Problema del Bandido de k-Brazos

María Castillo Blaya (m.castilloblaya@um.es), Juan José López Quiñonero (jjose.lopez@um.es)

Junio 2025

Abstract

En esta práctica se aborda el problema clásico del bandido de k-brazos, paradigma fundamental en aprendizaje por refuerzo. Se implementan diversos algoritmos (ϵ-greedy, UCB, Gradient Bandit y Thompson Sampling), se diseñan experimentos con distribuciones Bernoulli y normales, y se comparan métricas de rendimiento como recompensa promedio, porcentaje de acción óptima y regret acumulado.

1 Introducción

El problema del bandido de k-brazos (*k-armed bandit*) modela situaciones en las que un agente elige sistemáticamente entre varias opciones (brazos) con recompensas inciertas. Cada selección requiere equilibrar *exploración* (descubrir información) y *explotación* (aprovechar el mejor brazo conocido). Este problema es base para métodos de aprendizaje por refuerzo y aparece en aplicaciones como publicidad en línea y pruebas clínicas.

2 Desarrollo

Consideramos una máquina con k brazos, numerados $i = 1, 2, \ldots, k$. Al accionar el brazo i en el tiempo t, el agente recibe una recompensa aleatoria $R_{i,t}$ extraída de una distribución desconocida P_i con valor esperado $\mu_i = \mathbb{E}[R_{i,t}]$. El objetivo es maximizar la recompensa acumulada:

$$G_T = \sum_{t=1}^{T} R_{A_t,t}, \tag{1}$$

donde A_t es la acción elegida en el paso t. Equivalente es minimizar el regret:

$$R(T) = T\mu^* - \sum_{t=1}^{T} R_{A_t,t}, \quad \mu^* = \max_i \mu_i. \tag{2}$$

2.1 Programa Inicial

2.1.1 Entorno de Trabajo

El código base está desarrollado en Python 3.9, empleando las librerías `NumPy`, `Matplotlib` y `pandas`. Se parte del repositorio `https://github.com/ldaniel-hm/eml_k_bandit` y del notebook `bandit_experiment.ipynb`.

2.1.2 Estructura del Código

El proyecto consta de:

- `bandit.py`: definición de la clase Bandit con métodos de simulación de pasos de tiempo.
- `algorithms.py`: implementaciones de agentes ϵ-greedy, UCB1, Softmax y Thompson Sampling.
- `experiments.py`: funciones para ejecutar experimentos repetidos y recopilar métricas.
- `plots.py`: generación de gráficos de rendimiento.

3 Evaluación/Experimentos

3.1 Introducción al Problema de los K-Brazos y Métodos de Aprendizaje por Refuerzo

La primera parte se centra en la implementación y comparación de tres familias de algoritmos para el problema del bandido de k brazos: métodos -Greedy, Upper Confidence Bound (UCB) y algoritmos de ascenso de gradiente. Sus objetivos son poner en práctica cada agente, evaluar su comportamiento frente a distribuciones de recompensa Bernoulli, Normal y Binomial, y medir tres métricas clave: la recompensa promedio obtenida, el porcentaje de veces que se elige el brazo óptimo y el regret acumulado.

Los resultados muestran que un valor intermedio de (aproximadamente 0,1) logra el mejor equilibrio entre exploración y explotación, obteniendo recompensas elevadas sin incurrir en un regret excesivo. UCB1 destaca por reducir muy rápidamente el regret en las primeras etapas, gracias a su

estrategia sistemática de exploración de brazos poco probados. Los algoritmos de ascenso de gradiente, por su parte, convergen más rápido en entornos con recompensas continuas (distribuciones normales), demostrando además mayor robustez frente a la varianza de las recompensas. Finalmente, la naturaleza de la distribución influye de forma notable: en escenarios Bernoulli, los métodos basados en probabilidades (como Thompson Sampling o UCB) resultan más estables, mientras que en distribuciones normales los enfoques de gradiente ofrecen una pequeña ventaja. En conjunto, estos hallazgos subrayan que la elección del algoritmo óptimo depende tanto del trade-off exploración/explotación como de las características estadísticas del entorno de recompensas.

3.2 Estudio comparativo de algoritmos epsilon-greedy en un problema de k-armed bandit

Se estudian tres configuraciones del método ϵ-Greedy (ϵ=0, 0.01 y 0.1) en un horizonte de 1 000 pasos y 2 000 repeticiones, con el objetivo de ver cómo la tasa de exploración afecta a las métricas de rendimiento: recompensa promedio, porcentaje de selección del brazo óptimo y regret acumulado.

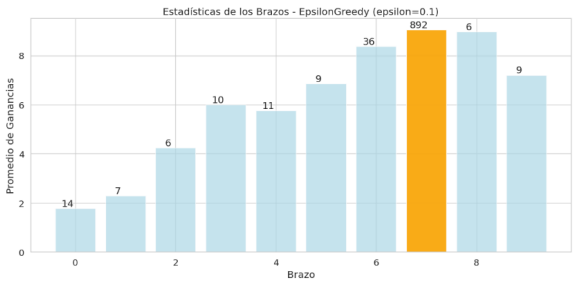

Figure 1: Estadísticas de selección de brazos para ϵ-greedy para distribución Normal con $\epsilon = 0{,}1$

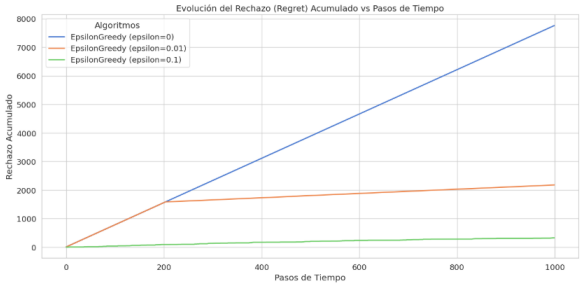

Figure 2: Regret acumulado en ϵ-greedy para distribución Normal.

En conclusión,

- ϵ = 0 (explotación pura): converge rápido pero queda atrapado en óptimos locales, obteniendo recompensas subóptimas y un regret alto.
- ϵ = 0.01: explora muy poco; mejora ligeramente sobre =0, pero la lenta incorporación de nueva información retrasa la convergencia.
- ϵ = 0.1: consigue la recompensa final más elevada y el regret acumulado más bajo, al equilibrar adecuadamente exploración y explotación.

En resumen, para entornos estocásticos con estructura desconocida, valores de ϵ en torno a 0.1 resultan recomendables para maximizar la eficiencia del agente.

3.3 Estudio comparativo de algoritmos UCB en un problema de k-armed bandit

Se evalúa y compara el desempeño de las variantes UCB1 y UCB2 en el problema de los múltiples brazos (multi-armed bandit) bajo incertidumbre, analizando cómo la elección de sus hiperparámetros (el factor de exploración c en UCB1 y el parámetro α en UCB2) y la naturaleza de las distribuciones de recompensa (Bernoulli, Binomial y Normal) influyen en la recompensa promedio y el regret; para ello, se diseña un experimento robusto de 1 000 pasos repetidos 500 veces, con el fin de obtener conclusiones estadísticamente significativas sobre el equilibrio entre exploración y explotación.

Figure 3: Recompensa promedio para UCB (distribución Normal).

Los resultados demuestran que:

- El rendimiento de UCB1 y UCB2 depende críticamente de la distribución de recompensas y de la elección de sus parámetros.
- UCB1 (c = 0.5) muestra gran solidez en entornos con recompensas continuas (por ejemplo, distribuciones normales), gracias a su balance estable entre exploración y explotación.
- UCB2 (α = 0.9) destaca en entornos discretos o muy ruidosos (Bernoulli, Binomial), donde la fórmula adaptativa de sus intervalos de confianza le da ventaja.
- Configuraciones de alta exploración (por ejemplo, c = 2 en UCB1 o α = 0.1 en UCB2) incrementan el regret y reducen la eficiencia global, pues dedican demasiados pasos a explorar brazos subóptimos.

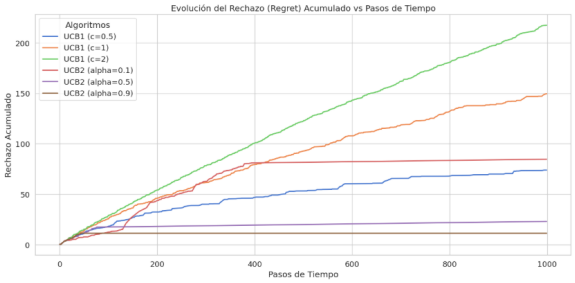

Figure 4: Regret acumulado para UCB (distribución de Bernoulli.

En definitiva, la selección adecuada del algoritmo y sus hiperparámetros es esencial para maximizar la recompensa promedio en problemas de decisión secuencial bajo incertidumbre.

3.4 Notebook4. Estudio comparativo de algoritmos de ascenso de gradiente en un problema de k-armed bandit

Por último, se compara el rendimiento de dos algoritmos de ascenso de gradiente -Gradient Bandit y Softmax- en un problema de k-armed bandit, estudiando cómo distintos valores del parámetro de temperatura en Softmax afectan la recompensa promedio y el regret; para ello evalúa ambos métodos bajo tres tipos de distribuciones de recompensa (Bernoulli, Binomial y Normal) y presenta sus resultados mediante gráficas de recompensas promedio que permitan identificar qué estrategia y configuración hiperparamétrica es más adecuada en cada contexto.

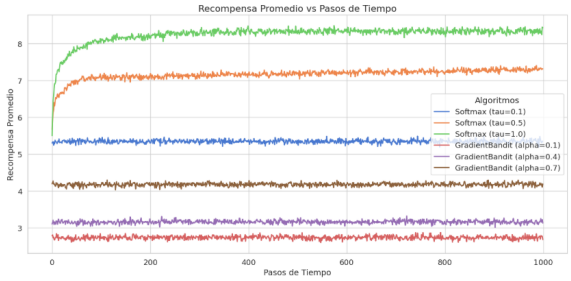

Figure 5: Recompensa promedio para Ascenso de Gradiente en distribución Normal.

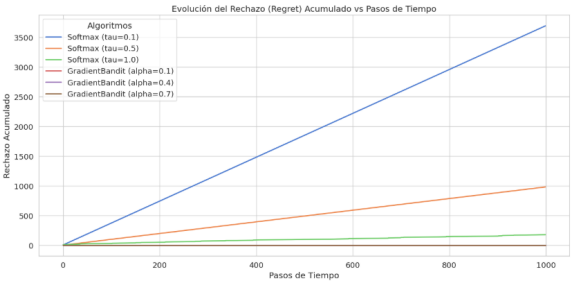

Figure 6: Regret acumulado para Ascenso de Gradiente en distribución Binomial.

Se concluye que:

- Gradient Bandit consigue el menor regret en entornos discretos (Bernoulli y Binomial), destacando su robustez cuando las recompensas son binarias o de recuento.
- Softmax (τ = 1.0) ofrece el mejor rendimiento en entornos con recompensas continuas (distribución

Normal), gracias a un balance más suave entre exploración y explotación.

- En Binomial, tanto Gradient Bandit como Softmax con $\tau = 1.0$ son competitivos, reflejando que un nivel medio de exploración basta para este tipo de ruido.
- Para Bernoulli, los mejores resultados se obtienen con Gradient Bandit o Softmax con una temperatura baja ($\tau = 0.1$), que prioriza la explotación de brazos prometedores.
- En Normal, Softmax con $\tau = 1.0$ domina, puesto que su exploración gradual capta mejor las variaciones continuas de las recompensas.

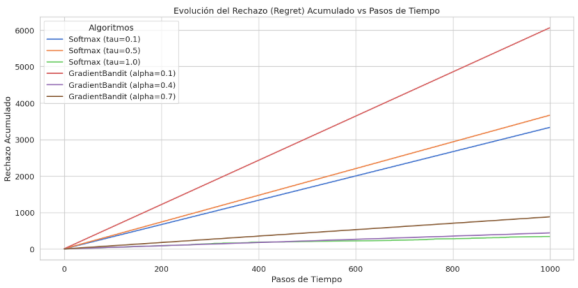

Figure 7: Regret acumulado para Ascenso de Gradiente en distribución Normal.

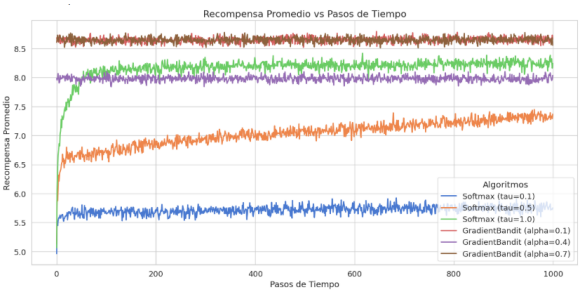

Figure 8: Recompensa promedio para Ascenso de Gradiente en distribución Binomial.

En resumen, Gradient Bandit es ideal para problemas con recompensas discretas, mientras que Softmax requiere un ajuste fino de τ para adaptarse eficazmente a cada distribución.

4 Conclusiones

Los métodos bayesianos (Thompson Sampling) demostraron ser los más eficientes en entornos con recompensas Bernoulli. UCB1 es una buena alternativa sin necesidad de parámetros externos. Epsilon-Greedy, aunque simple, requiere calibrar ϵ cuidadosamente. Gradient Bandit ofrece ventajas en distribuciones continuas.

References

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

[2] Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2), 235-256.

[3] Lattimore, T., & Szepesvári, C. (2020). *Bandit Algorithms*. Cambridge University Press.

A Código Ejemplar

A continuación se muestra un fragmento de la implementación de Thompson Sampling para distribuciones Bernoulli:

```
class ThompsonBernoulli:
    def __init__(self, k):
        self.k = k
        self.alphas = [1]*k
        self.betas = [1]*k

    def select_arm(self):
        samples = [np.random.beta(a, b) for a, b in zip(
        return int(np.argmax(samples))

    def update(self, chosen, reward):
        self.alphas[chosen] += reward
        self.betas[chosen] += 1 - reward
```