Implementación de aprendizaje automático para la selección de mejores anuncios en páginas web

Ing. César Gastón Cárdenas Córdova

14 de junio de 2020

Índice

Índice de figuras

1. Introducción

El aprendizaje automático, aplicado a la selección óptima de anuncios en páginas web, es una buena alternativa para mejorar los ratios de conversión al momento de promocionar productos o campañas. Este hecho ha conllevado a que algunas compañías incluyan este tipo de servicios de optimización en su portafolio de soluciones, como por ejemplo Optimizely, Dynamic Yield y Frosmo. [1]

El aprendizaje por refuerzo (reinforcement learning) es la base principal para el desarrollo de esta tecnología. Este consiste, en términos generales, en la programación de agentes inteligentes que exploren las posibles acciones, para resolver un problema determinado, y registren (aprendan) las recompensas o penalidades obtenidas en cada una. En otras palabras, el sistema experimenta a través de prueba y error para alcanzar el mayor beneficio posible. [2]

En el presente trabajo, se propone el diseño de un servicio Restful básico, que implemente el algoritmo Upper Confidence Bound (UCB), para optimizar la selección de anuncios en un caso práctico definido. Asimismo, se muestra cómo este método aporta mejores beneficios que una solución tradicional, donde se seleccionan los anuncios de forma arbitraria. Aunque existen otros tipos de algoritmos que es posible aplicar; como por ejemplo: Thompson sampling, Epsilon greedy policy, Q learning, etc; escapa fuera del alcance la revisión de los mismos, debido a que no es parte del objetivo encontrar la mejor solución de aprendizaje por refuerzo que resuelva el problema.

Así pues, para lograr el objetivo planteado, se ha dividido el documento en 6 bloques. En primer lugar, se define un caso de negocio habitual que se quiere optimizar. En segundo lugar, se explica a grandes rasgos el problema del bandido multibrazo y las similitudes con el caso práctico planteado. Luego, se detalla cómo funciona el algoritmo UCB y los parámetros que utiliza para seleccionar los óptimos. Después de esto, se muestra el análisis para identificar los casos de uso y los componentes necesarios para la implementación. Finalmente, en el acápite siguiente, se presentan los resultados que ofrece el servicio revisando diferentes situaciones que pueden darse.

2. Caso práctico

Supongamos que una empresa, que cuenta con diez mil usuarios, desea promocionar la venta de un nuevo producto, colocando un anuncio al momento que el cliente ingresa al portal web de la compañía. Bajo este escenario pueden ocurrir dos posibles eventos:

1. El usuario se siente interesado por el anuncio y accede al mismo, haciendo clic en un botón.

2. El usuario no está interesado en el anuncio y lo cierra.

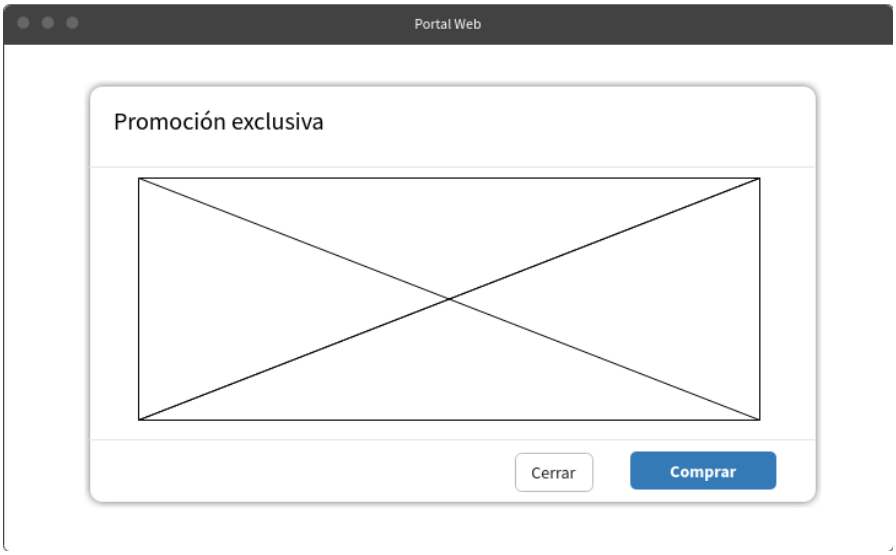

Fig. 1: Interfaz de ejemplo del anuncio en la página web

La figura 1 muestra un ejemplo de las opciones que se mostrarán al usuario. Asimismo, se han elaborado tres posibles anuncios para incentivar la compra del nuevo producto. Todos ellos se ven muy buenos y es complicado saber con certeza cual influirá más en el cliente. Debido a esto, es necesario que la parte del anuncio en la aplicación no sea una imagen estática, sino un componente que varíe cada vez que se ingresa al portal.

Por las características del caso, podría ser una alternativa emplear pruebas A/B [3]. Sin embargo, el dueño de la compañía necesita obtener ingresos lo más pronto posible, por lo que no puede darse el lujo de tomar un período de prueba para los anuncios. Debido a esta necesidad, el departamento de tecnología decide emplear técnicas de machine learning como alternativa de solución al problema.

3. Problema del bandido multibrazo

En un subcampo de la teoría de probabilidad estadística se plantea el problema del bandido multibrazo, que consiste en el siguiente enunciado:

"Supongamos que en un casino existe un número K de maquinas tragamonedas, donde la probabilidad de ganar en cada una de ellas es diferente y desconocida. ¿Con qué máquinas jugarías y en qué orden para máximizar la suma de recompensas obtenidas?" [4]

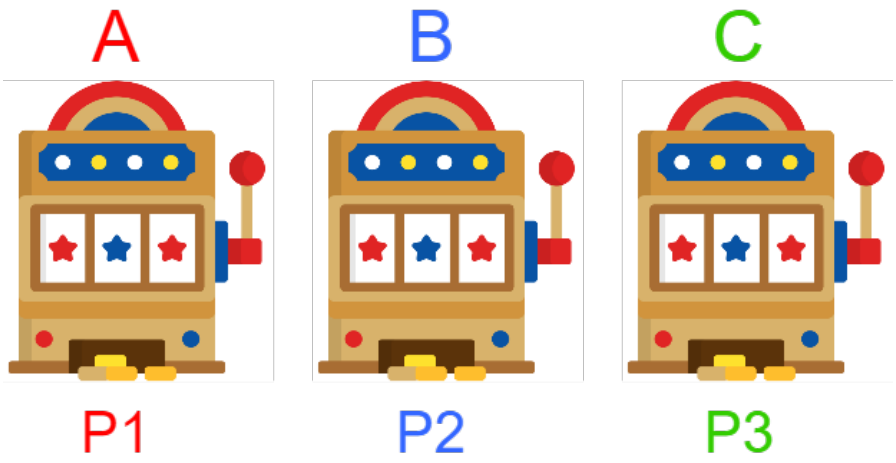

Fig. 2: Máquinas tragamonedas

Por ejemplo, si se tuvieran N fichas por jugar en las tres máquinas de la figura 2; cada una con probabilidad de éxito P_1, P_2 y P_3; para obtener una ganancia X. Lo primero que podríamos pensar, para resolver el problema, es utilizar una fracción del total de fichas disponibles para tratar de determinar el porcentaje de éxito de cada máquina (exploración) y, luego, apostar el resto en la que se obtengan mejores resultados (explotación).

Esta situación dilemática, entre cuánto explorar y explotar para maximizar la ganancia, puede afrontarse de diferentes formas. Una de ellas es el aprendizaje por refuerzo, campo de estudio de la inteligencia artificial, donde se proponen un conjunto de algoritmos que permitan a un agente inteligente determinar acciones que consigan la mayor recompensa posible, sin conocimiento previo del entorno. Bajo este modelo, las primeras decisiones tomadas por el agente tendrán cierto grado de aleatoriedad. Pero, en cada caso, se asignará una recompensa o penalidad; dependiendo de si fue una acción correcta o erronea, respectivamente. [5]

Como se puede apreciar, el problema descrito es muy similiar al caso práctico planteado en la sección anterior. Las máquinas tragamonedas estarían dadas por los anuncios que deseamos mostrar a los usuarios y cada una de las probabilidades de ganar el premio mayor

sería la probabilidad de que la persona acceda a comprar el producto cuando apostamos por mostrar un determinado anuncio.

4. Upper confidence bound

Upper confidence bound (UCB) o, en castellano, Límite de confianza superior es un algoritmo de aprendizaje por refuerzo determinista con un criterio de selección basado en el principio de optimismo frente a la incertidumbre. En otra palabras, esto significa que la política del algoritmo intentará explorar una opción menos conocida en cada elección, con el «optimismo» de que sea más beneficiosa que las opciones que ya conoce. Este proceso se repetirá hasta que el algoritmo determine cual de todas las que ha probado reiteradas veces ha tenido mayor éxito y descartará las demás. [6]

Por ejemplo, en la figura 3 se muestran los resultados que obtiene un agente al realizar elecciones con el algoritmo UCB en un bandido de 3 brazos. Cada marca roja indica que se obtuvo un resultado negativo, mientras que las marcas verdes representan recompensas. Los números asignados a cada decisión indican el orden en que fueron tomadas.

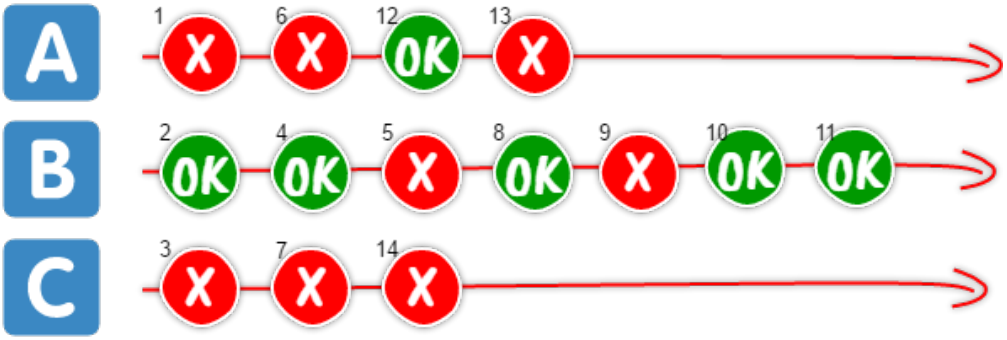

Fig. 3: Simulación de decisiones del UCB

Para entender cómo han ocurrido estás elecciones primero vamos a definir la matriz $n_{i,j}$, donde se van a contabilizar las veces en las que se elije cada opción i en el tiempo j, para $j \in [1, N]$. Dado que en el ejemplo tenemos 14 muestras de las decisiones del algoritmo, se tomará ese valor para N. La siguiente matriz muestra cómo incrementa el conteo de cada opción elegida, a lo largo de las 14 muestras:

$$n_{i,j} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 3 & 4 & 4 \\ 0 & 1 & 1 & 2 & 3 & 3 & 3 & 4 & 5 & 6 & 7 & 7 & 7 & 7 \\ 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 3 \end{bmatrix}$$

Una vez que se tiene cada opción contabilizada, es necesario identificar cual es el beneficio probable que ofrecen en base a lo que se va explorando. Para representar esto, utilizaremos una matriz $P_{i,j}$ que representa el porcentaje de veces que se ha obtenido recompensas en la opción i en el tiempo j. Por ejemplo, en la siguiente matriz podemos ver que $P_{2,5} = 0{,}67$

porque la alternativa B fue seleccionada 3 veces y se obtudo 2 recompensas ($\frac{2}{3} = 0{,}67$)

$$P_{i,j} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0{,}33 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0{,}67 & 0{,}67 & 0{,}67 & 0{,}75 & 0{,}6 & 0{,}67 & 0{,}71 & 0{,}71 & 0{,}71 & 0{,}71 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Si utilizaramos simplemente estas probabilidades, para determinar que decisiones tomar en cada caso, sería muy complicado saber cuando intentar explorar una nueva alternativa. Ya que, en las primeras elecciones $P_{2,j}$ obtiene altos porcentajes de recompensas y esto impediría que considere escoger A o C, basados unicamente en este criterio. Entonces, sería necesario que el algoritmo tenga un «incentivo» para explorar opciones nuevas cada momento. El UCB resuelve este inconveniento utilizando la siguiente fórmula para calular el indicador de selección:

$$\lambda_{i,j} = P_{i,j-1} + \sqrt{\frac{\gamma_{j-1}}{n_{i,j-1}}}$$

El valor de $\lambda_{i,j}$ representa un índice de beneficio que se obtiene al seleccionar la alternativa i en el tiempo j. Como se mencionó antes, $P_{i,j-1}$ sería la probabilidad de acierto en la elección anterior a j y $n_{i,j-1}$ la cantidad de veces que se elegió la alternativa i hasta el j anterior. De esta forma, al estar $n_{i,j-1}$ en el denominador de la función, el índice crecerá menos en la alternativa que se haya seleccionado mas veces que las otras. El parámetro $\gamma_{i,j}$ de la fórmula, es una función que crece a medida que se hacen más elecciones:

$$\gamma_j = 2 \ln j, \forall j \in [1, N]$$

$$\gamma_j = \begin{bmatrix} 0 & 1{,}39 & 2{,}20 & 2{,}78 & 3{,}22 & 3{,}58 & 3{,}89 & 4{,}16 & 4{,}39 & 4{,}61 & 4{,}80 & 4{,}97 & 5{,}13 & 5{,}29 \end{bmatrix}$$

Con todas estas fórmulas y la información que presenta el gráfico 3 ya es posible determinar el indicador que orientó al algoritmo a tomar una u otra opción. Por ejemplo, para calcular la decisión 6 del algoritmo sería de la siguiente forma:

$$\lambda_{1,6} = P_{1,5} + \sqrt{\frac{\gamma_5}{n_{1,5}}} = 0 + \sqrt{\frac{3{,}22}{1}} = 1{,}7941$$

$$\lambda_{2,6} = P_{2,5} + \sqrt{\frac{\gamma_5}{n_{2,5}}} = 0{,}67 + \sqrt{\frac{3{,}22}{3}} = 1{,}7025$$

$$\lambda_{3,6} = P_{3,5} + \sqrt{\frac{\gamma_5}{n_{3,5}}} = 0 + \sqrt{\frac{3{,}22}{1}} = 1{,}7941$$

De estas tres alternativas el UCB eligirá a la mayor, que en este caso podrían ser tanto la opción A como la opción C. Asimismo, se puede observar que en B, a pesar que el porcentaje de acierto es mayor, no llegó a superar en puntaje a las otras alternativas debido a ya fue elegida 3 veces en otras oportunidades.

j	Indicador UCB		
	$\lambda_{1,j}$	$\lambda_{2,j}$	$\lambda_{3,j}$
1	-	-	-
2	-	-	-
3	-	-	-
4	1.4823	2.4823	1.4823
5	1.6651	2.1774	1.6651
6	1.7941	1.7025	1.7941
7	1.3386	1.7596	1.8930
8	1.3950	1.8056	1.3949
9	1.4420	1.7697	1.4420
10	1.4823	1.5375	1.4823
11	1.5174	1.5428	1.5174
12	1.5485	1.5420	1.5485
13	1.6204	1.5569	1.5764
14	1.3825	1.5703	1.6015

Tab. 1: Indicador de criterio de selección UCB

En las primeras elecciones, el UCB intentará explorar todas las opciones al menos una vez, antes de empezar a tomar el valor de $\lambda_{i,j}$. Esto puede verse en la tabla 1, donde se encuentran, marcados en color verde, los indicadores que representan el nivel de confianza máximo en cada una de las 14 decisiones del caso de ejemplo, planteado al inicio de esta sección.

En este apartado se ha logrado explicar a grandes razgos cómo funciona el UCB para encontrar el «brazo» óptimo, en un problema de bandido multibrazo. A pesar que la implementación del algoritmo no es compleja y puede ser replicada en cualquier lenguaje de programación, en este trabajo se empleó el paquete UCB de Kurth Ericson para Node.Js, disponible en NPMjs.

5. Implementación

En esta sección, se describe la propuesta para implementar aprendizaje automático, con el algoritmo UCB, en la selección de anuncios más influyente para los clientes. Para esto, se parte del supuesto que la aplicación web de la compañía ya está desarrollada con todas sus funcionalidades para ofrecer y vender productos. Asimismo, los anuncios que van a mostrarse, que en este caso vendrían a ser archivos de imágenes, también ya han sido diseñados y están listos para usarse en la página web. Como se mencionó en el caso práctico, se tienen tres posibles y los identificaremos como A, B y C.

En la figura 4 se muestra el proceso que se seguirá al momento de mostrar el banner publicitario al usuario. La secuencia consta de 7 pasos, que se detallan a continuación:

1. El usuario ingresa al portal web para ver los productos.
2. Al cargar la página se invoca al servicio para seleccionar cual de los 3 anuncios se va a mostrar.
3. El servicio utilizará el algorimo UCB para determinar el anuncio más adecuado, de acuerdo a los indicadores de éxito/fracaso que tiene registrado.
4. Se carga la imagen del anuncio seleccionado por el servicio.
5. El usuario al ver el anuncio tiene dos opciones, cerrarlo o ir a la ventana de compra.
6. Para cualquiera de los dos casos la web invocará a un método que registre la selección del usuario.
7. El servicio registra la compra como recompensa y el cierre de ventana como penalidad en los indicadores del algoritmo UCB.

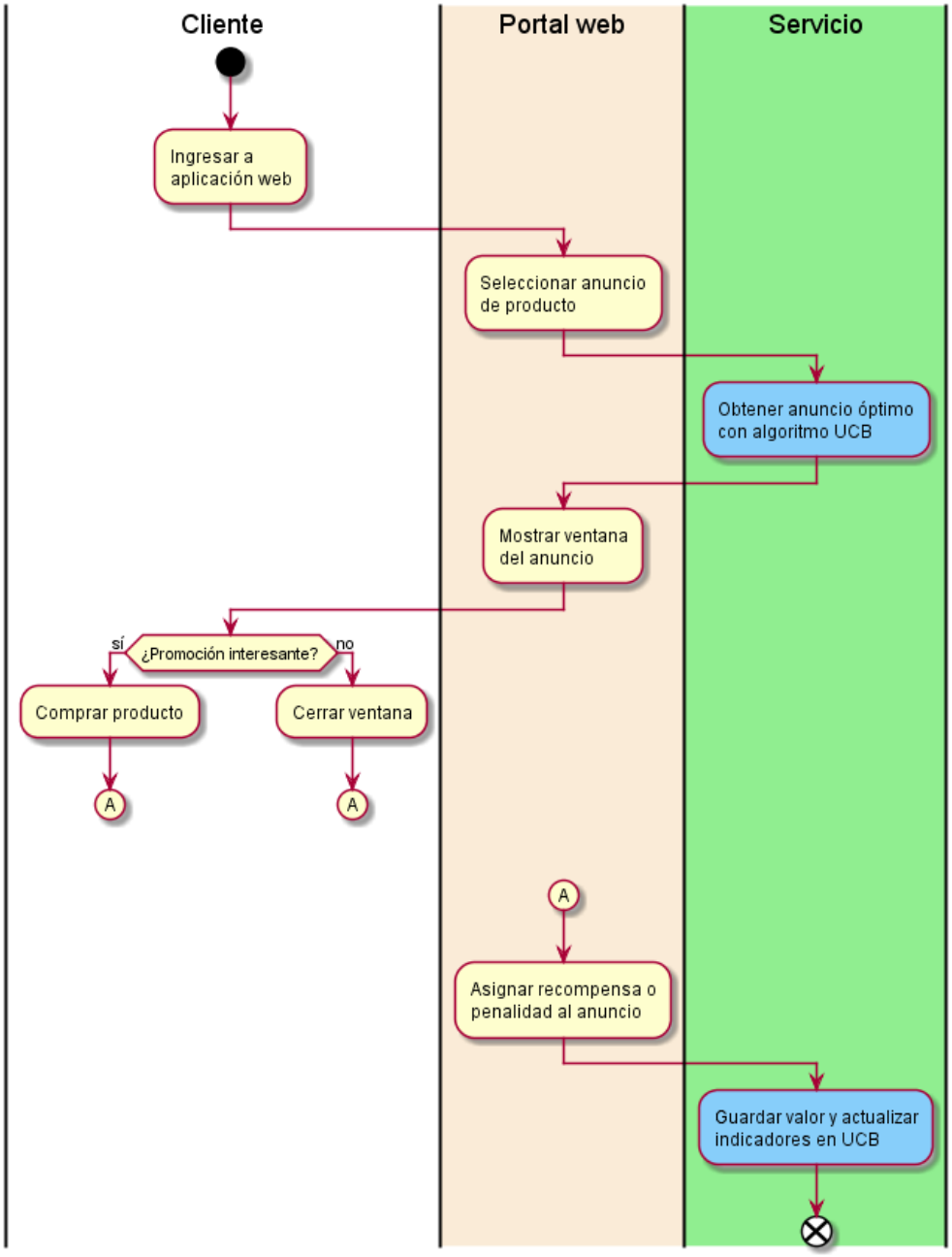

Fig. 4: Flujo de proceso de la funcionalidad

Las tareas marcadas en celeste del gráfico 4, representan las funcionalidades que van a desarrollarse. Como se puede observar, quien hace uso del servicio no es el cliente directamente, sino el portal web. Por tanto, este vendría hacer el actor para los dos casos de uso definidos, tal como muestra la figura 5.

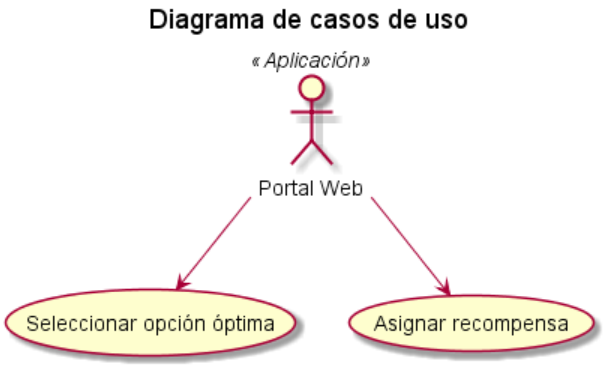

Fig. 5: Casos de uso del servicio

Para cumplir con los requerimientos identificados, se necesitarán basicamente tres componentes. En primer lugar, el módulo con la implementación del algoritmo UCB, el servicio que expondrá las funcionalidades a la aplicación cliente y, finalmente, el componente de persistencia; para guardar el estado de los indicadores del algoritmo. En el siguiente gráfico se puede apreciar la interacción de cada uno de estos componentes.

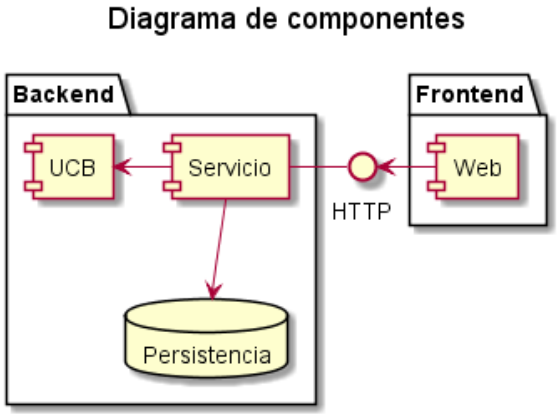

Fig. 6: Diseño de componentes propuesto

- **UCB:** Para este componente se propone la utilización del paquete UCB para Node.js.
- **Servicio:** Como este es el encargado de hacer uso de la libreria UCB, que está en Node.js, es posible usar cualquiera de los frameworks web disponibles para ese entorno de ejecución, como por ejemplo: Express, Koa, Fastify, entre otros.
- **Persistencia:** Para este componente puede emplearse cualquier medio de persistencia de datos, desde un archivo plano hasta un servidor de base de datos SQL o basado en documentos.

Hasta este momento, se tienen definidos los casos de uso y los componentes necesarios para implementarlos. Ahora, se explicará cómo estos intervienen en la secuencia pasos necesarios para llevar a cabo cada funcionalidad.

- **Inicialización:** Cuando el servicio se reinicia por un despliegue o caída, es necesario verificar si existen parámetros del algoritmo UCB en base de datos, para asignarlos al momento de crear el objeto. Esto con el fin de que el sistema continue el aprendizaje donde se quedó. Obviamente, la primera vez que inicie el sistema no existirá información previa y empezará a aprender desde cero.

- **Obtener anuncio:** Para obtener el anuncio óptimo, se hará uso del método select del paquete UCB. Este devuelve el número de brazo que se está seleccionando, haciendo referencia al problema del bandido multibrazo. Como se tienen tres anuncios diferentes A, B y C; los posibles valores (brazos) que se podrán retornar serán 0, 1 y 2. Luego, la parte front se encargará de interpretar este valor y mostrar el anuncio correspondiente.

- **Asignar recompensa:** Cada vez que se muestra un anuncio el cliente tiene dos opciones, comprar el producto o cerrar la ventana, según sea el caso se asignará una recompensa positiva o negativa, respectivamente. Para esto, el servicio empleará el método reward, que se encargará de actualizar los indicadores de éxito de cada opción y retornalos. Asimismo, este recibe como parametros el número de brazo, que representa el indicador del anuncio que se está evaluando, y un número binario, donde 0 indica que no el usuario cerró la ventana y 1 que accedió a comprar el producto. Cuando se tenga la respuesta del método, esta se guardará en base de datos para que pueda ser consultada posteriormente en la parte de inicialización.

La representación gráfica de estos procesos puede visualizarse en la figura 7.

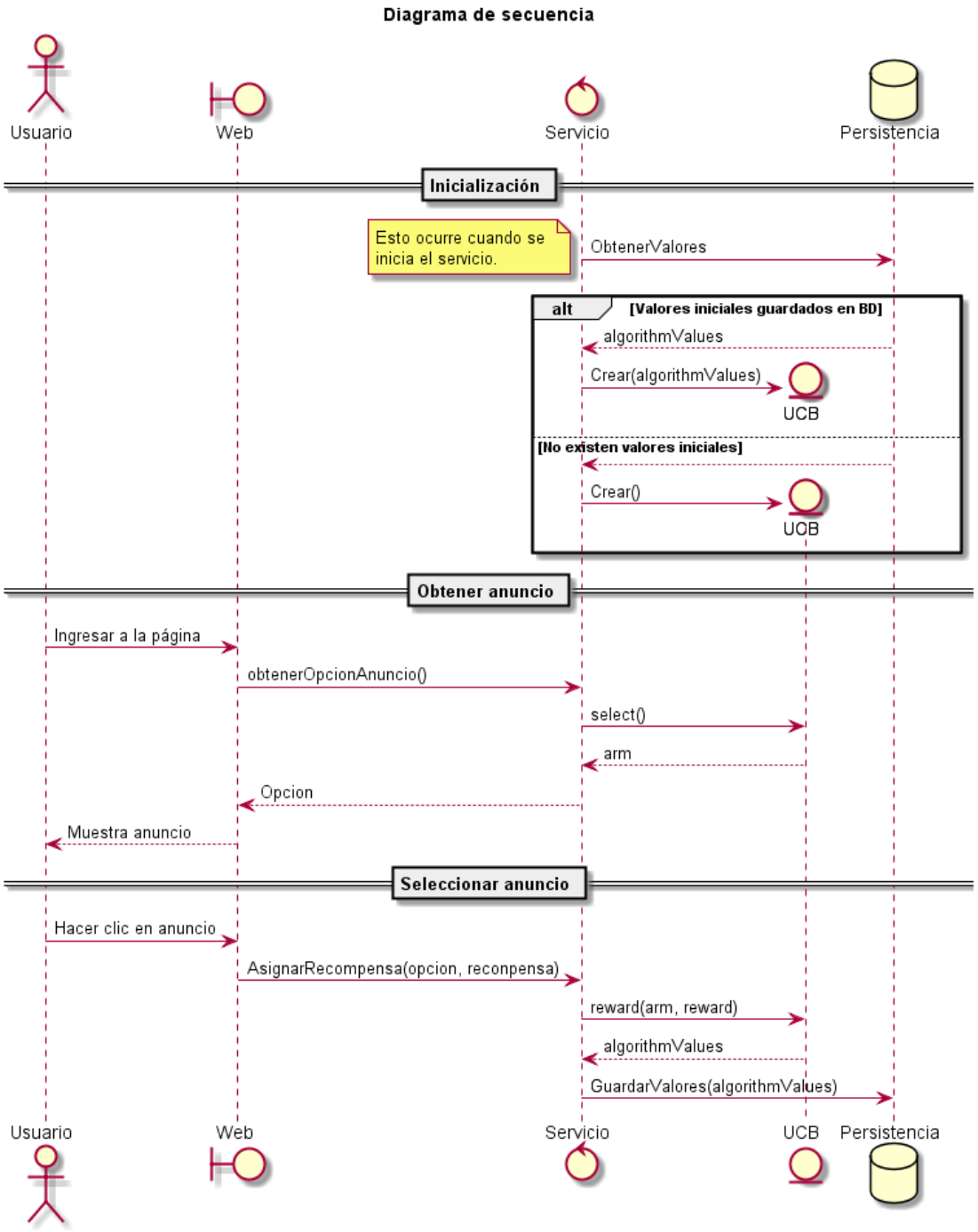

Fig. 7: Implementación de paquete UCB para Node.js

El código en javascript, donde se implementa esta lógica, está publicado en Github. Puede consultarse y reutilizarse libremente.

6. Pruebas

Para evaluar la solución se han preparado 6 casos de pruebas, simulados con 10000 usuarios, con las siguientes condiciones:

1. La probabilidad de éxito de los anuncios A, B y C es de 30%, 50% y 20%; seleccionándolos aleatoriamente.
2. La probabilidad de éxito de los anuncios A, B y C es de 30%, 50% y 20%; seleccionándolos con el servicio UCB.
3. La probabilidad de éxito de los anuncios A, B y C es de 30%, 80% y 20%; seleccionándolos aleatoriamente.
4. La probabilidad de éxito de los anuncios A, B y C es de 30%, 80% y 20%; seleccionándolos con el servicio UCB.
5. La probabilidad de éxito de los anuncios A, B y C es de 22%, 25% y 20%; seleccionándolos aleatoriamente.
6. La probabilidad de éxito de los anuncios A, B y C es de 22%, 25% y 20%; seleccionándolos con el servicio UCB.

Como se puede observar, se están utilizando dos métodos de elección diferentes. El primero es aplicando un criterio arbitrario, mientras el segundo es la propuesta de aprendizaje automático. Esto, con la finalidad de poder hacer una comparación de ambos resultados e identicar que tanto difieren en la ganancia final.

La ejecución de las pruebas fueron realizadas con la herramienta Postman, donde se elaboró el siguiente script para probar el servicio de asignación de recompensa:

```
pm.sendRequest("http://localhost:5000/api/ucb", function(err, response) {
    var selection = response.json()
    pm.collectionVariables.set("option", selection.opcion)

    var prob = [0.3, 0.5, 0.2]
    var num = Math.random()
    if(num <= prob[selection.opcion])
        pm.collectionVariables.set("reward", 1)
    else
        pm.collectionVariables.set("reward", 0)
});
```

7. Resultados

La evaluación de resultados se realizó utilizando 10000 casos de prueba, donde se seleccionaron anuncios utilizando los criterios del algoritmo UCB y se simularon las acciones del usuario de comprar o cerrar la ventana de la promoción. Para esto, a cada una de las promociones A, B y C se les ha asignado una probabilidad de venta de 30 %, 50 %, 20 % respectivamente. Con este escenario, se espera que el algoritmo explore los tres banners, inicialmente, hasta que, llegado cierto punto, pueda inferir con certeza que el anuncio B es el que tiene más posibilidades de vender el producto.

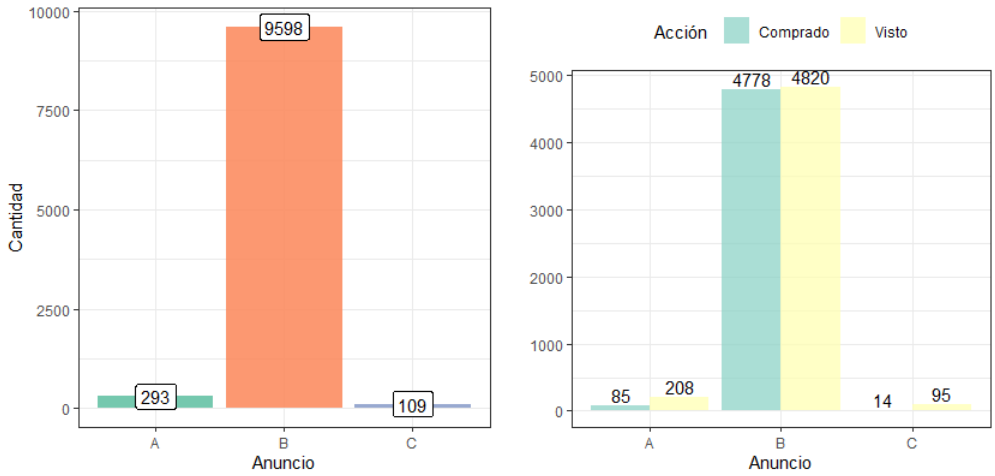

Fig. 8: Resultados de selección de anuncios con UCB

En la figura 8 se muestran dos gráficos de barras, obtenidos con los resultados de las 10000 pruebas realizadas al algoritmo UCB. Al lado izquierdo, se tienen las cantidades totales que el algoritmo seleccionó cada banner. Por otra parte, al lado derecho, se presenta un gráfico comparativo de las ocaciones en las que se optó por comprar el producto versus las veces en las que se visualizó el anuncio.

Como se puede apreciar el UCB eligió en más del 95 % de los casos (9598) al anuncio B, convirtiendo aproximadamente la mitad estos en ventas para la empresa. Por otro lado, el anuncio C fue el que menos se tomó en cuenta, tan solo fue elegido 109 veces por el algoritmo, y objtuvo unicamente 14 ventas, debido a que solo tenía 20 % de probabilidad de éxito. Finalmente, la opción A fue una alternativa tentativa al comienzo y se seleccionó 293 veces, pero por su 30 % de efectividad fue también descartada.

Estos resultados evidencian que el UCB funciona de forma óptima para el caso planteado. Sin embargo, cabe también preguntarse que hubiese pasado si aplicabamos un criterio de selección más común, como el aleatorio, para saber si la diferencia de conversión en ventas justifica el esfuerzo de implementar aprendizaje por refuerzo.

En ese sentido, se desarrolló otro bloque de 10000 pruebas al mismo escenario, aplicando una selección de anuncios al azar, para comparar este comportamiento y beneficios con los obtenidos en el bloque anterior de pruebas.

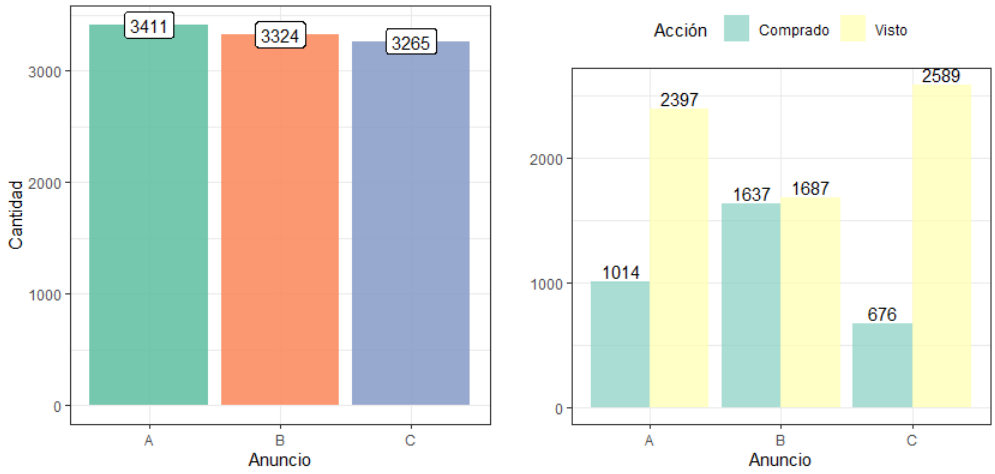

Fig. 9: Resultados de selección de anuncios al azar

De forma análoga al caso anterior, se tienen los gráficos de la figura 9, donde se puede apreciar al lado izquierdo como los 3 anuncios fueron elegidos casi de forma homogénea. Asimismo, al lado derecho, se tiene un comparativo de la cantidad de casos en las que se consiguió vender el producto versus las veces en las que solo fue visto, para cada banner. A pesar que el anuncio B consiguió mayor venta, por tener un porcentaje de éxito superior al resto; la selección estocástica ha provocado que se pierdan muchas oportunidades apostando demasiadas ocasiones por los anuncios A y C, que tienen baja probabilidad de convertirse en venta (30 % y 20 %, respectivamente). Esto es visible en el alto número de anuncios solo vistos, un total de 6673.

Otra diferencia notable entre los dos criterios de selección evaluados, es la cantidad de selecciones positivas obtenidas o cantidad de compras del cliente. El UCB permitió que la efectividad del anunció B acumule un total de 4778, mientras que la aleatoriedad obtuvo cantidades inferiores distribuidas en tre las tres opciones (ver figura 10). En cifras generales, el aprendizaje por refuerzo alcanzó casi las 5000 ventas, mientras que el otro método empleado se aproximó a las 3500 (ver figura **??**). Esta es una brecha importante, pero no es algo que va a darse siempre porque está estrechamente relacionado a la probabilidad de éxito de los anuncios.

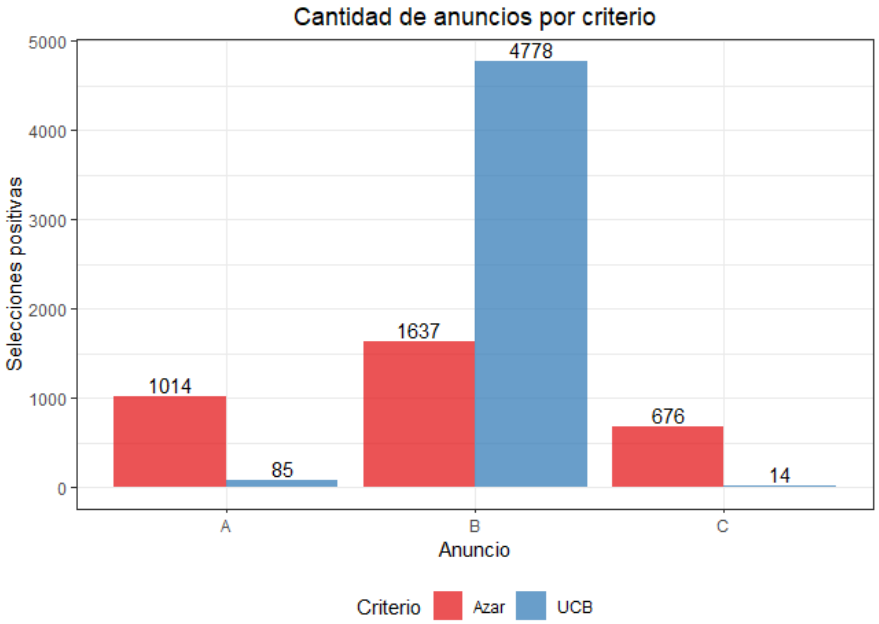

Fig. 10: Comparación de anuncios seleccionados UCB vs azar

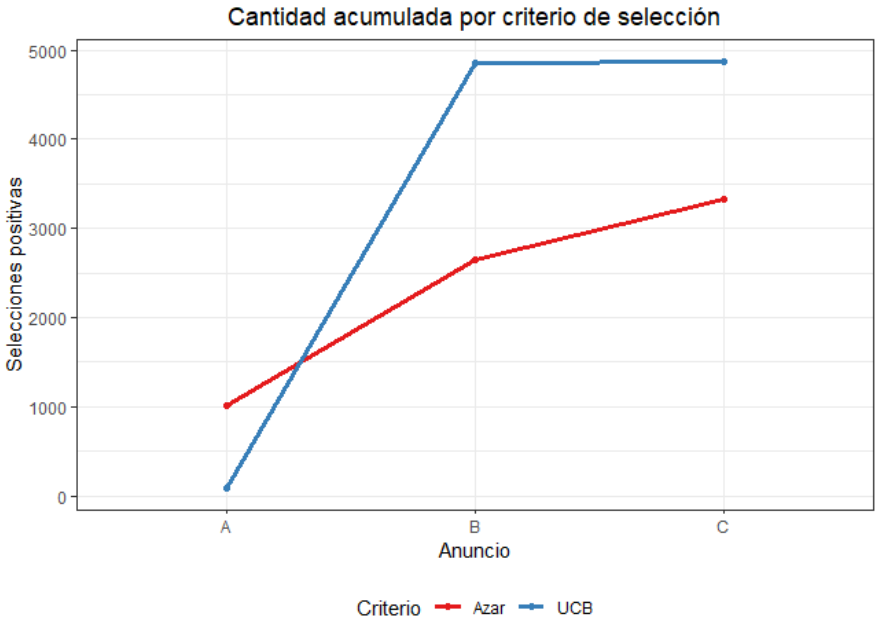

Fig. 11: Total de anuncios seleccionados por cada criterio

Para demostrar la relevancia de la probabilidad de venta de cada anuncios, se ha analizado los porcentajes globales de éxito obtenidos con cada método. En primer lugar, con el criterio estocástico se alcanzó un total de 33 %, que es equivalente a la probabilidad promedio de los 3 banners.

$$\frac{P_A + P_B + P_C}{3} = \frac{30 + 50 + 20}{3} = 33{,}33\,\%$$

Por otro lado, el aprendizaje por refuerzo obtuvo un total de 49 %, que es aproximadamente la probabilidad de éxito del anuncio B (ver figura 12).

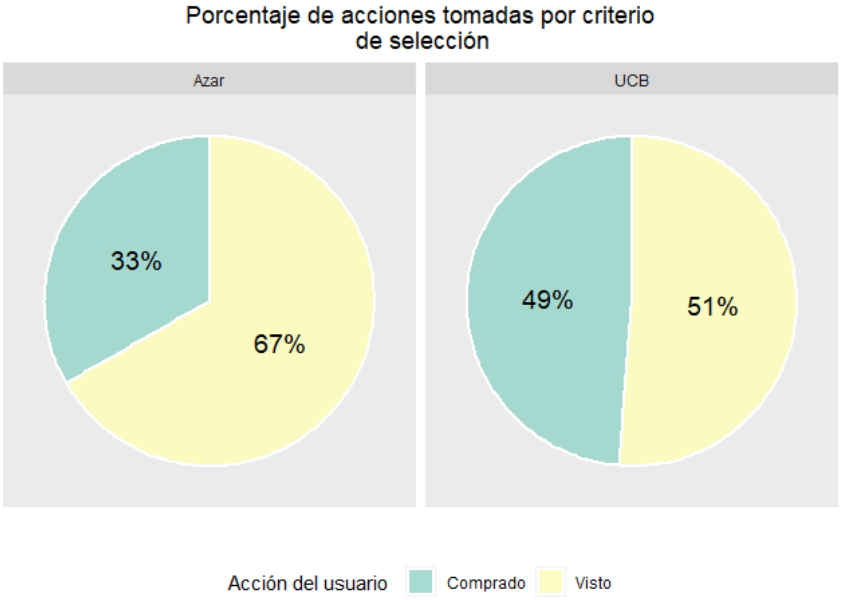

Fig. 12: Porcentaje de acciones tomadas por el usuario por criterio

Ahora si se plantea otro escenario en el que $P_B = 80\,\%$, con 10000 casos de prueba, se puede apreciar que se sigue el comportamiento mencionado anteriormente. La selección aleatoria consiguió que el cliente comprara en un 43 % de las veces y el UCB alcanzó un 80 %, coincidente con el nuevo valor para P_B (ver figura 13).

$$\frac{P_A + P_B + P_C}{3} = \frac{30 + 80 + 20}{3} = 43{,}33\,\%$$

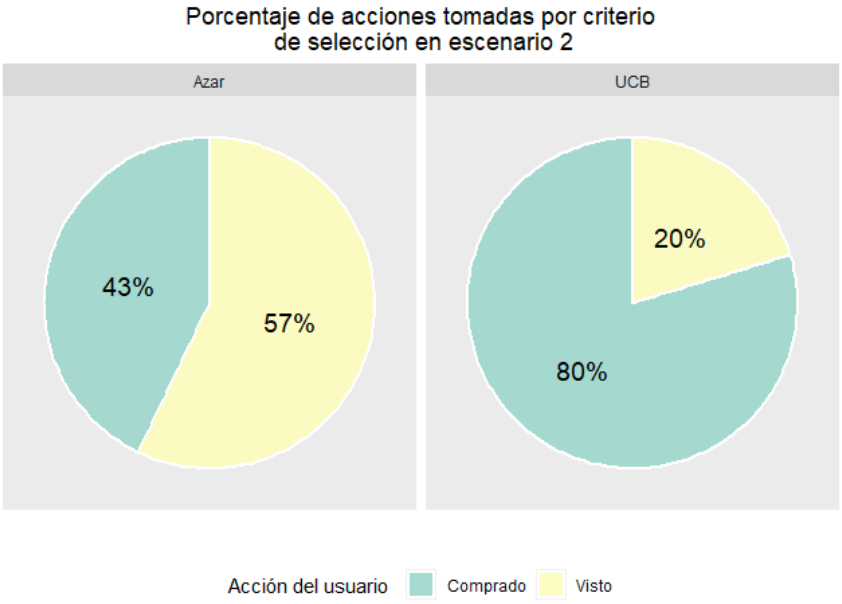

Fig. 13: Porcentaje de acciones tomadas por el usuario por criterio

Este comportamiento, evidencia que el porcentaje de vender un producto con un grupo de anuncios dependa de la probabilidad de éxito del mayor de ellos, si se aplica aprendizaje por refuerzo para la selección de los mismos. Esto significa, que si todos los banners son similarmente buenos o malos, el algoritmo no aportará un beneficio muy notorio. Por otro lado, cuando uno de ellos sobresale del resto el UCB esa capaz de identificarlo y explotar ese potencial.

8. Conclusiones

El objetivo principal de este trabajo, es proponer una solución de aprendizaje automático para resolver un caso práctico del problema del bandido multibrazo. Para esto, se planteó el diseño básico de un servicio que permita a cualquier página web, donde se requiera promocionar un producto, hacer uso del aprendizaje por refuerzo para seleccionar el mejor anuncio y, de esta forma, obtener más ventas en comparación a una selección aleatoria.

Para el desarrollo del servicio se utilizaron las tecnologías Node.js con Fastify y MongoDB. Sin embargo, es posible emplear cualquier otro framework web, lenguaje de programación o base de datos que convenga según el caso. Ya que, como se ha explicado, el algoritmo UCB tiene una implementación sencilla y es facilmente replicable. También, podría optarse por utilizar alguna otra libreria con un algoritmo diferente de aprendizaje por refuerzo, y obtener resultados similares.

Los resultados obtenidos han sido favorables en comparación a un método tradicional aleatorio, ya que se consigue aprovechar el ratio de conversión mayor de entre todos los posibles anuncios. Además, gracias al aprendizaje en tiempo real, se evitó la inversión de tiempo exclusivo para exploración, que normalmente es necesario en las pruebas A/B.

Es necesario considerar que existen algunas circunstancias en las que el UCB no será capaz de obtener resultados superlativos. Por ejemplo, cuando la efectividad de conversión de las opciones es similar, el algoritmo necesitará tomar más decisiones antes de converger. Pero, inclusive en esos casos, la pérdida de oportunidades es mínima debido a que en esa situación la selección entre una alternativa u otra es casi irrelevante. En segundo lugar, cuando se tienen pocos clientes (200 a menos) el algoritmo dificilmente podrá encontrar el óptimo. Como el sistema aprende «sobre la marcha», si se tienen pocas interacciones es probable que el aprendizarje quede a medias, provocando que los resultados se asemejen mucho a los de una politica de seleccion arbitraria.

El aprendizaje por refuerzo, aplicado a la optimización de anuncios en páginas web, es capaz de aportar grandes beneficios a la converción en una promoción o campaña. Además, su desarrollo e implementación es relativamente rápido, si no existen restricciones de infraestructura, alrededor de una o dos semanas. Finalmente, la única forma de saber con certeza si se obtendrán o no mejoras es implantando la solución; debido a que la efectividad del algoritmo depende mucho de la cantidad de clientes y la probabilidad de éxito de los anuncios, que dificilmente pueda conocerse a priori.

Referencias

[1] D. Arumughom, "Multi-armed bandit algorithms for website optimization," https://browsee.io/blog/bandit-algorithms-for-website-optimization/, Noviembre 2019, [Online; accedido 23-Mayo-2020].

[2] M. Silva, "Aprendizaje por refuerzo: Introducción al mundo del rl," https://medium.com/aprendizaje-por-refuerzo-introducci%C3%B3n-al-mundo-del/aprendizaje-por-refuerzo-introducci%C3%B3n-al-mundo-del-rl-1fcfbaa1c87, Abril 2019, [Online; accedido 02-Mayo-2020].

[3] J. Rojas, "A/b testing: una buena forma de medir el éxito de tus campañas," https://velogig.com/a-b-testing-mide-tus-acciones-de-marketing/, Mayo 2019, [Online; accedido 09-Abril-2020].

[4] C. Hubbs, "Multi-armed bandits: Ucb algorithm," https://towardsdatascience.com/multi-armed-bandits-ucb-algorithm-fa7861417d8c, Enero 2020, [Online; accedido 09-Abril-2020].

[5] M. Sanz, "Introducción al aprendizaje por refuerzo. parte 1: el problema del bandido multibrazo," https://medium.com/@markelsanz14/introducci%C3%B3n-al-aprendizaje-por-refuerzo-parte-1-el-problema-del-bandido-multibrazo-afe05c0c372e, Marzo 2020, [Online; accedido 09-Abril-2020].

[6] banditalgs.com, "The upper confidence bound algorithm," https://banditalgs.com/2016/09/18/the-upper-confidence-bound-algorithm/, Setiembre 2016, [Online; accedido 09-Abril-2020].