# Comparativa de algoritmos clásicos de aprendizaje por refuerzo sobre el problema del bandido de $K$ brazos

Ana Gil Molina
José María García Ortiz
Levi Malest Villarreal

9 de marzo de 2025

## Resumen

En este documento se presentará un estudio comparativo del desempeño de ciertos algoritmos clásicos del aprendizaje por refuerzo, sobre el problema del bandido de $K$ brazos. El objetivo de este estudio es analizar las diferencias en el comportamiento de los algoritmos según las particularidades de este *entorno* y determinar cuáles de ellos resuelven mejor el problema; basándonos en unas conclusiones respaldadas por los estudios (replicables) que pueden encontrar en [1].

Los estudios planteados consisten en analizar gráficamente, para cada algoritmo, el porcentaje de elecciones de brazo óptimo en cada etapa del aprendizaje; y también el *regret* acumulado en cada una de estas etapas. Con esto, tenemos suficiente información para evaluar el rendimiento de los algoritmos, sobre bandidos de distribución Normal, Bernoulli y Binomial de forma aislada. También podemos observar si los cambios en la distribución de los brazos afectan al rendimiento de los algoritmos, o si por contra, resultan independientes de ello.

## 1. Introducción

### 1.1. Descripción del problema

El problema del bandido de k brazos es un marco fundamental en el aprendizaje por refuerzo y la toma de decisiones secuenciales bajo incertidumbre. Inspirado en el dilema de un jugador en un casino que enfrenta múltiples máquinas tragamonedas (cada una con una distribución desconocida de recompensas), este problema captura la tensión entre la exploración y la explotación: ¿debería el jugador seguir eligiendo la máquina que ha dado mejores resultados hasta ahora o probar nuevas máquinas en busca de una recompensa aún mayor?

Matemáticamente, el problema se modela como un conjunto de $K$ brazos, cada uno asociado a una distribución de recompensas desconocida. El objetivo del agente es maximizar su recompensa acumulada a lo largo de un horizonte temporal, seleccionando iterativamente un brazo y recibiendo una recompensa en función de la distribución subyacente del mismo. En este contexto, se definen métricas clave como el **porcentaje de elecciones del brazo óptimo**, la recompensa promedio obtenida por cada brazo y el ***regret acumulado***, que mide la diferencia entre la recompensa óptima posible y la recompensa efectivamente obtenida por el agente.

Esta problemática es de gran interés teórico y práctico, ya que se relaciona directamente con situaciones del mundo real donde se deben tomar decisiones bajo incertidumbre, tales como la selección de estrategias de inversión, la asignación de recursos en publicidad o la optimización de procesos en ingeniería. La complejidad del problema surge de la necesidad de balancear la explotación de las opciones ya conocidas y la exploración de alternativas potencialmente más rentables, para minimizar el regret acumulado.

### 1.2. Motivación

En el contexto de este problema, la principal dificultad que encontramos es la elección de un método inteligente de selección de brazo en cada tirada, que maximice nuestras ganancias a corto, medio o largo plazo. Pero además, incluso si disponemos de una batería de algoritmos capaces de decidir *con buen criterio* sobre qué brazo conviene escoger en cada tirada, deberíamos tener en cuenta la distribución de probabilidad asociada a cada uno de los brazos del bandido, es decir, el algoritmo debería conocer estas distribuciones (o una estimación de las mismas) para construir su criterio y regularlo en función del bandido que nos encontremos.

Otra posibilidad sería disponer de algoritmos suficientemente robustos como para aprender correctamente con independencia de las distribuciones de los brazos del bandido. Sin embargo, la incertidumbre a priori es doble: primero, no conocemos aún el desempeño *en general* de cada uno de los algoritmos de los que disponemos, y segundo, desconocemos si su rendimiento puede variar al hacerlo las distribuciones de los brazos del bandido, lo cual podría condicionar nuestra elección de algoritmo antes de empezar a jugar.

### 1.3. Objetivos del trabajo

Debido a lo que se acaba de comentar, nuestro objetivo en este trabajo consiste en determinar qué algoritmo rinde mejor con según qué bandido, teniendo en cuenta (si procede) la distribución de los brazos del mismo.

### 1.4. Estructura del documento

En lo sucesivo, dedicaremos la sección 2 al desarrollo teórico del problema y de los métodos utilizados para su resolución en el estudio. La sección 3 contendrá el pseudocódigo de dichos algoritmos, y en la cuarta desarrollaremos los experimentos

realizados, para elaborar una conclusión sobre todo el trabajo en la quinta sección.

# 2. Formalización del problema y elementos de un agente de decisión

Para desarrollar los experimentos necesitamos conocer con detalle la modelización del problema en términos matemáticos, así como la concepción teórica de los algoritmos utilizados, basada en [2].

El problema del bandido de $K$ brazos es un problema de toma de decisiones secuencial en el que un agente de decisión (o algoritmo) debe elegir una de $K$ acciones (brazos) en cada paso temporal con el objetivo de maximizar la recompensa acumulada a lo largo del tiempo.

## 2.1. Round-Robin: Un primer enfoque *naive*

Una forma natural de enfrentarse al problema del bandido, suele ser la estrategia Round-Robin, que consiste en seleccionar brazos al azar de manera equiprobable en cada **instante de decisión** $t$. De esta forma se consigue explorar todas las decisiones posibles y obtener un retorno *razonable* entre el mejor y el peor posible. Es decir, la estrategia se enfoca en cambiar de brazo constantemente para no incurrir en un error de decisión permanente o mayoritario al menos, cosa que ocurriría por ejemplo si seleccionamos uno de los brazos con peores recompensas promedio y nos aferramos a él en nuestras decisiones.

Esta forma conservadora de enfrentar el problema, también es ciertamente *naive* ya que por su naturaleza, no permite aprovechar la información que ofrece cada decisión que tomamos, con el objetivo de aprender de ello y mejorar nuestra **política de decisiones**.

Si queremos usar una estrategia de decisión que aproveche esta información para mejorar las futuras decisiones que tomemos, necesitamos un algoritmo que construya un **agente de decisión**, con una política de decisiones propia y capaz de aprender de sus propias decisiones.

A continuación, presentamos una formalización de los elementos básicos de un agente de decisión:

## 2.2. Espacio de Acciones

El conjunto de acciones está dado por:

$$\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$$

donde cada acción $a_k$ representa la selección de un brazo específico.

## 2.3. Recompensas

Cada acción $a_k$ está asociada a una recompensa aleatoria $R_t$, distribuida de acuerdo a una distribución desconocida:

$$R_t \sim P(R|A_t = a_k)$$

con una esperanza matemática definida como el valor esperado de la recompensa para la acción $a_k$:

$$q^*(a_k) = \mathbb{E}[R_t|A_t = a_k].$$

Este valor $q^*(a_k)$ se conoce como el **valor verdadero de la acción**, y en este contexto, coincide por definición con la esperanza de la distribución asociada al brazo $a_k$.

## 2.4. Objetivo del Aprendizaje

El objetivo del agente es maximizar la recompensa acumulada a lo largo del tiempo, lo que equivale a seleccionar acciones óptimas con la mayor frecuencia posible. La mejor acción es aquella que maximiza la recompensa esperada:

$$a^* = \arg\max_{a \in \mathcal{A}} q^*(a).$$

a lo cual nos referimos también como ***brazo óptimo***.

## 2.5. Estimación de Valores de Acción

Dado que los valores verdaderos $q^*(a)$ son desconocidos, el agente mantiene estimaciones

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{I}_{A_t=a}}{\sum_{i=1}^{t-1} \mathbb{I}_{A_t=a}}$$

de los valores de acción, que se actualizan iterativamente a partir de las recompensas observadas:

$$Q_{t+1}(a) = Q_t(a) + \frac{1}{N_t(a)}(R_t - Q_t(a))$$

donde $N_t(a)$ es el número de veces que la acción $a$ ha sido seleccionada hasta el instante $t$.

## 2.6. Estrategia de Selección de Acción

Cada agente de decisión integra una estrategia para seleccionar qué acción es la mejor en cada instante $t$. Lo ideal es que la estrategia de un agente se actualice de forma eficiente consiguiendo identificar (y seleccionar en lo sucesivo) de forma temprana el brazo con mayor valor verdadero esperado. Sin embargo, en la práctica, el agente no conoce los valores estimados de los brazos hasta que los selecciona un número *suficiente* de veces, incluyendo el brazo óptimo. Esto significa que para poder elaborar la mejor estrategia de decisión, necesita seleccionar distintos brazos numerosas veces, para asignarles un valor estimado que se asemeje al valor verdadero, y entonces realizar elecciones de acción fiables (óptimas o cuasi-óptimas) en base a estos valores.

De esta forma, podemos afirmar que cada agente de decisión cuenta con un componente **exploratorio** y un componente de **explotación**, según los cuales, respectivamente, se compara el valor de las acciones conocidas con acciones nuevas; y se aprovechan los valores actuales para elegir la acción que más valor estimado posee para maximizar el retorno total a largo plazo.

Como veremos en la siguiente sección, podemos trabajar con diferentes agentes, cada uno con una forma distinta de actualizar los valores estimados de las acciones y de regular su balance exploración-explotación.

# 3. Algoritmos

## 3.1. $\varepsilon$-greedy

Este algoritmo toma una probabilidad preestablecida $\varepsilon$ de explorar seleccionando una acción o brazo sub-óptimo, mientras que elige como acción el brazo óptimo (de acuerdo a los valores estimados) con probabilidad 1-$\varepsilon$. Primero realiza una ronda exploratoria hasta seleccionar cada posible brazo una vez, y a partir de entonces, opera iterando el procedimiento de selección probabilístico que acabamos de comentar. Las etapas del algoritmo son las siguientes:

1. **Inicialización** ($t = 0$)
   - de valores estimados: $Q_0(a) = 0 \ \ \forall a \in \mathcal{A}$
   - del conteo de elecciones: $N(a) = 0 \ \ \forall a \in \mathcal{A}$
2. **Ronda exploratoria inicial**
3. **Selección de brazo** ($A_t = a$) **y obtención de recompensa** ($R_t$)
4. **Actualizar estimación del valor de la acción seleccionada**

$$Q_{t+1}(a) = Q_t(a) + \frac{1}{N_t(a)}(R_t - Q_t(a))$$

5. **Actualizar contadores de elección**

$$N(a) \mathrel{+}= 1$$

6. **Actualizar el instante de decisión**

$$t \mathrel{+}= 1$$

A continuación mostramos el pseudocódigo que lleva a cabo estos procesos:

**Algorithm 1** - $\varepsilon$-greedy

**Input:** $\varepsilon$, $K$, $\mathcal{A}$, $T$
**for** $a$ in $\mathcal{A}$ **do**
    $Q_0(a) \leftarrow 0$ , $N(a) \leftarrow 0$
**end for**
**while** $t < T$ **do**
    $A_t = a$ según la probabilidad $\varepsilon$
    Ejecutar $a$ y recibir recompensa $R_t$
    $N(a) \leftarrow N(a) + 1$
    $Q(a) \leftarrow Q(a) + \frac{1}{N(a)}(r - Q(a))$ç
    $t \leftarrow t + 1$
**end while**

## 3.2. UCB1

El algoritmo Upper Confidence Bound 1 (UCB1) supone una estrategia de exploración-explotación basada en la teoría de optimización en línea. A diferencia de $\varepsilon$-greedy, que elige acciones al azar con una probabilidad fija, UCB1 equilibra la exploración y explotación seleccionando la acción con la mejor combinación de su estimación de recompensa promedio y una medida de incertidumbre basada en la cantidad de veces que ha sido elegida.

La intuición detrás de UCB1 es que se debe priorizar la exploración de acciones que no han sido seleccionadas con frecuencia, ya que podrían ser óptimas. Explícitamente:

$$A_t = \arg\max_{a \in \mathcal{A}} \left( Q_t(a) + c\sqrt{\frac{\ln(t)}{N(a)}} \right)$$

donde $c$ es un parámetro que permite ajustar el balance exploración-explotación.

Los pasos que sigue el algoritmo son:

1. **Inicialización** ($t = 0$)
   - de valores estimados: $Q_0(a) = 0 \ \ \forall a \in \mathcal{A}$
   - del conteo de elecciones: $N(a) = 0 \ \ \forall a \in \mathcal{A}$
2. **Selección de brazo** ($A_t = a$) **y obtención de recompensa** ($R_t$)
3. **Actualizar estimación del valor de la acción seleccionada**

$$Q_{t+1}(a) = Q_t(a) + \frac{1}{N_t(a)}(R_t - Q_t(a))$$

4. **Actualizar contadores de elección**

$$N(a) \mathrel{+}= 1$$

5. **Actualizar el instante de decisión**

$$t \mathrel{+}= 1$$

Y en pseudocódigo:

**Algorithm 2** - UCB1

**Input:** $c$, $K$, $\mathcal{A}$, $T$
**for** $a$ in $\mathcal{A}$ **do**
    $Q_0(a) \leftarrow 0$, $N(a) \leftarrow 0$
**end for**
**while** $t \leq T$ **do**
    $a = A_t \leftarrow \arg\max_{a \in \mathcal{A}} \left( Q_t(a) + c\sqrt{\frac{\ln t}{N(a)}} \right)$
    Ejecutar $a$ y recibir recompensa $R_t$
    $N(A_t) \leftarrow N(A_t) + 1$
    $Q(A_t) \leftarrow Q(A_t) + \frac{1}{N(A_t)}(R_t - Q(A_t))$
    $t \leftarrow t + 1$
**end while**

## 3.3. UCB2

UCB2 es una extensión del algoritmo UCB1, que mejora la forma en que se realiza la exploración. Mientras que UCB1 utiliza un índice basado en una estimación de recompensa y un término de incertidumbre, UCB2 incorpora una penalización adicional en función del número de selecciones de la acción. Este ajuste permite una exploración más agresiva de las acciones menos seleccionadas, lo que puede ser beneficioso cuando se enfrenta a entornos con recompensas altamente variables. Explícitamente:

$$A_t = \arg\max_{a \in \mathcal{A}} \ UCB2(a) := Q(a) + \sqrt{\frac{(1+\alpha)\ln\left(\frac{t}{\tau(k_a)}\right)}{2\tau(k_a)}}$$

donde $\tau(k_a) = \lceil (1+\alpha)^{k_a} \rceil$ determina el número de veces que la acción $a$ será seleccionada en una época.

Las etapas del algoritmo coinciden genéricamente con las descritas anteriormente en $UCB1$, pero merece la pena mostrar su pseudocódigo:

**Algorithm 3** - UCB2

**Input:** $\alpha$, $K$, $\mathcal{A}$, $T$
**for** $a$ in $\mathcal{A}$ **do**
    $Q_0(a) \leftarrow 0$, $N(a) \leftarrow 0$
**end for**
**while** $t \leq T$ **do**
    $a = A_t \leftarrow \arg\max_{a \in \mathcal{A}} \ UCB2(a)$
    Ejecutar $a$ y recibir recompensa $R_t$
    $N(A_t) \leftarrow N(A_t) + 1$
    $Q(A_t) \leftarrow Q(A_t) + \frac{1}{N(A_t)}(R_t - Q(A_t))$
    $t \leftarrow t + 1$
**end while**

## 3.4. Softmax

El algoritmo Softmax selecciona las acciones en función de una distribución de probabilidad, en lugar de elegir siempre la acción con la mayor recompensa estimada o tomar decisiones aleatorias. La probabilidad de seleccionar una acción está determinada por la temperatura $\tau$, que controla el equilibrio entre la exploración y la explotación. En cada paso $t$, el algoritmo asigna a cada acción $a$ una probabilidad de ser seleccionada basada en su valor estimado $Q_t(a)$. La probabilidad de seleccionar una acción $a$ se calcula usando una distribución Softmax:

$$P(A_t = a) = \frac{e^{Q_t(a)/\tau}}{\sum_{j=1}^{K} e^{Q_t(a)/\tau}}$$

Aquí tenemos su pseudocódigo:

**Algorithm 4** - Softmax Action Selection

**Input:** $\tau$, $K$, $\mathcal{A}$, $T$
**for** $a$ in $\mathcal{A}$ **do**
    $Q_0(a) \leftarrow 0$, $N(a) \leftarrow 0$
**end for**
**while** $t \leq T$ **do**
    $a = A_t \leftarrow \arg\max_{a \in \mathcal{A}} \left( \frac{e^{Q_t(a)/\tau}}{\sum_{j=1}^{K} e^{Q_t(a_j)/\tau}} \right)$
    Ejecutar $a$ y recibir recompensa $R_t$
    $N(A_t) \leftarrow N(A_t) + 1$
    $Q(A_t) \leftarrow Q(A_t) + \frac{1}{N(A_t)}(R_t - Q(A_t))$
    $t \leftarrow t + 1$
**end while**

## 3.5. Gradiente de preferencias

El algoritmo de gradiente de preferencias es un algoritmo basado en optimización del gradiente que busca maximizar las recompensas mediante la actualización de las preferencias de las acciones basándose en los resultados observados. El gradiente de las preferencias es la dirección de la mayor mejora en la recompensa esperada, y el algoritmo ajusta las probabilidades de selección de las acciones de acuerdo con el gradiente estimado. En el algoritmo de gradiente de preferencias, las probabilidades de seleccionar una acción $a$ están determinadas por una distribución de probabilidades parametrizada por un vector de preferencias de cada acción, $H_t(a)$.

Se divide en los siguientes pasos:

1. **Inicialización** $(t = 0)$
   - de valores estimados: $Q_0(a) = 0 \ \ \forall a \in \mathcal{A}$
   - del conteo de elecciones: $N(a) = 0 \ \ \forall a \in \mathcal{A}$
2. **Selección de brazo** $(A_t = a)$ **y obtención de recompensa** $(R_t)$
3. **Actualizar las preferencias de las acciones**

$$H_{t+1}(a) := H_t(a) + \alpha \left(R_t - \overline{R}_t\right)(1 - \pi_t(a))$$

$$H_{t+1}(b) := H_t(b) - \alpha \left(R_t - \overline{R}_t\right)\pi_t(b) \ , \ \forall \ b \neq a$$

4. **Actualizar contadores de elección**

$$N(a) \mathrel{+}= 1$$

5. **Actualizar el instante de decisión**

$$t \mathrel{+}= 1$$

teniendo en cuenta que $\overline{R}_t$ denota la recompensa promedio del agente en el instante $t$.

El pseudocódigo correspondiente es:

**Algorithm 5** - Gradiente de Preferencias

**Input:** $\alpha$, $K$, $\mathcal{A}$, $T$
**for** $a$ in $\mathcal{A}$ **do**
    $Q_0(a) \leftarrow 0$ , $N(a) \leftarrow 0$ , $H_o(a) \leftarrow 0$
**end for**
$\overline{R}_t \leftarrow 0$
**while** $t < \ T$ **do**
    Seleccionar acción mediante $A_t$ según la probabilidad:

$$P(A_t = a_k) = \frac{e^{H_t(a)}}{\sum_{j=1}^{K} e_t^H(a_j)}$$

    Ejecutar $A_t$ y recibir recompensa $R_t$
    $\overline{R}_t \leftarrow \overline{R}_t + \frac{1}{t}(R_t - \overline{R}_t)$
    **for** $k = 1$ to $K$ **do**
        Actualizar preferencia del brazo $k$:
    **end for**
**end while**

# 4. Desarrollo del experimento

## 4.1. Descripción del experimento y métricas empleadas

El experimento que se ha realizado consiste en tomar dos *métricas* para analizar y comparar el rendimiento de los distintos agentes implementados en el entorno del bandido de 10 brazos. Estas métricas son el **porcentaje de elecciones de brazo óptimo en función del tiempo** y el **regret acumulado en función del tiempo**, que se define como:

$$R(T) = \sum_{t=1}^{T} (q^*(A_t) - R_t)$$

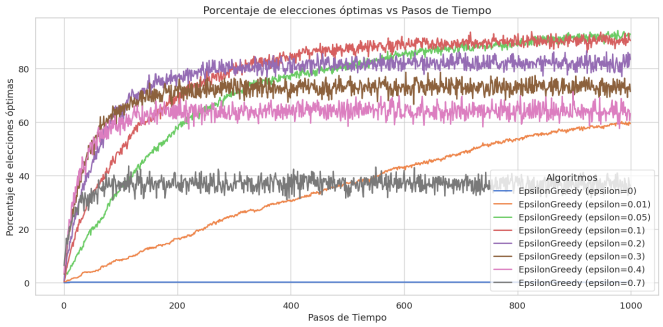
.

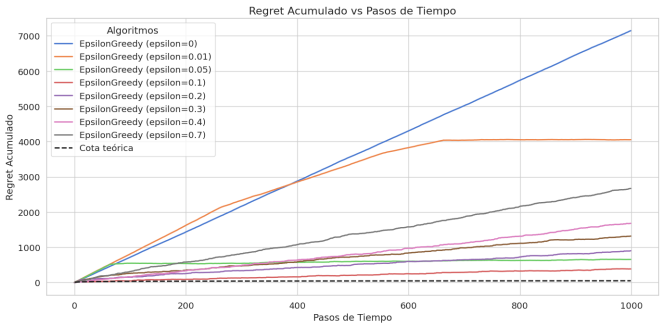
También contamos con otro tipo de medidas, pero hemos centrado el análisis sobre estas dos, ya que recogen información suficiente como para estudiar el desempeño de cada agente tanta corto, medio como largo plazo, sin ofrecer información redundante entre ambas.

También, el estudio pretende diferenciar los rendimientos en base a las distribuciones de los brazos del bandido, para responder a la pregunta de si esto es determinante en el desempeño de un agente cualquiera. Por ello, el análisis de resultados se divide según las 3 distribuciones de probabilidad: Normal, Bernoulli y Binomial ($1 \leq n \leq 50$).

## 4.2. Análisis de los resultados

### 4.2.1. Performance de la familia $\varepsilon$-greedy con brazos normales



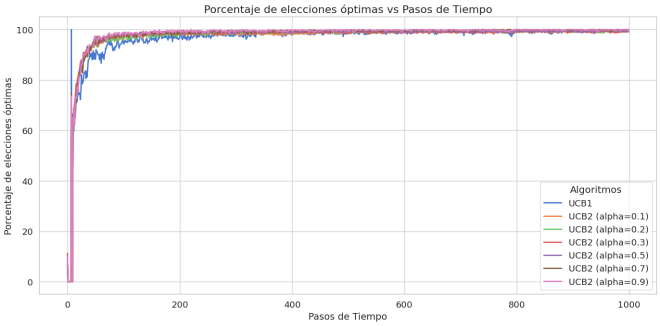



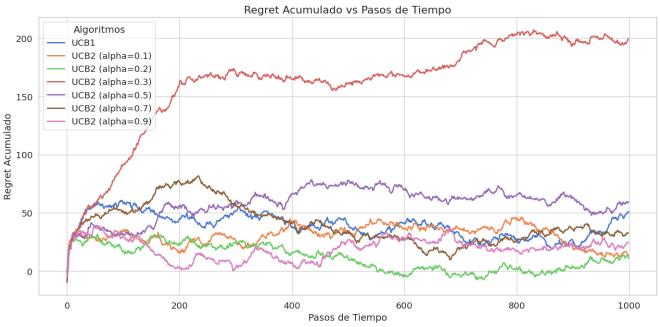

En estas gráficas sobre bandido con brazos Normales, podemos apreciar que el algoritmo $\varepsilon$-greedy que hace un mayor número de elecciones óptimas es el $\varepsilon = 0{,}05$ seguido de $\varepsilon = 0{,}1$. También resultan ser los que menos regret acumulan, acercándose a la cota inferior de *Lai* y *Robbins*. Esto significa que el rendimiento de estos algoritmos es realmente bueno a medio plazo, aunque los porcentajes de elecciones óptimas aún son algo mejorables.

Además, se hace evidente cómo el parámetro $\varepsilon$ ajusta la exploración, resultando en un mayor o menor porcentaje de elecciones óptimas y regret acumulado, incluso en los casos más extremos ($\varepsilon = 0$) donde el regret se dispara (en azul) y se selecciona persistentemente un brazo sub-óptimo, reflejando que no hay aprendizaje por parte del agente

### 4.2.2. Análisis general con brazos de distribución Normal

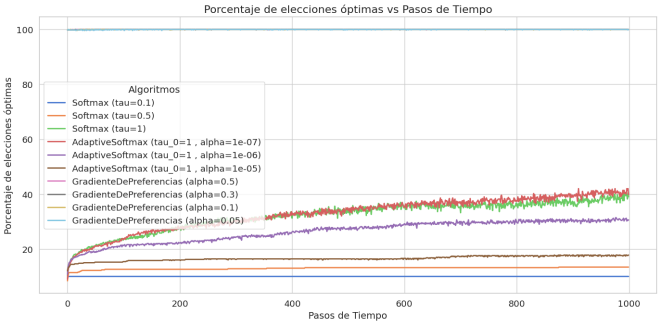
- **Familia** $UCB$

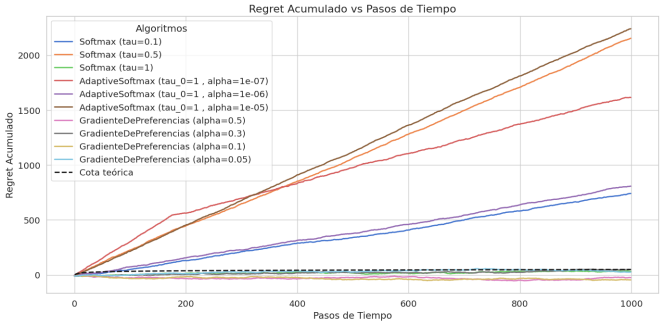



Podemos observar que todos los algoritmos UCB aprenden muy rápido cuál es el brazo óptimo y lo eligen de forma perpetua frente al resto. Al observar la gráfica de regret acumulado, salta a la vista un comportamiento erráticodel mismo, lejos de lo esperable para el regret de un algoritmo bien ajustado ya entrenado. Sin embargo, hay un motivo por el cuál no es de extrañar esta gráfica, y es debido a las variaciones de los retornos muestrales respecto del retorno esperado del brazo óptimo. Se elige constantemente el brazo óptimo con distribución Normal de desviación típica 1 y media cercana a 9, pero la desviación típica hace que a nivel muestral s perciban oscilaciones alrededor de 9 en las recompensas de cada tirada, y por tanto, sería verosímil encadenar muchas recompensas seguidas ligeramente alejadas de este valor. Por ello percibimos oscilaciones y cambios en la tendencia del regret acumulado. una forma de comprobar esto sería cambiando la semilla y confirmando que el comportamiento es similar pero no igual.

- **Familia ascenso de gradiente**

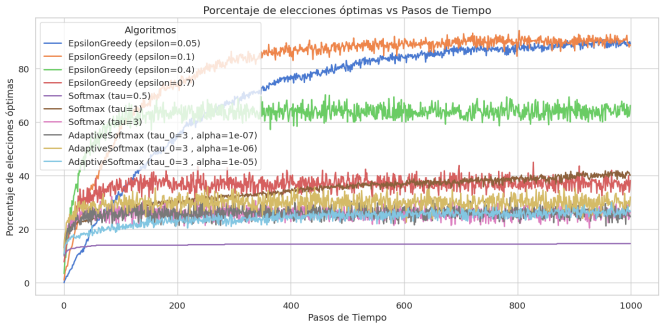


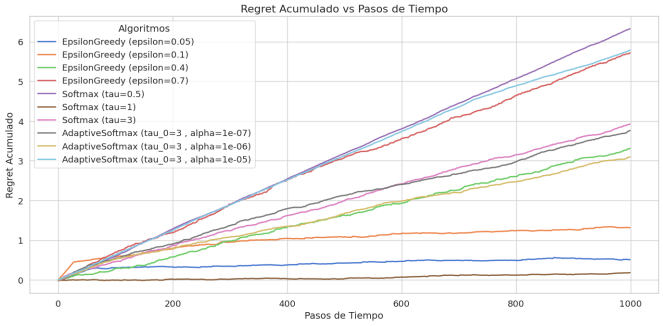



Con brazos normales, observamos que el rendimiento del algoritmo de ascenso de gradiente es el mejor de todos con diferencia, en cuanto a regret acumulado (medio plazo) y en cuanto a elecciones de brazo óptimo (largo plazo). El rendimiento de los métodos softmax son muy inferiores tanto a largo como medio plazo, con un porcentaje de elecciones óptimas inferior al 40 % y un regret acumulado que no se estabiliza en ningún caso.

Esto se debe a que la política de decisión que incorpora el softmax es demasiado poco exclusiva y reparte mucho la probabilidad de elección entre brazos sub-óptimos cercanos al brazo óptimo. Esto hace que el porcentaje de elecciones de brazo óptimo sea bastante pequeño y que por tanto, conforme evoluciona el algoritmo, el regret acumulado no se estabilice, trazando así una línea recta similar a un polinomio de grado 1 con pendiente positiva.

- **Softmax VS $\varepsilon$-greedy**

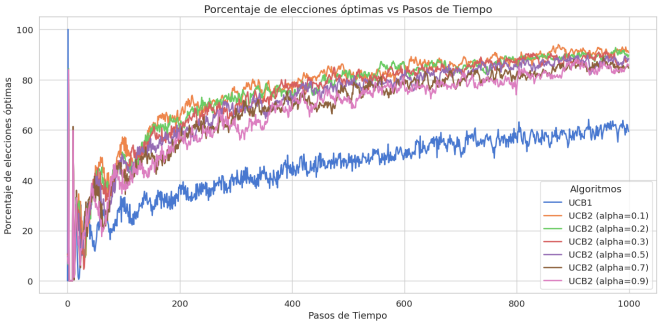


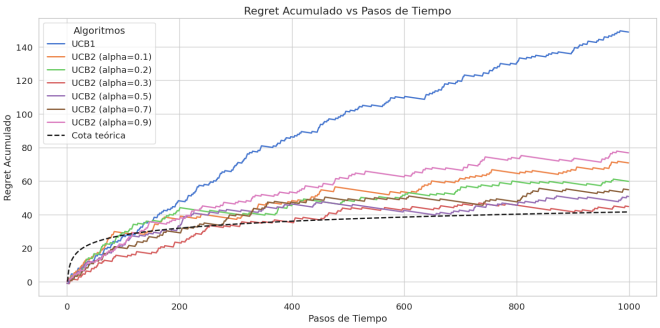




Por lo general, vemos que los $\varepsilon$-greedy funcionan mejor que el softmax. salvo, anecdóticamente, el Softmax en marrón que a medio plazo acumula menor regret que el resto de algoritmos de esta comparativa. Probablemente, debido a su forma de distribuir las probabilidades de elección entre los brazos sub-óptimos.

### 4.2.3. Análisis general con brazos de distribución Bernoulli

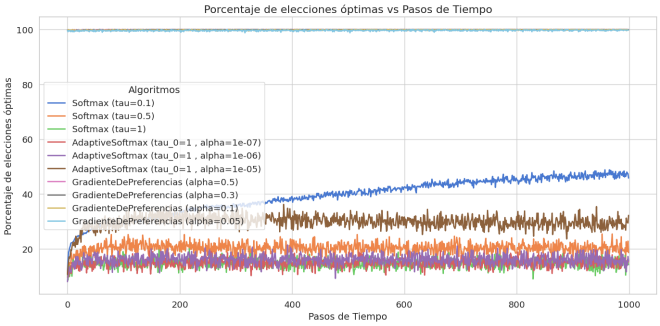
- **Familia $UCB$**

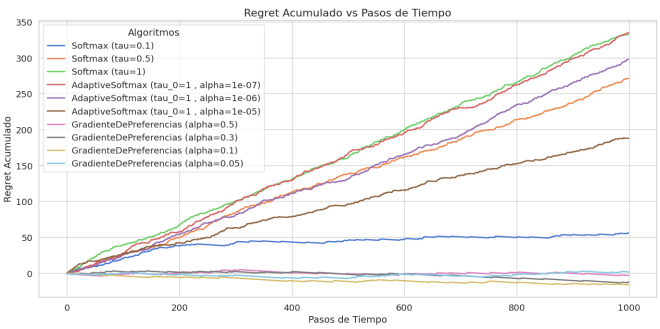



Merece la pena destacar que al cambiar a distribución Bernoulli, el comportamiento errático del regret de los métodos $UCB$ desaparece, y vemos como de hecho se ajusta finamente a la cota inferior, mostrando pues un gran rendimiento del $UCB2(\alpha = 0{,}2)$ que empeora ligeramente conforme aumentamos la exploración $\alpha$.

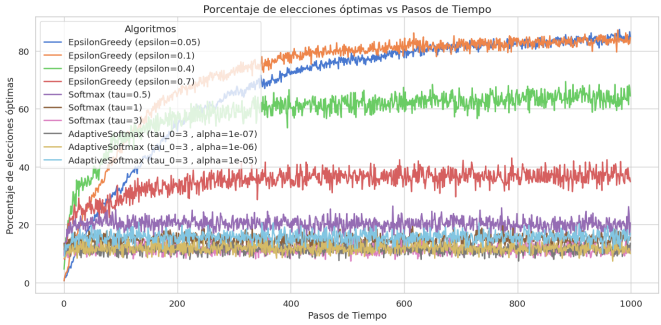
- **Familia ascenso de gradiente**

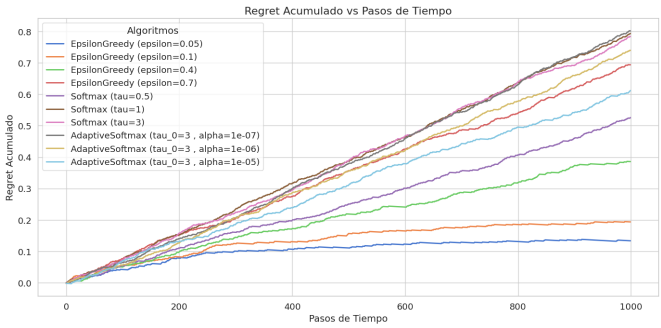


A penas hay cambios al cambiar de distribución en esta familia de algoritmos: El gradiente de preferencias mantiene su rendimiento y los softmax difieren en la misma medida en su desempeño.

- **Softmax VS $\varepsilon$-greedy**

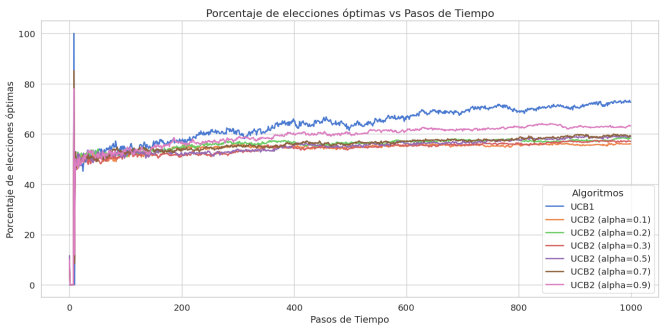


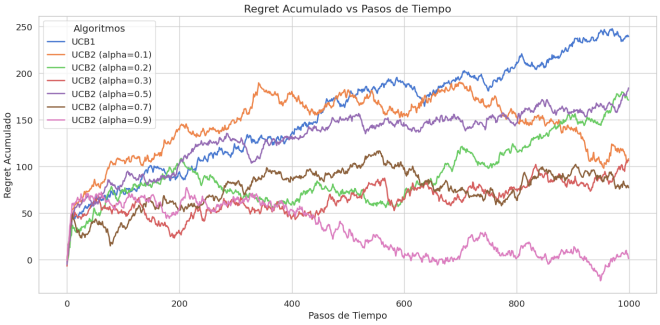




Los $\varepsilon$-greedy siguen dominando a los softmax al pasar a distribución Bernoulli, tanto en elecciones óptimas como en regret acumulado.

### 4.2.4. Análisis general con brazos de distribución Binomial

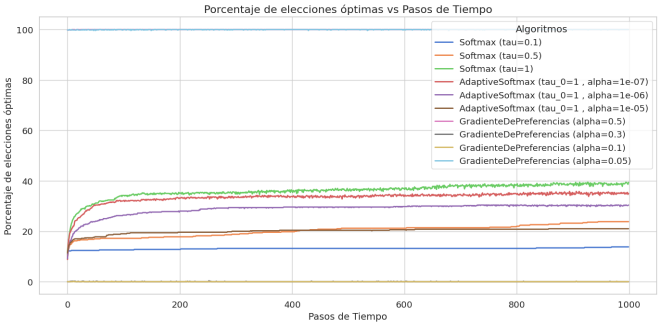
- **Familia $UCB$**

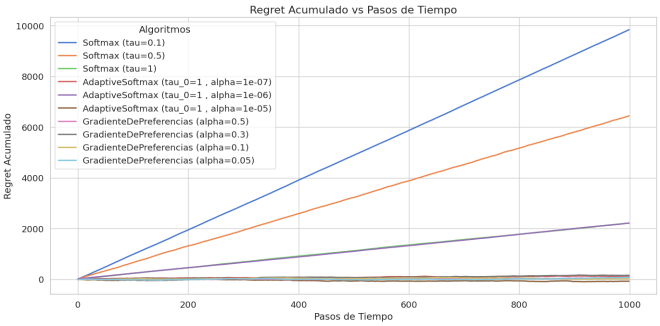


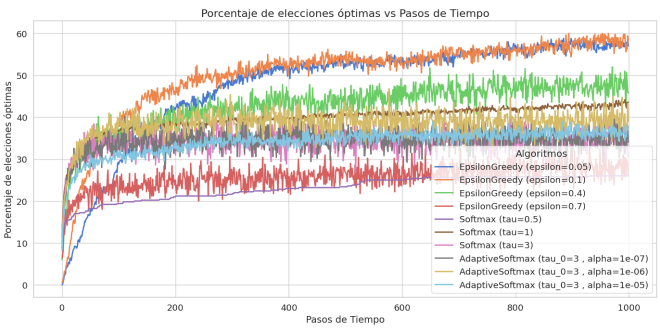

Aunque observamos menor variabilidad en cuanto a porcentaje de elecciones óptimas, el regret acumulado de los algoritmos $UCB$ se comporta de nuevo erráticamente al escoger un bandido de distribución Binomial. Esto se debe a que una distribución Binomial con $n$ grande se aproxima muy bien mediante una distribución Normal. Por eso, al variar $n$ entre 1 y 50, esperamos una mayor similitud con los resultados sobre un bandido de distribución Normal.

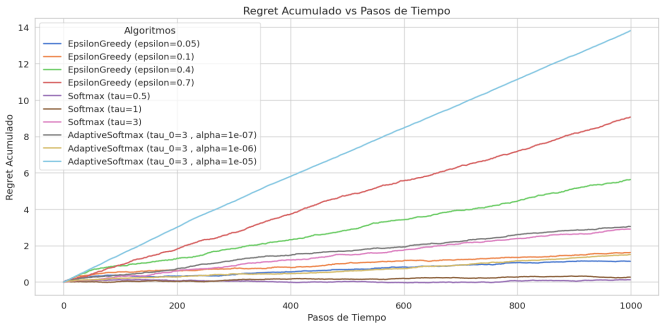
- **Familia ascenso de gradiente**




- **Softmax VS $\varepsilon$-greedy**






Sorprendentemente, en estas gráficas podemos comprobar cómo el rendimiento de los $\varepsilon$-greedy sigue cayendo al cambiar la distribución del bandido por una Binomial desde una Bernoulli. De hecho, aunque a largo plazo tomen mejores decisiones, vemos que a medio plazo (regret acumulado) algunos algoritmos de Softmax se muestran muy superiores a los $\varepsilon$-greedy mejor ajustados. Esto puede deberse a la diferencia de varianza de un brazo Binomial con $n$ intermedio o grande respecto a la varianza fija en 1 para el experimento con brazos normales.

## 5. Conclusiones

En base a todo lo anterior, podemos concluir que las distribuciones de los brazos del bandido afectan considerablemente al rendimiento de la mayoría de los algoritmos, salvo a los algoritmos de gradiente de preferencias, lo cual prueba que son muy robustos en comparación a los demás estudiados, y son los que mejor rendimiento ofrecen en cualquier caso y a corto, medio y largo plazo. Particularmente, en el caso de los algoritmos $\varepsilon$-greedy, hemos observado cómo sus porcentajes de elección óptima decaen globalmente al pasar de distribución Normal a Bernoulli, y de Bernoulli a Binomial. Siendo pues superados en rendimiento por otros algoritmos usando estas dsitribucinoes.

Además, hemos podido comprobar cómo de sensibles son estos algoritmos al ajuste de sus parámetros de exploración, aunque hemos visto que no siempre un agente más exploratorio tiene por qué acumular peores recompensas que uno centrado más en la explotación. De hecho, en múltiples ocasiones se ha mostrado gráficamente cómo algoritmos con menor porcentaje de elecciones de brazo óptimo en la iteración 1000 han presentado un menor regret acumulado en la iteración 1000 que otros algoritmos que seleccionaban con más frecuencia el brazo óptimo.

Esto se debe a la variedad de las distribuciones de los 10 brazos del bandido, y a que podemos encontrar varios brazos sub-óptimos cuyas recompensas promedio no se alejen mucho de la del brazo óptimo.

Asimismo, hemos podido medir el rendimiento de todos estos algoritmos tanto a corto y medio plazo usando el regret acumulado, como a largo plazo echando un vistazo a la evolución del porcentaje de elecciones óptimas. Esto nos da una idea de cómo es el rendimiento de un agente de forma global y permite seleccionar el mejor agente en base a nuestra estrategia de juego (según si queremos jugar poco, mucho o durante un tiempo *intermedio*).

## Referencias

[1] Gil, A. & García, J. & Malest, L. (2025). *Repositorio para la práctica I.* GitHub. Disponible en: https://github.com/JMGO-coding/k_brazos_GGM/blob/main/

[2] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.