Modelos de machine learning para Aprendizaje Reforzado: Selección de Tipos de Comida

Karla Orozco
Jonathan Zapata
Juan Esteban Chavarria
Juan Fernando Gallego

Maestría en ciencias de los datos y analítica,
Aprendizaje Automático Avanzado, 2022-1,
Universidad Eafit, Medellín, Colombia

Resumen

Este trabajo presenta los fundamentos teóricos y técnicos sobre el aprendizaje reforzado y como se aborda desde un ejemplo sencillo pero que permite un mejor entendimiento sobre este tema. Se construye a través de la metodología CRISP-DM, y se hace referencia a los enfoques con los que se trabaja el aprendizaje reforzado o LR(Learning Refoircement), método empleado en las diferentes investigaciones de los diversos autores y sus enfoques de solución para el problema en cuestión.

1. Introducción

Debido a los avances que se han presentado en los diversos ámbitos tecnológicos, con respecto a los dispositivo, máquinas y herramientas que ejecutan tareas por si mismos, surge la necesidad de que los algoritmos sean auto-entrenados, otorgando pesos o características de acuerdo a los errores en las decisiones que van tomando[1].

A través de los años, la necesidad biológica del ser humano para aprender por si mismo, lo ha llevado a aplicar esta modalidad de aprendizaje, el cual en primera instancia, se toma una decisión y se evalúa que tan buena o mala fue y en la siguiente oportunidad se toma una decisión diferente, esperado un mejor resultado; a este tipo de aprendizaje se le llama aprendizaje por reforzamiento y su finalidad es aprender a identificar las acciones que llevan a la toma de buenas decisiones. De esta manera podemos observar que el los diferentes ecosistemas los organismos actúan de esta manera, y así, van identificando nuevos lugares de donde proveerse de alimento o donde migrar de acuerdo a los cambios climáticos [2].

Actualmente, existen algoritmos como el algoritmo codicioso o el de los k bandidos en los cuales, en el primero, su objetivo principal es seleccionar de forma adaptativa operadores de alteración apropiados en cada paso del proceso de búsqueda, en función del estado de la búsqueda y el historial de rendimiento de los operadores [3]. En el segundo, su objetivo es seleccionar una acción de un conjunto de decisiones determinado; en donde el oponente o la parte contraria luego observa la acción presentada y selecciona una función de pérdida. Después de eso, se conoce la función de pérdida y posiblemente se sufra una pérdida [4].

A continuación se presenta la Figura 1, la cual permite comprender como es el comportamiento del aprendizaje reforzado.

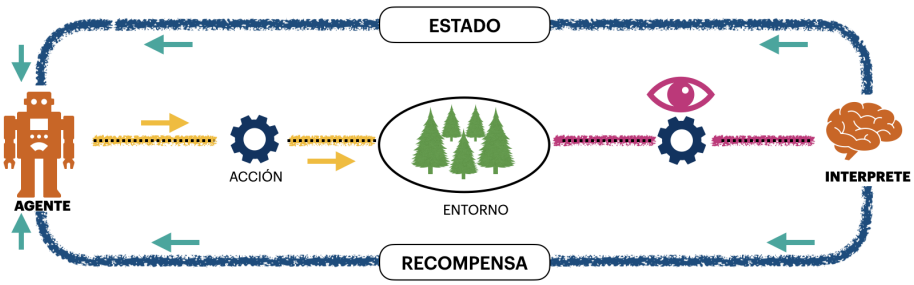

Figura 1: Aprendizaje Reforzado

Para la aplicación de este caso de estudio utilizaremos el ejemplo de la Figura 2, el cual aborda el problema de un niño con discapacidades el cual adquiere un software con el fin de obtener una alimentación diferente los días viernes, dicha alimentación se encuentra en el restaurante de su padre, una persona bastante ocupada; el niño oprime un botón y el software selecciona una de las opciones de menú para el y genera la orden de pedido, mientras su Padre se ocupa del negocio.

El algoritmo contiene una función que permite identificar que acción es la acción óptima, a esta función la llamamos función de acción-valor.

La función acción-valor óptima se define como la recompensa esperada al realizar una acción a_i para cada posible acción $i \in \{1, \ldots, k\}$. El objetivo de nuestro agente es que se maximice la expectativa de recompensa:

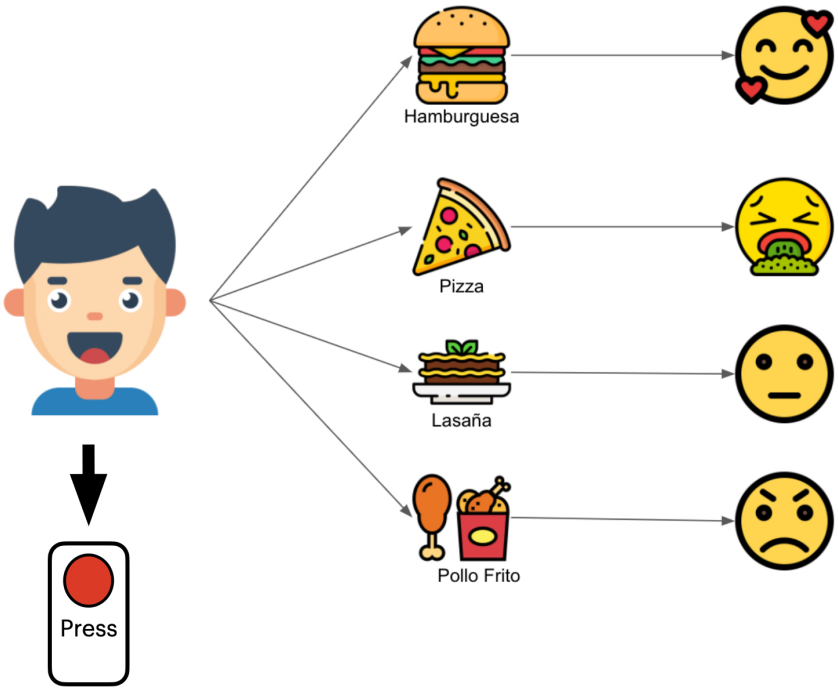

Figura 2: Ejercicio Selección de Alimentos

2. Metodología

Este trabajo se basa en la metodología CRISP-DM; esta ha sido aplicada en diferentes áreas de la industria y trabajos metodológicos e investigativos. Esta metodología promueve la solución de problemas y la posibilidad del relacionamiento con problemas del día a día. Esta metodologia esta compuesta de 6 permiten abordar los problemas de una manera incremental y secuencial, con el fin de hallar la posible solución. La figura 3 presenta el diagrama de dicha metodología. Las siguientes 5 secciones se dividirán siguiendo los 5 primeros pasos de CRISP-DM. El último paso no será tenido en cuenta ya que dentro de este trabajo no se realiza ningún despliegue o puesta en producción de los modelos entrenados.

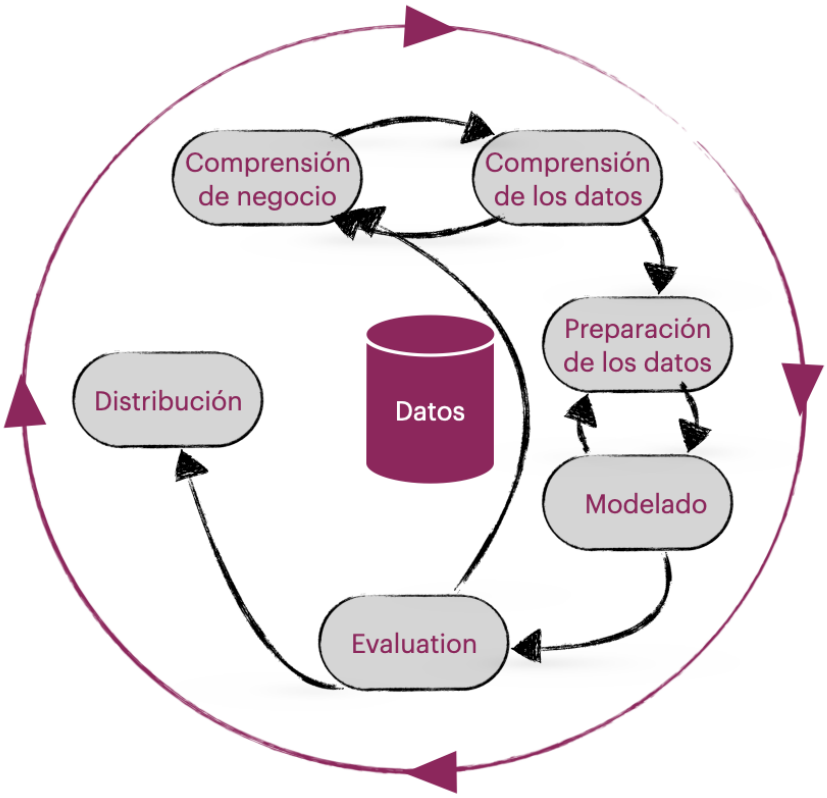

Figura 3: Metodología Crisp-DM

3. Comprensión del Negocio

En esta sección explicaremos mas a profundidad el problema que se resuelve por medio del aprendizaje reforzado.

El contexto es de un hombre que tiene un restaurante de comidas rápidas, el hombre en su trabajo diario de atender el negocio, velar por la satisfacción de sus clientes y manejar la contabilidad, no tiene tiempo de nada mas. Sin embargo un día, le prometió a su hijo discapacitado que todos los viernes podría ir al restaurante y aleatoriamente le daría un plato sorpresa del menú. Debido a su gran ocupación y a el deseo de cumplir dicha promesa a su hijo, le pidió a sus 2 hermanos un desarrollador de software y un Científico de datos que diseñaran una aplicación que tuviera un algoritmo que fuera aprendiendo de los gustos de su hijo y que al presionar un botón, generara no solo la selección del plato con los gustos de su hijo sino también, la orden de dicho plato.

A continuación se plantea el siguiente objetivo: al presionar un botón en la aplicación se ejecutará un algoritmo el cual va a seleccionar una comida aleatoria y luego la aplicación calificará con un emoji para proseguir con la generación de la orden. Después de un tiempo, se comenzará a preferir un tipo de comida y, de cierto momento en

adelante, solo come ese tipo de comida o continúa con la selección aleatoria y explorando sus opciones. Si solo come un tipo de comida, no podrá recopilar datos sobre las otras opciones gastronómicas.

4. Comprensión de los Datos

En esta sección se realizara una breve descripción de los datos y su estado. Se inicia cargando un data se que contiene 4 columnas con cada una de las opciones de la carta y 5786 registros entre los cuales se encuentran registros en "NaN", valores "0"ó "1". Estos valores la recompensa que recibe de 1 si le gusta la comida que probó en ese momento en particular y 0 en caso contrario.

	pollo_frito	pizza	hamburguesa	lagsana
0	0.0	1.0	NaN	NaN
1	0.0	1.0	1.0	1.0
2	0.0	1.0	NaN	NaN
3	NaN	1.0	0.0	NaN
4	NaN	1.0	0.0	NaN
...	...	...	...	...
5781	NaN	NaN	1.0	0.0
5782	NaN	NaN	1.0	0.0
5783	NaN	0.0	NaN	NaN
5784	NaN	0.0	NaN	NaN
5785	0.0	0.0	1.0	0.0

5786 rows × 4 columns

Figura 4: Carga de data inicial

Como se ve en la figura anterior, la data tiene campos en NaN, por lo cual se realiza la estandarización de los datos para que dichos campos queden con un valor cero 0 reemplazando así el NaN.

	pollo_frito	pizza	hamburguesa	lagsana
0	0.0	1.0	0.0	0.0
1	0.0	1.0	1.0	1.0
2	0.0	1.0	0.0	0.0
3	0.0	1.0	0.0	0.0
4	0.0	1.0	0.0	0.0
...	...	...	...	...
5781	0.0	0.0	1.0	1.0
5782	0.0	0.0	1.0	1.0
5783	0.0	0.0	0.0	0.0
5784	0.0	0.0	0.0	0.0
5785	0.0	0.0	1.0	1.0

5786 rows × 4 columns

Figura 5: Estandarización de los Datos

Luego de esto se genera la visualización de la media de los datos.

```
pollo_frito    0.248877
pizza          0.161770
hamburguesa    0.178362
lagsana        0.178362
dtype: float64
```

Figura 6: Media de los Datos

5. Preparación de los Datos

Veremos que medida que el agente tome más acciones, nuestras estimaciones se acercarán cada vez más a los valores reales de Q_* que se detallan en la siguiente ecuación. Generalizando el cálculo anterior, podemos realizar un seguimiento del promedio de muestra de nuestras recompensas de la siguiente manera:

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i$$

Sacamos la recompensa actual de la suma:

$$= \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i)$$

Escribimos el próximo valor estimado en términos del valor estimado anterior; para esto, multiplicamos y dividimos por $n-1$:

$$= \frac{1}{n}(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i)$$

El último término es nuestra estimación de valor actual:

$$= \frac{1}{n}(R_n + (n-1)Q_n)$$

ahora, terminamos actualizando la función de valor de acción; esto se conoce como la **regla de actualización incremental**:

$$Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n)$$

Finalmente lo que obtenemos para las nuevas estimaciones es:

$$\text{Nueva estimación} = \text{Vieja estimación} + \text{Tamaño del paso}[\text{Etiqueta - Vieja estimación}]$$

5.1. Selección de la acción

En realidad, el agente no elegiría una acción al azar. Más bien, elegiría lo que cree que es la mejor comida de acuerdo con su experiencia hasta ese momento. Este método de elección de acciones se conoce como **codicioso** con enfoque de selección de acciones. Si el agente solo elige la acción codiciosa, esto se consideraría una **explotación** acercada.

Sin embargo, también podría elegir acciones no codiciosas con cierta probabilidad ϵ para obtener más información sobre los otros valores de acción esto se conoce como **exploración**. Una combinación de ambos es formalmente llamada un **ϵ-metodo codicioso**.

Adicionalmente también existe un problema llamado **k-Problema de bandidos malos**.

Primero debemos entender que en este problema tenemos un agente que elige entre k-diferentes acciones, y recibe una recompensa de acuerdo a la acción que elige. En el problema se cumplen las siguientes condiciones:

- el **agente es el niño**
- las **k-diferentes acciones son los diferentes tipos de comida que podría pedir**
- Cada elección de acción o tipo de comida, generará una recompensa diferente

5.2. Agente codicioso

Primero implementamos un agente que adopte un enfoque codicioso. Inicialmente vamos a implementar la función argmax, que toma una lista de valores de acción y devuelve una acción con el valor más alto.

Implementamos nuestra función en lugar de usar la función argmax que usa 'numpy'; debido a que la función argmax de 'numpy' devuelve la primera instancia del valor más alto. No queremos que lo anterior suceda; ya que sesga al agente a elegir una acción específica en el caso de empates. En su lugar, queremos romper los lazos entre los valores más altos al azar:

6. Modelado y Evaluación

Aquí vamos a crear un GreedyAgent o Agente codicioso y se implamentara el método ´agent-step()´. Este método se llama cada vez que el agente da un paso. El método tiene que devolver la acción seleccionada por el agente. Este método también garantiza que las estimaciones del agente se actualicen en función de las señales que recibe del entorno:

La función agent-step toma una recompensa y observación y devuelve la acción que el agente elige en ese paso de tiempo.

Entrada o recompensa: la recompensa que el agente obtiene del entorno después de la última acción.

Observación: El estado que observan los agentes.

Devuelve: La acción elegida por el agente en el paso de tiempo actual

A continuación visualizamos los resultados para lo mencionado anteriormente con 250 ejecuciones y 1200 pasos:

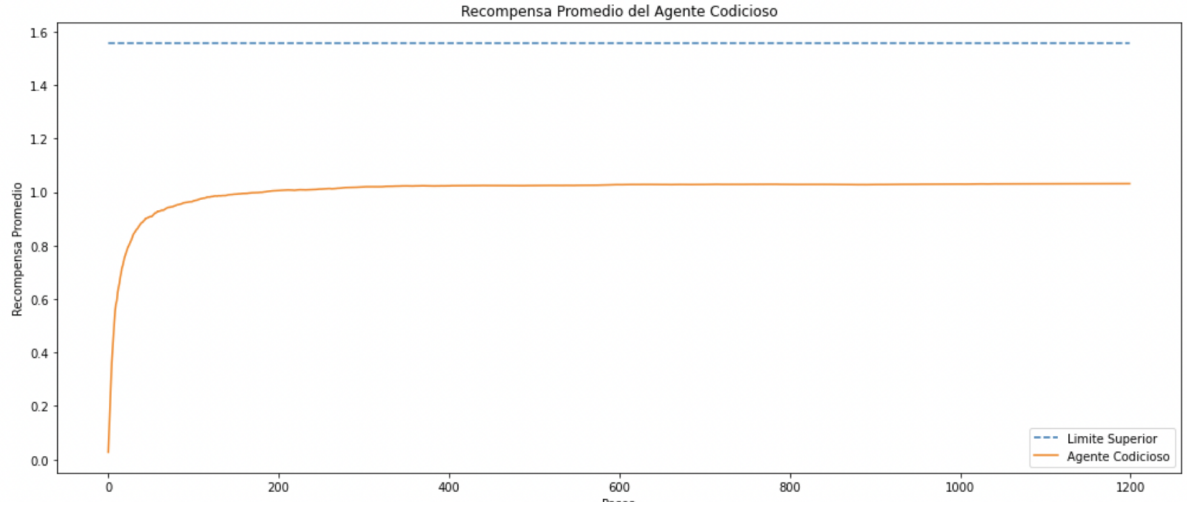

Figura 7: Ejecución 1 - Agente Codicioso

Ahora se adiciona un parámetro ϵ de 0.1 y se compara con el resultado anterior

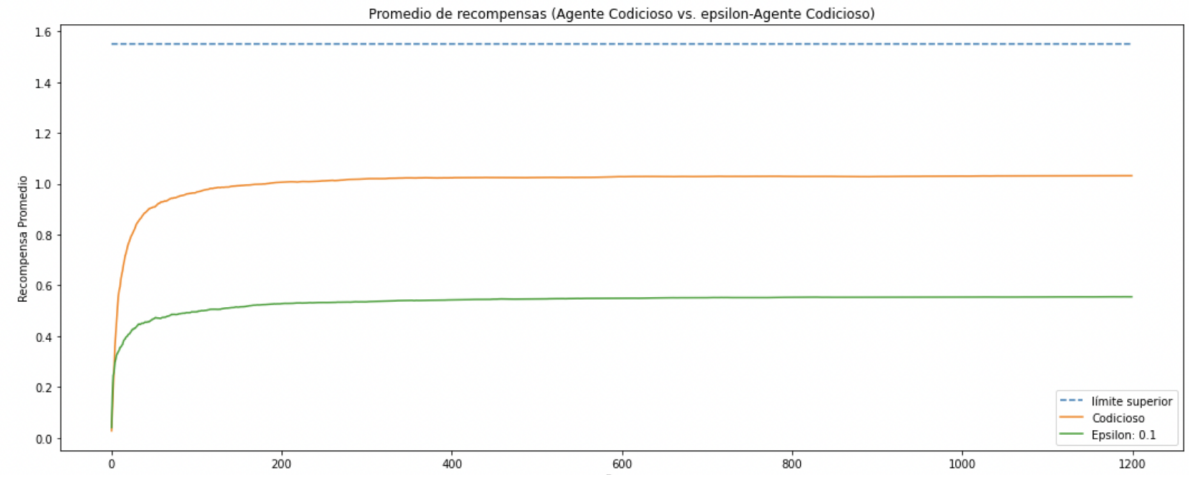

Figura 8: Ejecución 2 - Agente Codicioso con Parámetro ϵ=0.1

Ahora promediamos los resultados durante 250 ejecuciones. A continuación se muestran los resultados de cuatro ejecuciones individuales realizadas por el mismo agente.

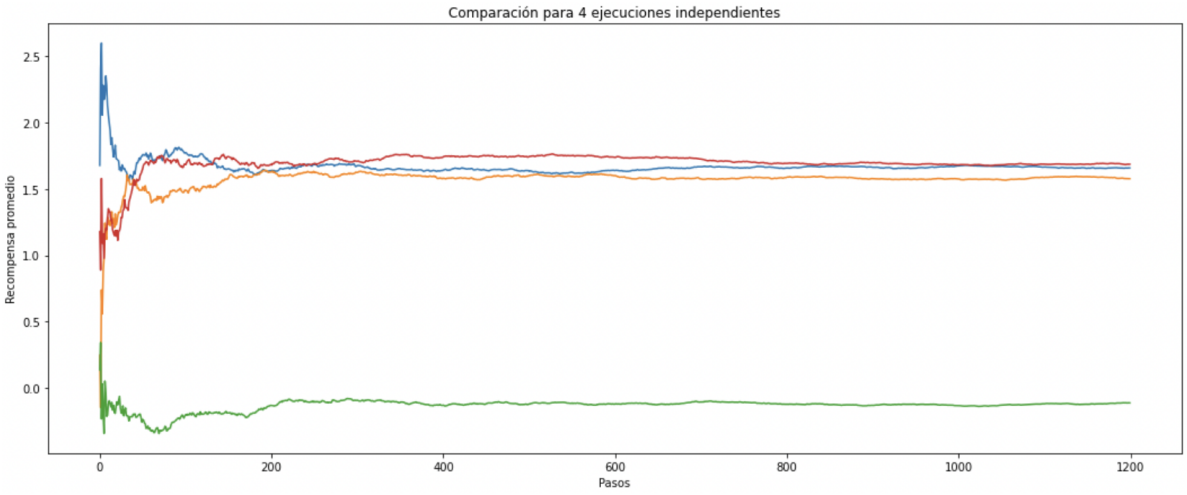

Figura 9: Promedio de Resultados para 4 Ejecuciones Independientes

También se probó cambiando valores diferentes para ϵ con el fin de obtener el desempeño. De esto se obtuvo el siguiente resultado:

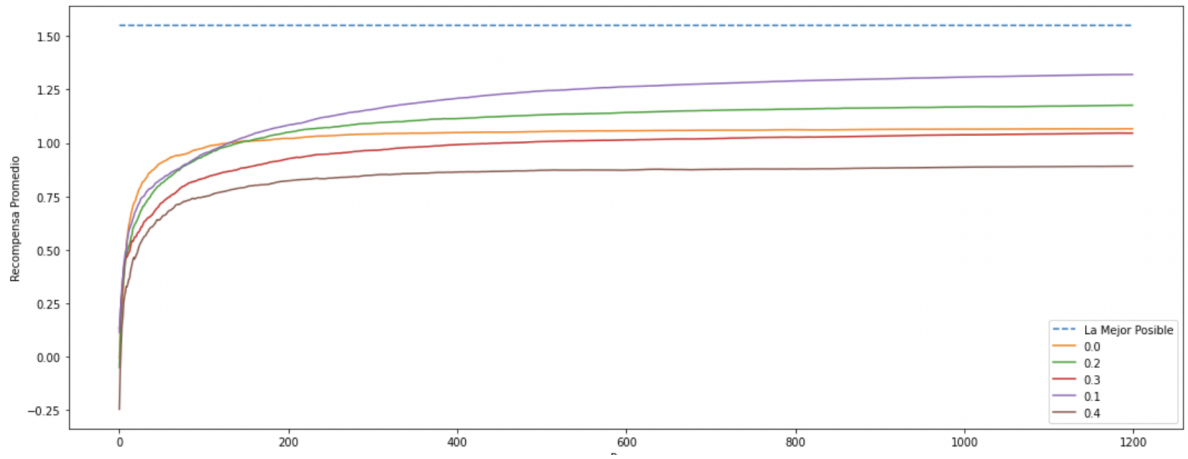

Figura 10: Resultado con Valores Diferentes para ϵ

Se puede observar que los mejores resultados son aquellos que se acercan a el limite superior, y cuyos valores para ϵ son 0.1 y 0.2. El numero de pasos nuevamente fueron 1200 y las ejecuciones realizadas fueron 250.

7. Conclusiones

Observamos como en las cuatro ejecuciones son diferentes debido a la aleatoriedad en el entorno y el agente. Los resultados variarán dependiendo de la acción con la que el agente comience aleatoriamente, o cuando elija explorar aleatoriamente. Incluso si el agente elige la misma acción, la recompensa se muestrea aleatoriamente a partir de una distribución gaussiana. Por lo tanto, hay muchas fuentes de variabilidad sobre las que tiene sentido promediar.

Adicionalmente, cabe resaltar que estos algoritmos computacionalmente son bastante robustos y cuando se trata de problemas complejos requieren recursos adicionales que soporten dicha ejecución.

Referencias

[1] Applications of reinforcement learning for building energy efficiency control: A review, 6 2022.

[2] Gang Chen, Yuwang Lu, Xin Yang, and Huosheng Hu. Reinforcement learning control for the swimming motions of a beaver-like, single-legged robot based on biological inspiration. *Robotics and Autonomous Systems*, page 104116, 4 2022.

[3] Maryam Karimi-Mamaghan, Mehrdad Mohammadi, Bastien Pasdeloup, and Patrick Meyer. Learning to select operators in meta-heuristics: An integration of q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 4 2022.

[4] Jueyou Li, Xiaomei Zhu, Zhiyou Wu, and Tingwen Huang. Online distributed dual averaging algorithm for multi-agent bandit optimization over time-varying general directed networks. *Information Sciences*, 581:678–693, 12 2021.