaña con el eslogan “*Bienvenido al lado Coca-Cola de la vida*”. Para ello se realizan diferentes anuncios de los cuales obtendremos una serie de datos en un Estudio de Mercado.

Se proponen cinco imágenes diferentes a tal respecto, de la siguiente manera:



Buscaremos aquella imagen que conforma una mayor Conversión en términos de Marketing, es decir, aquellos usuario que trasladan su fidelidad a una determinada marca en base a una Campaña Publicitaria. Explorando encontramos una Distribución similar (igual) a la que presentaban anteriormente las máquinas Tragaperras:



Volveríamos a elegir en este caso la Campaña asociada a la quinta Distribución. El desafío consiste en conformar estas Distribuciones en el menor Tiempo posible, que generalmente supondrá el menor Coste posible.

Estudiaremos cómo resolver esta problemática mediante las distintas técnicas de Aprendizaje Por Refuerzo que estudiaremos en las próximas lecciones.

Upper Confidence Bound (UCB)

Estudiaremos ahora el Algoritmo del Límite de Confianza Superior (Upper Confidence Bound) que se enmarca en la rama del Aprendizaje Por Refuerzo. Abordaremos mediante este Algoritmo el Problema del Bandido Multi-Brazo.

Actualmente una aplicación muy extendida de este Algoritmo se encuentra dentro del mundo de la Publicidad, en la que se pretende conocer la mejor Distribución de Probabilidad de entre un número determinado de anuncios.

Esencialmente este Algoritmo presenta los siguientes pasos:

1. Inicialmente contamos con D Brazos. Estos *Brazos* pueden ser anuncios que son mostrados a los usuarios de una página web cuando se conectan a ella.
2. Ejecutamos una Ronda para cada conexión que realiza un usuario a dicha página web.

3. Para cada Ronda N se elige uno de los anuncios que será mostrado al usuario.
4. Para cada Ronda N se genera una Recompensa que será de 1 en caso de que el usuario haga click en el anuncio, y que será de 0 en caso de que el usuario no haga click en el anuncio.
5. Para el número de Rondas que elijamos ejecutar debemos conseguir maximizar la Recompensa.

Si bien estos son los pasos del Algoritmo genérico, para implementar un Algoritmo de Límite Superior de Confianza debemos implementar los siguientes pasos:

1. Para cada Ronda N se consideran dos métricas bien diferenciadas:
 - A. Cantidad de veces que se selecciona el anuncio en dicha Ronda N.
 - B. Suma de Recompensas acumuladas del anuncio hasta dicha Ronda N.
2. Partiendo de las métricas anteriores calculamos otras dos métricas bien diferenciadas:
 - A. Recompensa Media del anuncio hasta dicha Ronda N, que presenta la forma:

$$\bar{r}_i(n) = \frac{R_i(n)}{N_i(n)}$$

- B. Intervalo de Confianza presente en la Ronda N, que presenta la forma:

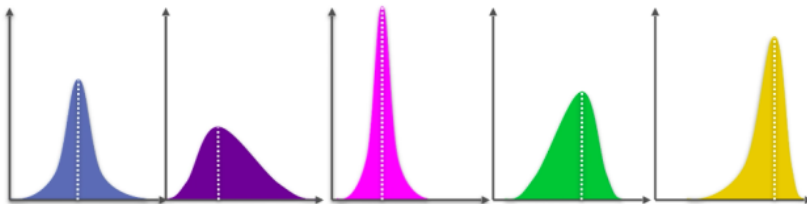
$$(\bar{r}_i(n) - \Delta_i(n), \bar{r}_i(n) + \Delta_i(n)) \quad \text{con} \quad \Delta_i(n) = \sqrt{\frac{3 \log(n)}{2N_i(n)}}$$

3. Seleccionamos el anuncio que presente un mayor Límite Superior para el Intervalo de Confianza calculado.

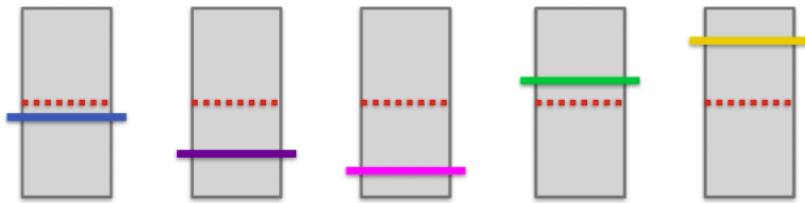
No entraremos en el este caso en los pormenores del desarrollo matemático. Basta con saber que serán de aplicación las fórmulas superiores y que permitirán calcular ese Límite de Confianza Superior.

El objetivo del desarrollo matemático anterior implica conocer la Distribución de Probabilidad presumible para cada Brazo utilizando el menor número de Rondas N posible, de tal manera que minimicemos el coste, ya sea Coste Temporal o Monetario.

Las Distribuciones de Probabilidad que conocemos de antemano para ejemplificar en este caso el Algoritmo, que ya vimos anteriormente, es la siguiente:



Para mayor conveniencia con nuestro Algoritmo pasemos a mostrar estas distribuciones en Vertical en lugar de en Horizontal, de tal manera que tenemos:

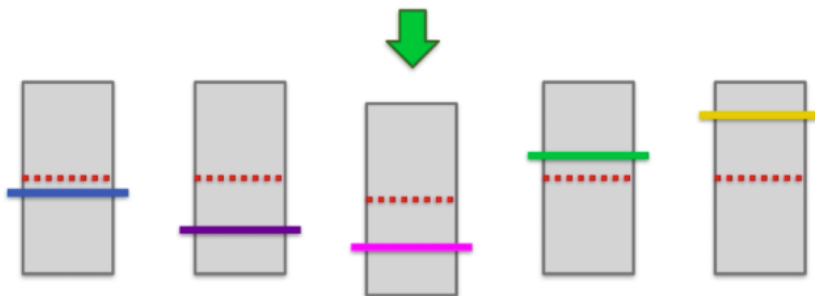


Las líneas en horizontal de cada uno de los colores representan el Valor Esperado como Valor Medio para cada una de las Distribuciones. Las líneas en horizontal discontinuas de color rojo son el valor supuesto que tenemos de inicio, en el que lógicamente partimos de un mismo valor medio supuesto para todos los casos.

Mediante sucesivas N Rondas de prueba el valor medio supuesto teórico inicial se irá moviéndose de tal manera que se va ajustando progresivamente tendiendo a la Distribución de Probabilidad de cada caso.

El tamaño de cada rectángulo representa el Intervalo de Confianza que se presenta para cada Ronda N en relación a que la Distribución de Probabilidad coincida con la que realmente presenta cada caso.

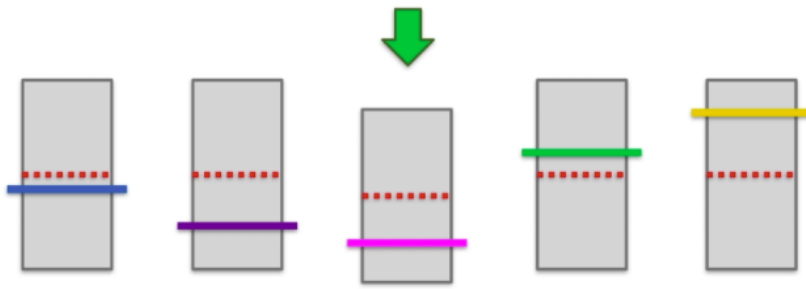
Supongamos que realizamos una primera Ronda en el ejemplo de los anuncios y que nuestro usuario no hace click sobre él. En este caso la Media aplicada a dicho anuncio disminuiría ligeramente, de tal manera que:



El Intervalo de Confianza (Tamaño del Rectángulo) de momento sigue siendo del mismo tamaño debido a que de momento tan solo ostento una Observación. La Amplitud del Intervalo de Confianza irá disminuyendo conforme vayamos realizando Observaciones.

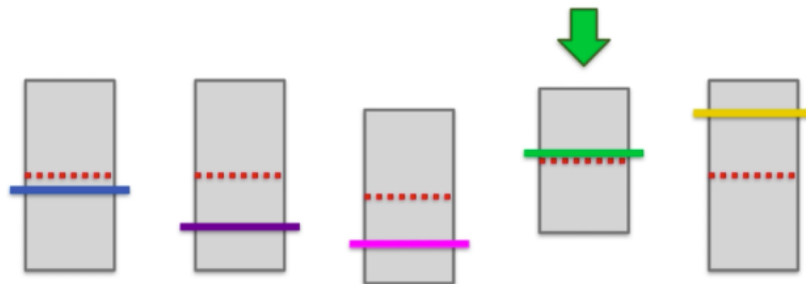
Cuántas más Observaciones efectuemos más seguros estaremos sobre la Convergencia de una determinada Distribución de Probabilidad, y más acotado resultará por tanto el Intervalo de Confianza.

Suponiendo más Observaciones obtendríamos una acotación para dicho Intervalo de Confianza que visualmente podemos representar de la siguiente manera:

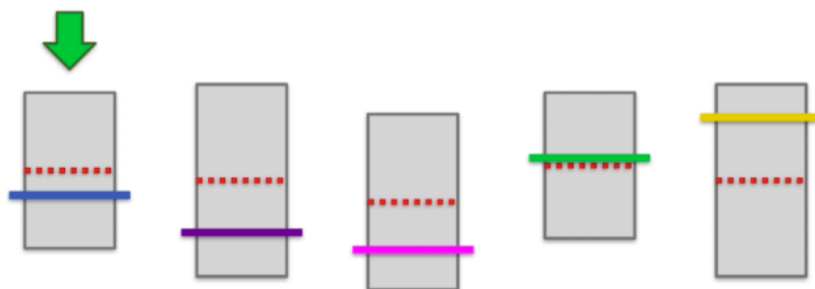


Observamos una pequeña disminución en el tamaño del Rectángulo que nos representa aquí el Intervalo de Confianza. El proceso de Convergencia en el estudio de la Distribución de Probabilidad generalmente resultará bastante lento.

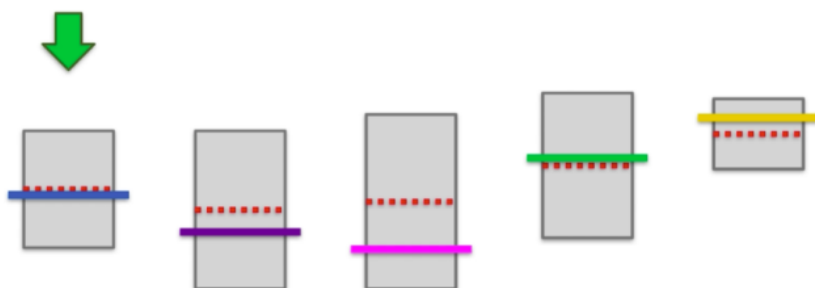
Suponemos ahora que mostramos otro anuncio, correspondiente a la posición cuarta, de tal manera que se aproxima la Media y se acota el Intervalo de Confianza, obteniendo en este caso un resultado tal que:



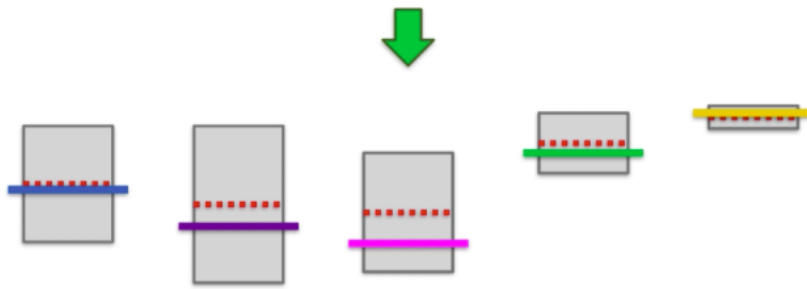
Elegimos un nuevo anuncio y se lo mostramos a nuestros usuarios, volviendo a evaluar su resultado en cuanto a la Media y al Intervalo de Confianza asociado. Obtendríamos en este caso:



Seguimos realizando estas operaciones generando Rondas sucesivas. Tras unas cuantas iteraciones más obtenemos el siguiente resultado generalizado:



Convendría que generalmente fuéramos iterando principalmente sobre el ítem que presenta el Límite de Confianza Superior, aunque no debemos iterar exclusivamente de esta manera dado que podríamos obtener resultados subóptimos. Tras otras cuantas iteraciones más obtendríamos el siguiente resultado:



Parece ser que tras unos cientos de iteraciones efectivamente nuestro Algoritmo converge marcando como Límite de Confianza Superior aquel que en este caso, al ser un ejemplo, sabíamos que iba a presentarlo.

Esta aproximación mediante Intervalos de Confianza resulta ser una solución bastante elegante para afrontar problemas reales en los que siempre se va a presentar un margen de duda.

Obtener Conjunto De Datos

Como ya comentamos en su momento podemos descargar nuestros DataSet de manera global clonando el Repositorio de GitHub de Juan Gabriel, o descargar cada uno de nuestros DataSet de manera independiente.

Igualmente podemos realizar una copia en nuestro Repositorio GitHub mediante un *Fork* del contenido. Para descargas sección a sección el enlace nos dirigirá a la web del equipo SuperDataScience, que se presentan perfectamente organizados.

De un modo u otro podemos disponer de todo el material desde buen inicio.

Upper Confidence Bound en Python – Paso 1: Importar DataSet

Comenzamos aquí la parte práctica de esta nueva rama del Machine Learning que venimos estudiando en esta sección denominada Aprendizaje Por Refuerzo (Reinforcement Learning).

Nos estamos acercando de esta manera al campo más amplio de la Inteligencia Artificial, dado que los Robots se construyen en general mediante Reinforcement Learning. Generamos un nuevo script denominado *upper_confidence_bound*.

Ejecutamos como siempre nuestro script para establecer adecuadamente nuestra carpeta de trabajo en *File Explorer*. Utilizaremos en este caso un DataSet relacionado con el mundo del Marketing denominado *Ads_CTR_Optimisation*.

Comenzamos nuestro código importando las librerías necesarias:

```
# Upper Confidence Bound (UCB)

# Importar Librerías
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Cargar DataSet
dataset = pd.read_csv("Ads_CTR_Optimisation.csv")
```

Cargamos igualmente el DataSet que vamos a estar utilizando. Explorando el DataSet encontramos que tenemos un conjunto de diez (10) anuncios y la información sobre los datos de Refuerzo aplicados para cada uno de dichos anuncios, correspondiéndose el uno (1) con una suerte de Recompensa y el cero (0) con una suerte de Castigo.

Index	Ad 1	Ad 2	Ad 3	Ad 4	Ad 5	Ad 6	Ad 7	Ad 8	Ad 9	Ad 10
0	1	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0
7	1	1	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	1	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

A efectos de imaginario podemos suponer que se trata de la campaña de publicidad de Coca-Cola que comentamos en las clases de Teoría. Se disponen diez (10) anuncios diferenciados que publicitamos en una Red Social para recabar los datos pertinentes.

La empresa de Marketing nos contrata a nosotros como Científicos de Datos para que, conforme a los datos extraídos, logremos evaluar qué anuncio de los diez (10) existentes es el que presenta mejor Tasa de Conversión de usuarios en clientes.

En este caso dispones únicamente de Variables Independientes que necesitamos convertir en Variables que un Algoritmo como Upper Confidence Bound sea capaz de evaluar. Construiremos nuestro Intervalo de Confianza alrededor de la Media de Click para cada uno de los anuncios.

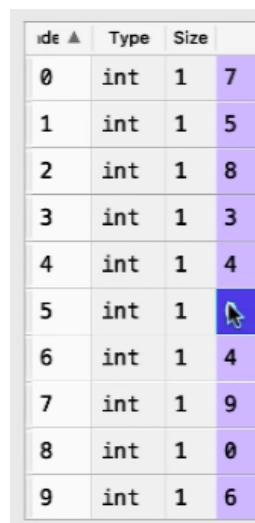
Existirá una estrategia previamente diseñada de tal manera que los resultados obtenidos en una Ronda influyan en qué se muestra en la siguiente Ronda.

Se trata por tanto de una estrategia dinámica que se va adaptando a los diferentes datos que vamos progresivamente recabando. Conforme los distintos anuncios reciban Recompensas por parte del Algoritmo se irán mostrando más asiduamente a los usuarios dado que disminuirá su Intervalo de Confianza en Límite Superior.

Nuestro DataSet se ha conformado con 10.000 Rondas de Anuncios. Evaluando las Recompensas obtenidas construiremos el Intervalo de Confianza para nuestro DataSet. Para ir alimentando nuestro Algoritmo realizo una selección aleatoria para ir conformando la acumulación de Recompensas, con el siguiente código:

```
# Implementar Selección Aleatoria
import random
N = 10000
d = 10
ads_selected = []
total_reward = 0
for n in range(0, N):
    ad = random.randrange(d)
    ads_selected.append(ad)
    reward = dataset.values[n, ad]
    total_reward = total_reward + reward
```

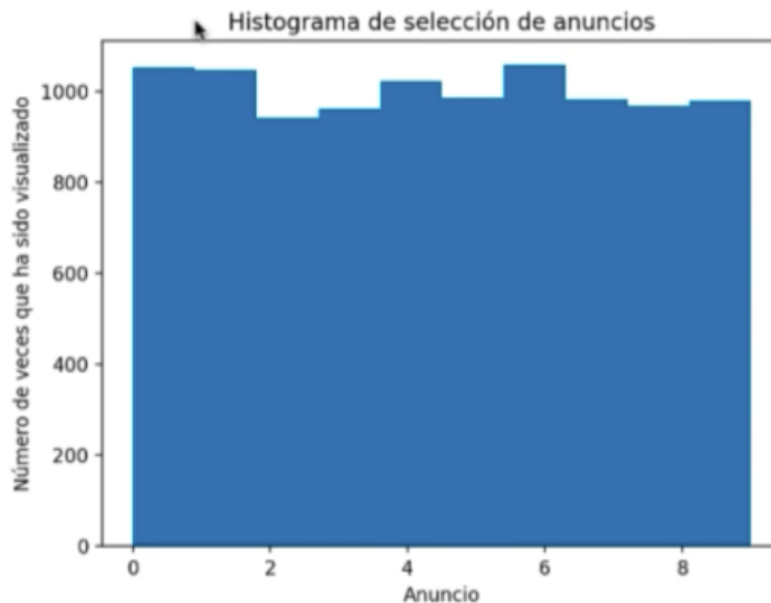
Mediante el código anterior obtenemos una Recompensa de 1.242, extremo que significa que de los 10.000 usuarios a los que he mostrado un anuncio, 1.242 de ellos han clicado sobre dichos anuncios. En la Variable *ads_selected* podemos observar los distintos anuncios seleccionados para cada iteración:



idx ▲	Type	Size	
0	int	1	7
1	int	1	5
2	int	1	8
3	int	1	3
4	int	1	4
5	int	1	4
6	int	1	4
7	int	1	9
8	int	1	0
9	int	1	6

Cuando utilizamos factores aleatorios para mostrar la información obtenemos resultados diferentes para cada una de las ejecuciones del código. El valor por tanto de la Recompensa irá cambiando conforme ejecutemos nuestro código. En cualquier caso los valores resultarán próximos entre sí.

Convendría en este punto implementar un Histograma de Frecuencias para visualizar qué hemos implementado exactamente. Para ello conformamos el siguiente código:



Podemos observar que todos los anuncios se han mostrado aproximadamente con la misma frecuencia. Es lo que cabe esperar para una reproducción aleatoria.

La mejora de nuestro Algoritmos mediante Recompensas y Castigos sucesivos implicará que cambiará la frecuencia con que se muestra alguno de los anuncios, despuntando sobre el resto.

Upper Confidence Bound en Python – Paso 2: Implementar Algoritmo

En el paso anterior además de importar nuestro DataSet efectuamos una aproximación aleatoria. Comenzaremos ahora la implementación de nuestro Algoritmo de Límite de Confianza Superior para comprobar que ofrece mejores resultados que una aproximación aleatoria.

Decir que no tenemos ningún paquete de Python que implemente automáticamente el Algoritmo de Límite de Confianza Superior. Aprenderemos por tanto a implementar el Algoritmo completo por nuestra cuenta. Implicarán más líneas de código de las que venimos estando acostumbrados.

Recordemos que para el primer paso del Algoritmo necesitamos considerar por un lado el número de veces que se selecciona un determinado anuncio hasta una determinada Ronda, y por otro lado el sumatorio de Recompensas que presenta dicho anuncio considerando hasta dicha determinada Ronda. Esto atiende a la fórmula:

$$\bar{r}_i(n) = \frac{R_i(n)}{N_i(n)}$$

Comenzamos implementando primeramente el número de veces que resulta seleccionado un determinado anuncio hasta la Ronda N. Para ello implementamos un Vector que contendrá dicho número de veces. Hacemos:

```

# Algoritmo UCB
import math
N = 10000
d = 10
number_of_selections = [0] * d
sums_of_rewards = [0] * d
ads_selected = []
for n in range(0, N):
    max_upper_bound = 0
    ad = 0
    for i in range(0, d):
        if(number_of_selections[i]>0:
            average_reward = sums_of_rewards[i] / number_of_selections[i]
            delta_i = math.sqrt(3/2*math.log(n+1)/number_of_selections[i])
            upper_bound = average_reward + delta_i
        else:
            upper_bound = 1e400
        if upper_bound > max_upper_bound:
            max_upper_bound = upper_bound
    ad = i

```

Como segundo término realizamos el Sumatorio de las Recompensas acumuladas hasta la Ronda N. Par ello implementamos otro Vector que contendrá dicho Sumatorio. Calculamos seguidamente la Recompensa Media así como el Intervalo de Confianza, extremos que podemos calcular perfectamente con los Vectores ya implementados.

Recordemos que el Intervalo de Confianza atiende a la fórmula:

$$(\bar{r}_i(n) - \Delta_i(n), \bar{r}_i(n) + \Delta_i(n)) \quad \text{con } \Delta_i(n) = \sqrt{\frac{3 \log(n)}{2N_i(n)}}$$

Implementamos dos bucles for anidados dado que debemos calcular cada una de las métricas para cada uno de los diez (10) anuncios existentes. Debemos recordar implementar inicialmente la librería *Math* para realizar las operaciones matemáticas pertinentes.

Mediante sumas, multiplicaciones, divisiones, logaritmos y un bucle hemos conseguido implementar un Algoritmo que permite seleccionar el mejor de los anuncios, sin requerir excesivas iteraciones, mediante una aproximación del Límite de Confianza Superior como mejor estrategia para manejar la incertidumbre.

Dado que los Índices en Python comienzan en cero (0) durante las diez (10) primeras Rondas todo nuestro Bucle permanecerá en cero (0).

Upper Confidence Bound en Python – Paso 3: Entrenar Algoritmo

En la clase anterior Juan Gabriel nos dejó como reto el entender por qué inicializabamos nuestro algoritmo con un *upper_bound* tan elevado como 10^400 . Para entenderlo veamos qué ocurre la primera vez que llamamos a nuestro algoritmo en la primera Ronda.

Para una primera iteración de nuestro Algoritmo el anuncio número cero (0) resultará marcado con un *upper_bound* de 10^400 . Para esta primera Ronda queremos que cuando un anuncio entra en el bucle y calculamos su *upper_bound* me interesará que todos los anuncios resulten marcados con un Límite Superior bastante alto.

Para una primera Ronda N me interesará que todos los anuncios queden seleccionados al menos una vez como el de Límite Superior.

Esta marcación de todos los anuncios como de Limite Superior permitirá al Algoritmo iterar en sucesivas Rondas en igualdad de condiciones para todos esos anuncios, que en primera instancia es lo que nos interesa. Añadimos en este punto a nuestro código el anuncio que haya sido elegido como mejor iteración en cada momento:

```
# Algoritmo UCB
import math
N = 10000
d = 10
number_of_selections = [0] * d
sums_of_rewards = [0] * d
ads_selected = []
total_reward = 0
for n in range(0, N):
    max_upper_bound = 0
    ad = 0
    for i in range (0, d)
    if(number_of_selections[i]>0:
        average_reward = sums_of_rewards[i] / number_of_selections[i]
        delta_i = math.sqrt(3/2*math.log(n+1)/number_of_selections[i])
        upper_bound = average_reward + delta_i
    else:
        upper_bound = 1e400
    if upper_bound > max_upper_bound:
        max_upper_bound = upper_bound
    ad = i
    ads_selected.append(ad)
    number_of_selections(ad) = number_of_selections(ad) + 1
    reward = dataset.values[n, ad]
    sums_of_rewards[ad] = sums_of_rewards[ad] + reward
    total_reward = total_reward + reward
```

Actualizamos igualmente el Número de Selecciones y la Suma de Recompensas. Con lo anterior hemos construido convenientemente nuestro Algoritmo de manera completa.

Si bien son más líneas de código de las que estamos acostumbrados, vemos que tampoco resulta excesivamente complejo implementar un Algoritmo completo sin ayuda de una librería de terceros.

Veamos ahora los resultados que nos arroja nuestro Algoritmo de Upper Confidence Bound, que eventualmente queremos comparar con el Algoritmo Aleatorio y las Recompensas que nos arrojaba este.

Una vez ejecutado obtenemos una Recompensa Total de 2.178, que resulta netamente superior a las 1.242 que presentaba nuestro Algoritmo Aleatorio inicial. Esto significa que los anuncios que se muestran a los usuarios están mucho más optimizados.

Mediante el Algoritmo de Upper Confidence Bound obtendríamos el doble de Recompensas en una Tragaperras que mediante un abordaje puramente aleatorio.

Cabe preguntarnos en este punto cual ha sido el mejor anuncio a mostrar a nuestros usuarios. Conforme avanzan las sucesivas Rondas nuestro Algoritmo va convergiendo y se empiezan a mostrar mucho más algunos anuncios que otros.

Estudiando la Variable `ads_selected` observaremos que el anuncio que presenta mejor tasa de Conversión de usuarios en clientes es el anuncio número cinco (5). Con esta Convergencia estamos consiguiendo duplicar el número de ventas que obtendríamos de una manera puramente aleatoria.

Upper Confidence Bound en Python – Paso 4: Visualización

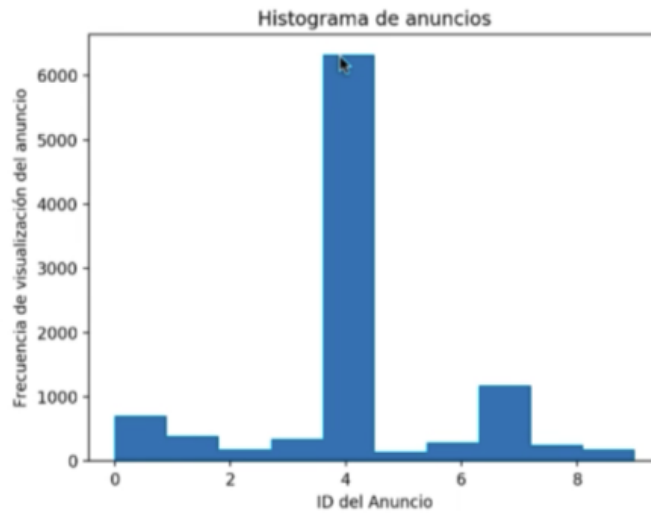
Recordemos que en nuestra clase anterior habíamos conformado nuestro Algoritmo de Límite de Confianza Superior (Upper Confidence Bound) para obtener el anuncio óptimo de entre todos los posibles.

Esta implementación ha permitido dos cosas. Por un lado obtener un resultado con el doble de Recompensas que si hubiéramos realizado un abordaje aleatorio. Por otro lado obtuvimos al anuncio cinco (5) como el anuncio óptimo que debería ser mostrado.

Elaboremos en este caso un Histograma de Resultados para poder observar convenientemente cada uno de los distintos anuncios. Hacemos:

```
# Histograma de Resultados
plt.hist(ads_selected)
plt.title("Histograma de Anuncios")
plt.xlabel("ID del Anuncio")
plt.ylabel("Frecuencia de Visualización del Anuncio")
plt.show()
```

Aplicamos el método `hist` al Vector de Anuncios. Ejecutando el código anterior obtenemos como resultado:



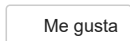
Vemos claramente como el anuncio número cinco (5), en la posición cuatro (4) proporcionada por Python, es el que presenta mayor tasa de Conversión, y por tanto es al anuncio al que Converte nuestro Algoritmo otorgando mayor número de visualizaciones.

Acabamos de implementar nuestro primer Algoritmo de Aprendizaje Por Refuerzo (Reinforcement Learning).

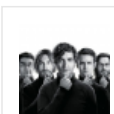
Upper Confidence Bound en R – Paso 1: ...

... [Resta por implementar el Modelo de Upper Confidence Bound con tecnología R] ...

Comparte esto:



Sé el primero en decir que te gusta.



Acerca de DaneriCoding

"La tecnología puede ser la fuente de la Trascendencia" Steve Jobs

[Ver todas las entradas por DaneriCoding →](#)

Esta entrada fue publicada en [MACHINE LEARNING \(II\)](#). Guarda el [enlace permanente](#).

Ciencias Apple

Crea un blog o un sitio web gratuitos con WordPress.com.

