UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Estudios Superiores
Acatlán

Practica 1:

Upper Confidence Bound y muestreo de Thompson

Aprendizaje de maquina y minería de datos avanzados

Integrantes:

Gustavo Adolfo Alvarez Hernández

Profesor:

Eduardo Eloy Loza Pacheco

FES Acatlán, 7 de febrero 2023

Objetivo

Que el alumno aprenda los fundamentos del aprendizaje por refuerzo por medio de dos algoritmos de aprendizaje por refuerzo sobre la elección de campañas publicitarias en búsqueda de tomar las campañas más vistas.

Resumen

En la presente práctica se abarca la problemática de selección de anuncios con mayores clics de los usuarios, buscando así en cada etapa seleccionar el anuncio que nos de mayor resultado. Por lo cual se implementan tres aproximaciones.

1. Tomar al azar entre los 10 anuncios diferentes que se tienen a lo largo de todos los usuarios.
2. Implementar el algoritmo de Upper Confidence Bound (UCB)
3. Implementar el algoritmo de muestreo de Thompson

En general las últimas dos aproximaciones tienen un desempeño y selección de anuncios bastante similar, dando una recompensa promedio de 1288, por su parte al azar queda por debajo de este valor y su distribución de selección es uniforme, por lo cual obtuvo 1117 de recompensa.

Antecedentes

En esta sección abarcaremos brevemente el funcionamiento de los dos algoritmos para tener una mejor idea de la implementación y los resultados.

El algoritmo UCB, se basa en la selección de la opción o camino que traiga el mayor beneficio, lo cual se parece a un algoritmo voraz, pero se añade un elemento de incertidumbre que se hace más grande en medida que no se ha explorado cierta opción, por lo cual la elección de que decisión tomar se basa en la siguiente formula:

$$A_t = \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \right]$$

Donde:

- $Q_t(a)$ es el valor estimado de tomar la acción a en el tiempo t
- $N_t(a)$ es el número de veces que a ha sido selecciona en el tiempo t
- c es el valor de confianza que determina el nivel de exploración

Analizando la formula véase que c es un hiperparámetro que nos dirá que tanto peso tiene el no haber explorado a en ese tiempo, lo cual hará que sea seleccionada esa opción y así evitamos que solo tome una opción a lo largo del tiempo. En cuanto al valor estimado suele ser la recompensa

obtenida entre el número de veces que se escogió esa opción en ese tiempo.
Por su parte, el muestro de Thompson tiene un enfoque más estadístico, en el cual la recompensa no está ponderada, esto es solo acepta casos de se recibe recompensa o no, o también llamadas recompensas binarias. Por lo cual se basa en seleccionar el siguiente paso por medio de una distribución probabilística, que en este caso será una $Beta(\alpha, \beta)$.

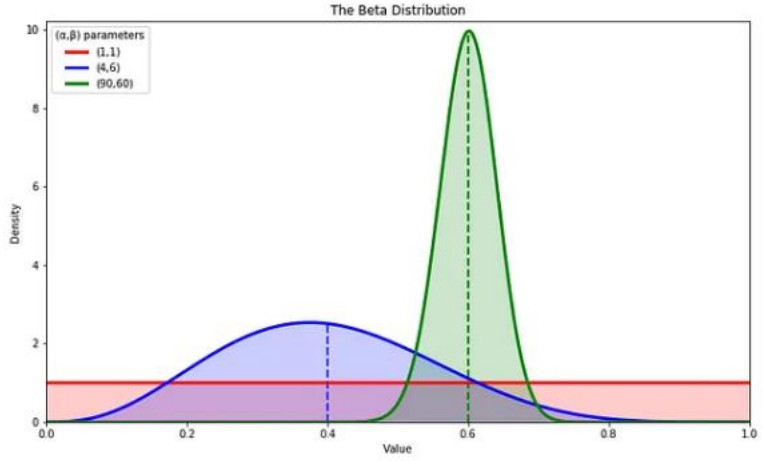
Esto dado que sus parámetros pueden ser considerados como el número de sucesos éxitos (Se realizo la acción y se obtuvo resultado) como α y el numero de fallos como β. Además de que la media de la distribución estará dada por:

$$\frac{\alpha}{\alpha + \beta} = \frac{\text{numero de exitos}}{\text{Total de intentos}}$$

Este algoritmo parte de la idea de que en medida que tenemos más información sobre las decisiones que tomamos, podemos saber que tan factible es tomar dicha opción. Esto introduce los conceptos de probabilidad a priori y a posteriori.

La probabilidad a priori es la probabilidad antes de haberse realizado el experimento, en nuestro caso al iniciar no sabemos cuál es mejor opción, por lo cual todas las acciones son igual de buenas y se considera una $beta(1,1)$ ya que se comporta como una uniforme y todas las opciones son igual de probables a escoger.

Por su parte, la probabilidad a posteriori está dada por la información dada después de realizar el experimento, esto es dado que ya tomé tal acción que tan bien o mal me fue, por lo cual los valores de los parámetros se irán actualizando y haciendo que la distribución se cargue hacia las opciones que traen mejor resultado, o lo mismo que con una α más grande, lo cual es más evidente en la siguiente gráfica.



Con estas distribuciones podemos ver que en al inicio no tenemos certeza sobre cuál es la mejor opción, pero en medida que vamos explorando las opciones la distribución empieza a reducir el número de posibilidades hasta decantarse por unas cuantas.

Desarrollo

1. Importar los módulos necesarios

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random
import math
```

2. Cargamos los datos sobre los anuncios

```
data=pd.read_csv("Ads_CTR_Optimisation.csv")
```

3. Procedemos a generar el enfoque aleatorio que consiste en seleccionar al azar los anuncios y se suma el valor del anuncio

```
N=10000
d=10
anuncios_seleccionados=[]
recompensa_total=0
for n in range(0,N):
    anuncio=random.randrange(d)
    anuncios_seleccionados.append(anuncio)
    recompensa=data.values[n,anuncio]
    recompensa_total=recompensa_total+recompensa
print(recompensa_total)
```

```
1171
```

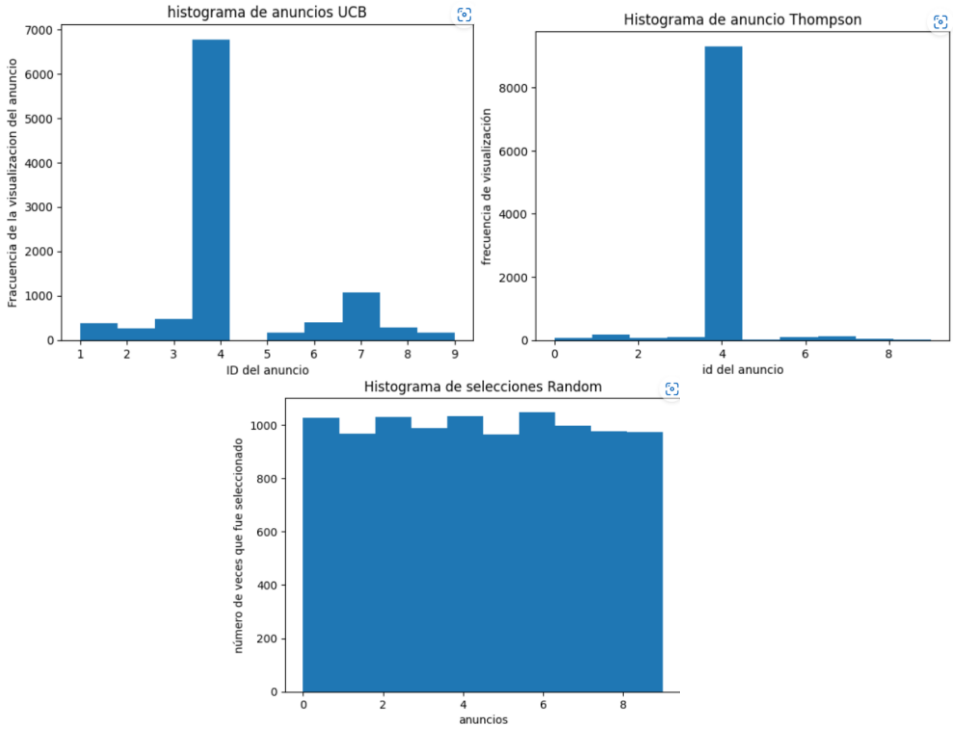
4. Para UCB generamos listas para mapear la recompensa del anuncio y las veces que se selecciona. Al inicio se seleccionan todos los anuncios y ya posteriormente se calcula A_t y se compara con la máxima banda obtenida, si se supera se selecciona ese anuncio.

```
N=10000
d=10
anuncio=0
numeroDeSelecciones=[0]*d
sumaDeLasRecompensas=[0]*d
anuncios_seleccionados=[]
recompensaTotal=0
for n in range(0,N):
    maximo_intervalo_superior=0;
    for i in range(0,d):
        if numeroDeSelecciones[i]>0:
            recompensamedia=sumaDeLasRecompensas[i]/numeroDeSelecciones[i]
            delta_i=math.sqrt(3/2*math.log(n+1)/numeroDeSelecciones[i])
            intervalodeconfianzasuperior=recompensamedia+delta_i
        else:
            intervalodeconfianzasuperior= i*400*10#ie400
        if intervalodeconfianzasuperior>maximo_intervalo_superior:
            maximo_intervalo_superior=intervalodeconfianzasuperior
            anuncio=i
    anuncios_seleccionados.append(anuncio)
    numeroDeSelecciones[anuncio]=numeroDeSelecciones[anuncio]+1
    recompensa=data.values[n,anuncio]
    sumaDeLasRecompensas[anuncio]=sumaDeLasRecompensas[anuncio]+recompensa
    recompensaTotal=recompensaTotal+recompensa
```

5. Para el muestreo de Thompson, procedemos a generar listas para ir guardando los valores de éxitos y fracasos e ir calculando la $beta(\alpha, \beta)$ correspondiente y así tomar el anuncio que arroje la distribución.

```
N=10000
d=10
numerodeRecompensas1=[0.000001]*d
numerodeRecompensas0=[0.000001]*d
anuncios_seleccionados=[]
recompensaTotal=0
for n in range(0,N):
    maximo_valor_aleatorio=0;
    anuncio=0
    for i in range(0,d):
        random_beta=random.betavariate(numerodeRecompensas1[i]+1,numerodeRecompensas0[i]+0)
        if random_beta>maximo_valor_aleatorio:
            maximo_valor_aleatorio=random_beta
            anuncio=i
    anuncios_seleccionados.append(anuncio)
    recompensa=data.values[n,anuncio]
    if recompensa==1:
        numerodeRecompensas1[anuncio]+=1
    else:
        numerodeRecompensas0[anuncio]+=1
    recompensaTotal+=recompensa
```

6. Con fin de ilustrar como se comporto cada algoritmo, se muestran los histogramas de decisión de anuncios de cada algoritmo



7. Y por último se contrasta las recompensas obtenidas por cada implementación.

```
Recompensa Random: 1117
Recompensa UBC: 1235
Recompensa Thompson: 1235
```

Conclusiones

Después de realizar las tres implementaciones podemos observar que aún la peor la implementación que es al azar no es tan mala aproximación debido que los anuncios no tienen comportamientos tan distintos entre ellos.

Por su parte los otros dos algoritmos de aprendizaje por refuerzo, aunque tienen metodologías de selección totalmente distintas y por ende distribución de selecciones distintas, las recompensas en varios intentos no necesariamente son iguales, pero si muy similares por lo cual cualquiera de las dos implementaciones suena suficientemente bueno.

Un punto para resaltar y que puede ser crucial a la hora de decidir entre estos dos algoritmos es que Thompson (al menos la implementación mostrada) no tiene peso en las recompensas, solo es binaria; por lo cual si su problema pondera las recompensas este algoritmo no sería útil.

Referencias

Roberts, S. (2 de Nov de 2020). *Towards Data Science*. Obtenido de Thompson Sampling: https://towardsdatascience.com/thompson-sampling-fc28817eacb8

Roberts, S. (26 de Oct de 2020). *Towards Data Science*. Obtenido de The Upper Confidence Bound (UCB) Bandit Algorithm: https://towardsdatascience.com/the-upper-confidence-bound-ucb-bandit-algorithm-c05c2bf4c13f

Zhou, Z.-H. (2016). *Machine Learning.* Singapore: Springer Nature Singapore.