UNIVERSIDAD DE MURCIA

FACULTAD DE INGENIERÍA INFORMÁTICA

MASTER EN INTELIGENCIA ARTIFICIAL

UNIVERSIDAD
DE MURCIA

Extensiones de Machine Learning

Práctica 1

Profesor:

Luis Daniel Hernández Molinero

Estudiantes:

Antonio Marín Ortega (antonio.marino@um.es)

Darío Escudero de Paco (dario.escuderop@um.es)

Francisco Javier Pérez Pujalte (fj.perezpujalte@um.es)

Índice

1. Introducción **1**

2. Algoritmos propuestos **1**
2.1. ε-greedy 1
2.2. Algoritmo UCB (UCB1 y UCB2) 2
2.3. Algoritmos Softmax y Gradiente de Preferencias 3

3. Evaluación y Experimentos **4**

4. Resultados de la familia ε-greedy **5**
4.1. Distribución Normal 5
4.2. Distribución Bernoulli 6
4.3. Distribución Binomial 6
4.4. Conclusión general 6

5. Resultados de la familia UCB **6**
5.1. Distribución Normal 7
5.2. Distribución Bernoulli 7
5.3. Distribución Binomial 7
5.4. Conclusión general 7

6. Resultados de la familia Gradient (Softmax y Gradient Bandit) **7**
6.1. Distribución Normal 8
6.2. Distribución Bernoulli 8
6.3. Distribución Binomial 8
6.4. Conclusión general 8

7. Conclusiones generales **9**

1. Introducción

El problema del **bandido de k brazos** modela una situación clásica de toma de decisiones secuencial bajo incertidumbre. Se imagina una máquina tragaperras con k palancas (brazos), cada una con una distribución de recompensa fija pero desconocida. En cada turno t, el agente debe elegir un brazo a_t ($1 \leq a_t \leq k$) y recibe una recompensa r_t extraída de la distribución del brazo seleccionado. El objetivo del agente es maximizar la recompensa acumulada $\sum_{t=1}^{T} r_t$ a lo largo de T jugadas. Equivalentemente, se puede medir la *pérdida de rendimiento* frente a la estrategia óptima mediante el **rechazo** (*regret*), definido como $R(T) = T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} \mu_{a_t}\right]$, donde $\mu_i = \mathbb{E}[r \mid a = i]$ es la recompensa media del brazo i y $\mu^* = \max_i \mu_i$ es la media del mejor brazo. Este valor $R(T)$ cuantifica la pérdida esperada por no haber elegido siempre el brazo óptimo; un algoritmo ideal tendría $R(T)$ mínimo.

El bandido k-brazos ejemplifica el dilema **exploración-explotación**: el agente debe *explotar* el brazo que cree mejor para obtener recompensa, pero a la vez debe *explorar* otros brazos para obtener información de recompensa potencialmente mayor en el futuro. Encontrar el equilibrio óptimo entre explorar y explotar es fundamental para maximizar la recompensa acumulada. Este problema tiene aplicaciones en campos como la optimización de publicidad online, pruebas A/B, sistemas de recomendación, entre otros, donde se debe aprender qué opción produce la mejor recompensa mediante interacción secuencial con el entorno.

En la introducción de este informe se presenta el problema y su relevancia. A continuación, en la sección de **Desarrollo** se expone el contexto teórico y las técnicas existentes para enfrentar este dilema. La sección de **Algoritmos** detalla, con pseudocódigo y justificación, las familias de métodos estudiadas: ε-greedy, métodos UCB y métodos basados en gradiente (softmax/preferencias). Posteriormente, en la sección de **Evaluación** se describen los experimentos realizados en distintos entornos de recompensa (Bernoulli, Binomial y Normal), junto con los resultados obtenidos, gráficos comparativos y análisis de desempeño (recompensas obtenidas, frecuencia de selección del mejor brazo, evolución del regret, etc.). Finalmente, las **Conclusiones** resumen los hallazgos principales, discuten las diferencias entre los enfoques y recomiendan cuáles gráficas son más informativas para evaluar estos algoritmos.

2. Algoritmos propuestos

En esta sección se describen los algoritmos implementados para el problema de *multi-armed bandits*, agrupados por familia, junto a su pseudocódigo esencial. Todos los algoritmos asumen k brazos y se ejecutan durante un horizonte de T pasos.

2.1. ε-greedy

El método ε-greedy alterna entre exploración y explotación de forma simple: con probabilidad ε elige un brazo al azar (exploración), y con probabilidad $1 - \varepsilon$ selecciona el brazo con mayor estimación actual de recompensa (explotación). Las estimaciones de recompensa $\hat{\mu}_i$ para cada brazo i se actualizan de forma incremental tras cada jugada:

$$\hat{\mu}_{a_t} \leftarrow \hat{\mu}_{a_t} + \frac{1}{n_{a_t}} \left(r_t - \hat{\mu}_{a_t} \right)$$

donde n_{a_t} es el número de veces que el brazo a_t ha sido seleccionado hasta el paso t. Este esquema permite aproximar la esperanza real de recompensa de cada brazo conforme se acumulan observaciones.

Inicializar $\hat{\mu}_i \leftarrow 0$, $n_i \leftarrow 0$ para $i = 1, \ldots, k$;
for $t = 1$ **to** T **do**
  Generar $u \sim \mathcal{U}[0, 1]$;
  if $u < \varepsilon$ **then**
    Seleccionar un brazo a_t al azar uniforme;
  else
    Seleccionar $a_t = \arg\max_i \hat{\mu}_i$;
  end
  Observar recompensa r_t de a_t;
  Actualizar $n_{a_t} \leftarrow n_{a_t} + 1$;
  Actualizar $\hat{\mu}_{a_t} \leftarrow \hat{\mu}_{a_t} + \frac{1}{n_{a_t}}(r_t - \hat{\mu}_{a_t})$;
end
return $\{\hat{\mu}_1, \ldots, \hat{\mu}_k\}$

Algorithm 1: ε-greedy(ε, k, T)

En nuestros experimentos comparamos varios niveles de exploración: $\varepsilon = 0$ (estrategia puramente greedy sin exploración), $\varepsilon = 0{,}01$ (muy conservador), y $\varepsilon = 0{,}1$ (exploración moderada). Estos valores ilustran cómo el grado de exploración inicial afecta al rendimiento acumulado (*regret*).

2.2. Algoritmo UCB (UCB1 y UCB2)

El algoritmo **UCB1** (Upper Confidence Bound) selecciona en cada paso el brazo que maximiza el índice:

$$I_i(t) = \hat{\mu}_i + c\sqrt{\frac{2 \ln t}{n_i}}$$

donde c controla el nivel de exploración adicional. Inicialmente, cuando $n_i = 0$, se define $I_i = +\infty$ para forzar que cada brazo sea jugado al menos una vez (fase de inicialización).

Inicializar $\hat{\mu}_i \leftarrow 0$, $n_i \leftarrow 0$ para $i = 1, \ldots, k$;
for $t = 1$ **to** T **do**
  for *cada brazo* $i = 1, \ldots, k$ **do**
    if $n_i = 0$ **then**
      $I_i \leftarrow +\infty$;
    else
      $I_i \leftarrow \hat{\mu}_i + c\sqrt{\frac{\ln t}{n_i}}$;
    end
  end
  Seleccionar $a_t = \arg\max_i I_i$;
  Observar recompensa r_t de a_t;
  Actualizar $n_{a_t} \leftarrow n_{a_t} + 1$;
  Actualizar $\hat{\mu}_{a_t} \leftarrow \hat{\mu}_{a_t} + \frac{1}{n_{a_t}}(r_t - \hat{\mu}_{a_t})$;
end

Algorithm 2: UCB1(c, k, T)

El parámetro c modula el equilibrio entre exploración y explotación. Teóricamente, $c = 1$ garantiza las cotas logarítmicas de *regret* Auer et al., 2002, pero en la práctica puede ajustarse según el nivel de ruido de las recompensas. En nuestros experimentos probamos $c \in \{0{,}01,\ 1{,}0,\ 3{,}0\}$, observando que valores altos promueven mayor exploración inicial, mientras que valores bajos hacen que UCB1 se comporte de forma más explotadora, aproximándose a un ε-greedy casi determinista.

El algoritmo **UCB2** introduce un parámetro adicional $\alpha \in (0, 1)$ que controla el tamaño de los periodos (bloques) de repetición de cada brazo. En lugar de recalcular el índice en cada paso, UCB2 selecciona un brazo y lo juega durante varios pasos consecutivos antes de volver a evaluar los índices, amortiguando así la variabilidad de las estimaciones. A valores de α pequeños, UCB2 se aproxima al comportamiento de UCB1. En nuestras pruebas empleamos $\alpha \in \{0{,}01,\ 0{,}3,\ 0{,}5\}$ siguiendo la notación estándar de Auer et al., 2002. Observamos que UCB1 y UCB2 ofrecen rendimientos comparables, siendo UCB2 ligeramente superior en la mayoría de escenarios evaluados.

2.3. Algoritmos Softmax y Gradiente de Preferencias

En el método **softmax clásico**, las estimaciones de recompensa $\hat{\mu}_i$ de cada brazo se utilizan directamente para definir una política probabilística:

$$\pi_t(i) = \frac{\exp(\hat{\mu}_i/\tau)}{\sum_j \exp(\hat{\mu}_j/\tau)}$$

En cada paso, se selecciona un brazo de acuerdo a esta distribución (por ejemplo, generando un número uniforme aleatorio y eligiendo la categoría según las probabilidades acumuladas). Tras observar la recompensa r_t, se actualiza la estimación $\hat{\mu}_{a_t}$ mediante promedio incremental:

$$\hat{\mu}_{a_t} \leftarrow \hat{\mu}_{a_t} + \frac{1}{n_{a_t}}(r_t - \hat{\mu}_{a_t})$$

Este enfoque no utiliza gradientes explícitos; simplemente transforma las estimaciones actuales en una política estocástica mediante la función softmax. La temperatura τ controla el grado de

aleatoriedad: valores bajos tienden a políticas casi deterministas, mientras que valores altos promueven mayor exploración. En nuestros experimentos consideramos $\tau \in \{0{,}1, 1{,}0\}$ para comparar comportamientos más explotadores o exploratorios.

El algoritmo de **Gradiente de Preferencias** (o *Gradient Bandit*) realiza aprendizaje directo sobre los parámetros de la política. Inicializamos las *preferencias* $H_i = 0$ para cada brazo, que definen la política softmax:

$$\pi_t(i) = \frac{\exp(H_i)}{\sum_j \exp(H_j)}$$

Además, mantenemos un promedio móvil de las recompensas observadas $\bar{r}_t$, actualizado de forma incremental:

$$\bar{r}_t \leftarrow \bar{r}_{t-1} + \frac{1}{t}(r_t - \bar{r}_{t-1})$$

Tras observar la recompensa r_t, se actualizan las preferencias como:

$$H_{a_t} \leftarrow H_{a_t} + \alpha(r_t - \bar{r}_t)(1 - \pi_t(a_t))$$
$$H_j \leftarrow H_j - \alpha(r_t - \bar{r}_t)\pi_t(j), \quad \forall j \neq a_t$$

Este algoritmo realiza, en promedio, un ascenso de gradiente en la recompensa esperada (Sutton y Barto, 2018). El parámetro α es crítico: valores excesivamente grandes pueden inducir oscilaciones en las preferencias. El baseline $\bar{r}_t$ ayuda a reducir la varianza de los gradientes: al sustraer el promedio, sólo las recompensas por encima del promedio incrementan la preferencia del brazo jugado.

3. Evaluación y Experimentos

En esta sección se describen los experimentos realizados (repositorio Github, 2025) para comparar las distintas familias de algoritmos. Se utilizó un entorno de simulación desarrollado en Python (disponible en los notebooks entregados) que permite definir un problema de k-brazos con diferentes distribuciones de recompensa para los brazos. En concreto, evaluamos cada algoritmo en tres escenarios:

- **Recompensas Bernoulli**: cada brazo i produce recompensa 1 con probabilidad p_i o 0 con probabilidad $1 - p_i$. Es decir, $r_t \sim \text{Bernoulli}(p_i)$. Equivale a un clic (éxito) o no clic (fracaso) en un ejemplo de publicidad online.
- **Recompensas Binomiales**: cada brazo tiene un número fijo n de intentos por jugada y probabilidad p_i de éxito en cada intento, retornando el número de éxitos. $r_t \sim \text{Binomial}(n, p_i)$. En nuestro experimento tomamos $n = 10$; así un brazo con $p_i = 0{,}5$ tiene media $\mu_i = 5$. La distribución Binomial generaliza la Bernoulli (que es el caso $n = 1$).
- **Recompensas Normales**: cada brazo i da recompensas $r_t \sim \mathcal{N}(\mu_i, \sigma^2)$ aproximadamente. Implementamos esto directamente; por ejemplo, un brazo óptimo podría tener $\mu^* \approx 8$ con $\sigma = 1$. Esta configuración es comparable a la binomial gracias a la aproximación normal para $\text{Bin}(n, p)$ cuando n es suficientemente grande. Usamos distribuciones normales para ver el desempeño cuando las recompensas no están acotadas en $[0, 1]$ sino en un continuo (aunque limitamos σ moderadamente para evitar valores extremos).

En todos los casos, fijamos $k = 10$ brazos y seleccionamos las medias $\{\mu_i\}$ de forma aleatoria al comienzo de cada ejecución (y luego mantenidas fijas durante la ejecución). Para cada algoritmo y escenario, realizamos 500 ejecuciones independientes (*runs*) de $T = 1000$ pasos cada una, con diferente semilla, para estimar el rendimiento promedio. Emplear múltiples ejecuciones permite obtener resultados promediados más estables y medir la variabilidad.

Se analizaron cuatro métricas principales a lo largo de los pasos de tiempo:

- **Recompensa promedio por paso:** $\frac{1}{\text{runs}} \sum_{\text{ejec}} r_t$ en el paso t. Mide cuánto va ganando en promedio el algoritmo conforme aprende.
- **% de selección del brazo óptimo:** frecuencia con la que el brazo con mayor μ_i real fue seleccionado hasta el tiempo t. Tiende a 100 % si el algoritmo identifica correctamente el mejor brazo.
- **Rechazo acumulado (regret) por paso:** aquí calculado como $\frac{R(t)}{t}$, es decir, el regret promedio por jugada hasta t. Una estrategia óptima debería ver este valor decaer hacia 0 (pues a largo plazo casi siempre escoge el brazo óptimo).
- **Estadísticas de brazos:** al finalizar, registramos para cada brazo cuántas veces fue jugado y cuál fue su recompensa media obtenida. Esto se presenta en histogramas para entender la distribución de esfuerzos del algoritmo en exploración vs explotación.

A continuación, presentamos resultados destacados para cada familia de métodos, discutiendo cuáles gráficos aportan más información y comparando el comportamiento en las diferentes distribuciones de recompensa.

4. Resultados de la familia ε-greedy

Se evaluó el algoritmo ε-greedy sobre un problema de $k = 10$ brazos, considerando distribuciones Normal, Bernoulli y Binomial. Se realizaron 500 ejecuciones de 1000 pasos cada una. Los valores de ε evaluados fueron 0,0 (greedy puro), 0,01 (muy conservador) y 0,1 (exploración moderada).

4.1. Distribución Normal

El brazo óptimo fue el 8, con recompensa esperada de 8,73. Las recompensas medias alcanzadas fueron:

- $\varepsilon = 0{,}0$: 1,343
- $\varepsilon = 0{,}01$: 6,629
- $\varepsilon = 0{,}1$: 8,096

El algoritmo greedy puro ($\varepsilon = 0{,}0$) se quedó atrapado frecuentemente en brazos subóptimos debido a malas inicializaciones. La exploración con $\varepsilon = 0{,}01$ permitió corregir parcialmente estos errores iniciales, mientras que $\varepsilon = 0{,}1$ consiguió aproximarse muy cerca del valor óptimo.

4.2. Distribución Bernoulli

El brazo óptimo fue el 5, con recompensa esperada de 0,867. Las recompensas medias fueron:

- $\varepsilon = 0{,}0$: 0,395
- $\varepsilon = 0{,}01$: 0,687
- $\varepsilon = 0{,}1$: 0,788

La mayor varianza inherente a las recompensas Bernoulli penalizó severamente la ausencia de exploración en el caso $\varepsilon = 0{,}0$. Incluso un $\varepsilon = 0{,}01$ permitió obtener mejoras significativas, pero nuevamente la mejor exploración fue con $\varepsilon = 0{,}1$, que alcanzó más del 90 % de la recompensa óptima esperada.

4.3. Distribución Binomial

El brazo óptimo fue el 10, con recompensa esperada de 11,738. Las recompensas medias fueron:

- $\varepsilon = 0{,}0$: 3,348
- $\varepsilon = 0{,}01$: 8,469
- $\varepsilon = 0{,}1$: 10,653

En este caso, incluso en presencia de mayor rango de recompensas, la falta de exploración produjo bloqueos severos en brazos inferiores cuando $\varepsilon = 0{,}0$. La exploración progresiva con $\varepsilon = 0{,}01$ y especialmente con $\varepsilon = 0{,}1$ permitió acercarse a rendimientos muy próximos al óptimo.

4.4. Conclusión general

La familia ε-greedy mostró que un nivel de exploración bajo (e.g. $\varepsilon = 0{,}01$) puede mejorar sustancialmente los resultados respecto a greedy puro, pero para obtener convergencia rápida hacia el óptimo en problemas ruidosos es preferible un ε moderado (e.g. $\varepsilon = 0{,}1$).

5. Resultados de la familia UCB

Se evaluaron los algoritmos **UCB1** y **UCB2** en un problema de $k = 10$ brazos, bajo distribuciones Normal, Bernoulli y Binomial. Cada configuración se ejecutó 500 veces durante 1000 pasos. Los parámetros explorados fueron:

- UCB1: $c \in \{0{,}01,\ 1{,}0,\ 3{,}0\}$
- UCB2: $\alpha \in \{0{,}01,\ 0{,}3,\ 0{,}5\}$

5.1. Distribución Normal

El brazo óptimo fue el 8, con recompensa esperada 8,73. Los resultados obtenidos fueron:

- UCB1: 8,597 ($c = 0{,}01$), 8,678 ($c = 1{,}0$), 8,556 ($c = 3{,}0$)
- UCB2: 8,654 ($\alpha = 0{,}01$), 8,680 ($\alpha = 0{,}3$), 8,680 ($\alpha = 0{,}5$)

Todos los algoritmos convergieron cerca del óptimo. UCB1 con $c = 1{,}0$ y UCB2 con $\alpha = 0{,}3$ o 0,5 lograron el mejor equilibrio exploración-explotación.

5.2. Distribución Bernoulli

El brazo óptimo fue el 5, con recompensa esperada 0,867. Los resultados fueron:

- UCB1: 0,840 ($c = 0{,}01$), 0,792 ($c = 1{,}0$), 0,638 ($c = 3{,}0$)
- UCB2: 0,667 ($\alpha = 0{,}01$), 0,804 ($\alpha = 0{,}3$), 0,800 ($\alpha = 0{,}5$)

La varianza alta de Bernoulli penalizó las configuraciones con sobre-exploración ($c = 3{,}0$, $\alpha = 0{,}01$). UCB1 con $c = 0{,}01$ y UCB2 con $\alpha = 0{,}3$ alcanzaron los mejores rendimientos.

5.3. Distribución Binomial

El brazo óptimo fue el 10, con recompensa esperada 11,738. Los resultados fueron:

- UCB1: 11,670 ($c = 0{,}01$), 11,669 ($c = 1{,}0$), 11,619 ($c = 3{,}0$)
- UCB2: 11,668 ($\alpha = 0{,}01$), 11,669 ($\alpha = 0{,}3$), 11,668 ($\alpha = 0{,}5$)

En este caso, todos los algoritmos lograron rendimientos prácticamente idénticos, cercanos al óptimo.

5.4. Conclusión general

UCB mostró alta robustez adaptativa:

- En Gaussianas y Binomiales, cualquier configuración razonable alcanzó el óptimo.
- En Bernoulli, el exceso de exploración penalizó el rendimiento.
- UCB1 con $c = 1{,}0$ y UCB2 con $\alpha = 0{,}3$ ofrecieron los mejores compromisos globales.

6. Resultados de la familia Gradient (Softmax y Gradient Bandit)

Se evaluaron los algoritmos **Softmax Bandit** y **Gradient Bandit** sobre un problema de $k = 10$ brazos, con distribuciones Normal, Bernoulli y Binomial. Se realizaron 500 ejecuciones de 1000 pasos cada una. Los parámetros evaluados fueron:

- Softmax: $\tau \in \{0{,}01,\ 0{,}5,\ 1{,}0\}$
- Gradient Bandit (con baseline): $\alpha \in \{0{,}01,\ 0{,}05,\ 0{,}1\}$

6.1. Distribución Normal

El brazo óptimo fue el 8, con recompensa esperada 8,73. Las recompensas medias obtenidas fueron:

- Softmax: 4,957 ($\tau = 0{,}01$), 6,845 ($\tau = 0{,}5$), 7,616 ($\tau = 1{,}0$)
- Gradient Bandit: 7,361 ($\alpha = 0{,}01$), 8,336 ($\alpha = 0{,}05$), 8,415 ($\alpha = 0{,}1$)

Todos los algoritmos lograron aproximarse al óptimo, destacando Gradient Bandit con $\alpha = 0{,}1$ como el más eficiente, seguido de Softmax con $\tau = 1{,}0$.

6.2. Distribución Bernoulli

El brazo óptimo fue el 5, con recompensa esperada 0,867. Los resultados fueron:

- Softmax: 0,629 ($\tau = 0{,}01$), 0,601 ($\tau = 0{,}5$), 0,512 ($\tau = 1{,}0$)
- Gradient Bandit: 0,468 ($\alpha = 0{,}01$), 0,640 ($\alpha = 0{,}05$), 0,729 ($\alpha = 0{,}1$)

En este entorno ruidoso, el mejor rendimiento fue para GradientBandit con $\alpha = 0{,}1$, seguido de Gradient Bandit con $\alpha = 0{,}5$. Los valores bajos de τ y α mostraron dificultades de exploración.

6.3. Distribución Binomial

El brazo óptimo fue el 10, con recompensa esperada 11,738. Los resultados fueron:

- Softmax: 6,112 ($\tau = 0{,}01$), 8,153 ($\tau = 0{,}5$), 9,169 ($\tau = 1{,}0$)
- Gradient Bandit: 10,291 ($\alpha = 0{,}01$), 11,283 ($\alpha = 0{,}05$), 11,297 ($\alpha = 0{,}1$)

En este caso, Gradient Bandit superaron ampliamente a Softmax, convergiendo casi al óptimo para valores de $\alpha = 0{,}05$ y $\alpha = 0{,}1$.

6.4. Conclusión general

Los algoritmos de gradiente demostraron ser altamente competitivos:

- Gradient Bandit con $\alpha = 0{,}05$ y $\alpha = 0{,}1$ ofreció el mejor comportamiento global.
- Softmax mostró alta sensibilidad a la temperatura: $\tau = 0{,}01$ fue sistemáticamente insuficiente.
- En problemas con alta varianza inicial (como Bernoulli), los ajustes de $\alpha = 0{,}05$ y $\alpha = 0{,}1$ proporcionaron el mejor equilibrio entre exploración y explotación.

7. Conclusiones generales

En este trabajo se ha realizado un estudio comparativo exhaustivo de distintas familias de algoritmos para el problema clásico de *k-armed bandit*, analizando su comportamiento bajo entornos con distintas características de ruido y complejidad de recompensa: distribuciones Normal, Bernoulli y Binomial.

A lo largo de los experimentos se han observado de forma consistente los siguientes patrones globales:

- **Importancia de la exploración inicial:** Todos los algoritmos que no introducen exploración sistemática (caso $\varepsilon = 0$ o $\tau = 0{,}01$) mostraron claras dificultades para identificar el brazo óptimo cuando las primeras selecciones fueron desfavorables. Incluso pequeñas cantidades de exploración (e.g., $\varepsilon = 0{,}01$, $\tau = 0{,}5$, $c = 0{,}01$) permitieron mejoras sustanciales.
- **Rendimiento robusto de UCB:** Los algoritmos UCB1 y UCB2 demostraron gran capacidad adaptativa, logrando rápida convergencia en entornos Gaussianos y Binomiales, y manteniendo competitividad en Bernoulli, especialmente con configuraciones intermedias ($c = 1{,}0$, $\alpha = 0{,}3$). La estrategia basada en índices optimistas ha resultado especialmente eficaz para evitar bloqueos prematuros.
- **Sensibilidad de Softmax:** El algoritmo Softmax mostró fuerte dependencia de la temperatura τ. Temperaturas muy bajas condujeron a políticas excesivamente deterministas y subóptimas. Valores de $\tau = 1{,}0$ proporcionaron el mejor equilibrio global en la mayoría de los entornos.
- **Potencial de los métodos de gradiente:** Gradient Bandit, ajustando directamente las preferencias, ha demostrado ser muy competitivo especialmente en los entornos Normal y Binomial. La combinación de baseline y tasas de aprendizaje intermedias ($\alpha = 0{,}05$ y $\alpha = 0{,}1$) ha permitido alcanzar rendimientos sistemáticamente cercanos al óptimo con baja variabilidad y rápido decrecimiento del regret.
- **Entornos ruidosos (Bernoulli):** Estos han representado el mayor desafío para todos los algoritmos. La exploración insuficiente o el sobreajuste inicial penalizaron el rendimiento. No obstante, las configuraciones intermedias de todos los métodos consiguieron estabilizarse hacia valores cercanos al óptimo.

En conjunto, el estudio confirma que no existe un único algoritmo óptimo universal, sino que la elección de parámetros de exploración es crítica y dependiente del tipo de entorno. Sin embargo, UCB (ambas implementaciones) han resultado ser las más robustas en los tres tipos de brazos ganando al resto de implementaciones.

Referencias

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multi-armed bandit problem. *Machine learning*, *47*(2-3), 235-256.

Github, R. (2025). k_brazos_AM_DE_JP [Repositorio en GitHub]. https://github.com/ermaury/k_brazos_AM_DE_JP

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (Second edition). MIT press.