Universidad del Valle de Guatemala

Departamento de computación

Aprendizaje por refuerzo

Ing. Luis Alberto Suriano Saravia

Proyecto Final

Aprendizaje por refuerzo en juego de cartas y comparación del rendimiento de algoritmos

Marlon Hernández 15177

I. DESCRIPCIÓN DEL PROBLEMA

El juego de "Cho Dai Di" o póker chino, es un juego de cartas por turnos en el que los jugadores deben deshacerse de sus cartas lo más rápido posible formando combinaciones válidas. Los jugadores pueden elegir jugar una combinación de cartas que sea más alta que la combinación actual en la mesa o pasar. El objetivo final es quedarse sin cartas. La complejidad del juego radica en la toma de decisiones estratégicas, como cuándo jugar, qué combinaciones elegir y cuándo pasar.

II. ANÁLISIS

Para encontrar las mejores decisiones estratégicas el aprendizaje por refuerzo puede ser útil para desarrollar agentes que aprendan a jugar eficazmente. Dado que cada jugador tiene un conjunto único de cartas y hay una jerarquía de combinaciones (individuales, pares, tríos, escaleras, etc.), es importante desarrollar un entorno donde los agentes puedan aprender a identificar oportunidades y tomar decisiones basadas en recompensas.

Las decisiones estratégicas también dependen del estado actual del juego, el número de cartas que tienen los oponentes y las combinaciones que ya han sido jugadas, lo que lo convierte en un problema interesante para el aprendizaje por refuerzo.

Así mismo se puede hacer una comparación entre distintos algoritmos para ver su eficiencia y cómo aprenden las estrategias del juego y de qué manera se pueden utilizar para aprender otros
juegos. El juego "Cho Dai Di" presenta varios desafíos desde el punto de vista de la inteligencia artificial y el aprendizaje por refuerzo:

- **Multijugador y Competitividad**: Los agentes deben interactuar en un entorno donde las acciones de unos afectan directamente a los demás.

- **Espacio de Estados y Acciones Grande:** El número de posibles estados y combinaciones de acciones es enorme debido a la variedad de cartas y combinaciones posibles.

- **Reglas Complejas**: Las reglas del juego incluyen jerarquías de combinaciones, penalizaciones por movimientos inválidos y estrategias como pasar.

- **Balance entre Exploración y Explotación**: Los agentes deben aprender a balancear entre probar nuevas estrategias y aprovechar el conocimiento adquirido.

III. PROPUESTA DE SOLUCIÓN

Para abordar el problema, se propuso crear un entorno de simulación de "Cho Dai Di" utilizando la biblioteca Gym de OpenAI y desarrollar varios agentes de inteligencia artificial, cada uno con diferentes enfoques de aprendizaje por refuerzo:

- **Q-Learning Agent:** Un agente basado en tablas Q que aprende de la experiencia acumulada.
- **DQN Agent**: Un agente basado en Deep Q-Learning que utiliza redes neuronales para generalizar estados.
- **Policy Gradient Agent:** Un agente que aprende una política directamente mediante gradientes.
- **Sarsa Agent:** Un agente que utiliza la técnica de aprendizaje SARSA (State-Action-Reward-State-Action).

Estos agentes fueron entrenados y evaluados en un entorno donde se implementaron las reglas de juego, con combinaciones válidas y un sistema de recompensas que fomentaba el juego estratégico.

Cada agente deberá aprender a jugar el juego y competir contra los demás, para analizar su desempeño y estrategias aprendidas. Se utilizarán métricas como recompensas acumuladas, número de victorias, movimientos inválidos y tipos de combinaciones jugadas para evaluar y comparar a los agentes.

IV. DESCRIPCIÓN DE LA SOLUCIÓN

Se hizo el desarrollo de una clase ChoDaiDiEnv que hereda de gym.Env, para simular el entorno del juego. Las características principales incluyen:

- **Estado del Juego**: Representado por una matriz que indica las cartas que tiene cada jugador.

- **Acciones:** Los jugadores pueden jugar combinaciones válidas de cartas o pasar su turno.

- **Recompensas y Penalizaciones:** Se establecieron recompensas por jugar combinaciones válidas y ganar el juego, y penalizaciones por movimientos inválidos o pasar innecesariamente.

- **Reglas del Juego:** Se implementaron las reglas para validar combinaciones, determinar si una combinación es mayor que otra y gestionar los turnos.

Luego se desarrollaron cuatro agentes basados en diferentes algoritmos:

- **Q-Learning**

  - Utiliza una tabla Q para almacenar los valores Q de los estados y acciones. Se balancea entre exploración y explotación utilizando una política ε-greedy. Se actualizan los valores Q basándose en la recompensa obtenida y el valor máximo futuro estimado.

- **Deep Q-Network (DQN)**

  - Emplea una red neuronal para aproximar la función Q, permitiendo manejar espacios de estado y acción continuos o grandes. Se utiliza una red neuronal con capas densas y función de activación ReLU. Se utiliza experiencia previa para aprender y mejorar sus estimaciones Q. Implementa un mecanismo para evitar desbordamientos de índice en el espacio de acciones.

- **Policy Gradient**

  - Aprende directamente la política que maximiza la recompensa esperada utilizando gradientes de política. Es una red neuronal que produce una distribución de probabilidad sobre las acciones.

- **SARSA**

  - Actualiza los valores Q basándose en la acción que se tomó en el siguiente estado. También utiliza una política ε-greedy y aprende valores Q considerando las transiciones de estado-acción-estado-acción.

Se fueron mejorando las recompensas y penalizaciones para incentivar comportamientos deseados y desalentar acciones inválidas. También se ajustaron los hiperparámetros como la tasa de aprendizaje, factor de descuento y el número de episodios para mejorar la convergencia y el desempeño de los agentes.

V. HERRAMIENTAS APLICADAS

Se implementaron distintos algoritmos de aprendizaje por refuerzo:

- Q-Learning y SARSA: Métodos de aprendizaje por refuerzo basados en tablas Q.
- Deep Q-Network (DQN): Utiliza redes neuronales para aproximar funciones Q en espacios grandes.
- Policy Gradient: Aprende directamente una política que maximiza la recompensa esperada.

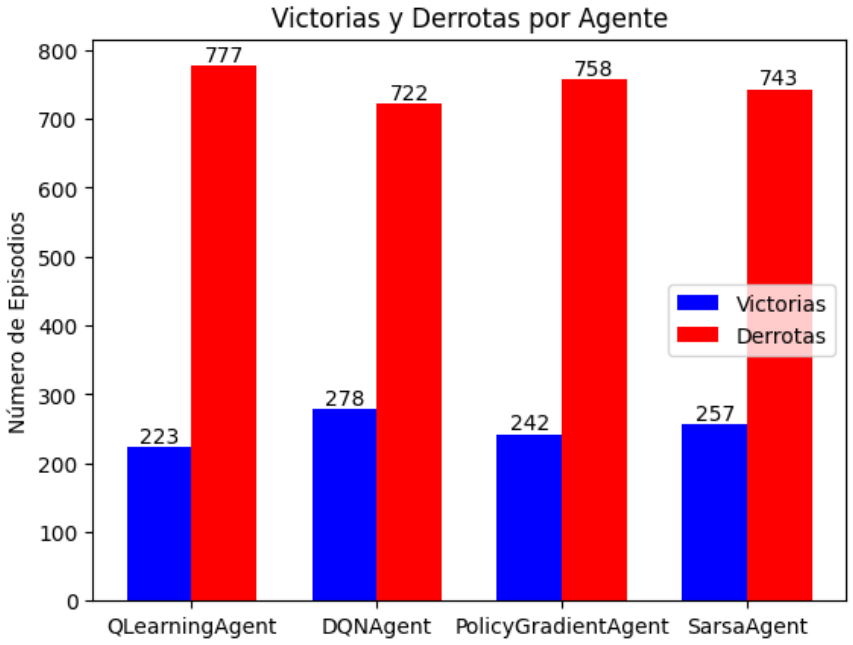
Se utilizaron los

- Librería Gym de OpenAI: Se utilizó Gym para crear el entorno personalizado del juego, aprovechando su estructura para gestionar espacios de estados y acciones, y facilitar la interacción entre los agentes y el entorno.
- PyTorch: Para los agentes que requieren redes neuronales (DQN y Policy Gradient), se empleó PyTorch:
- Redes Neuronales: Se definieron modelos secuenciales con capas lineales y funciones de activación ReLU y Softmax.

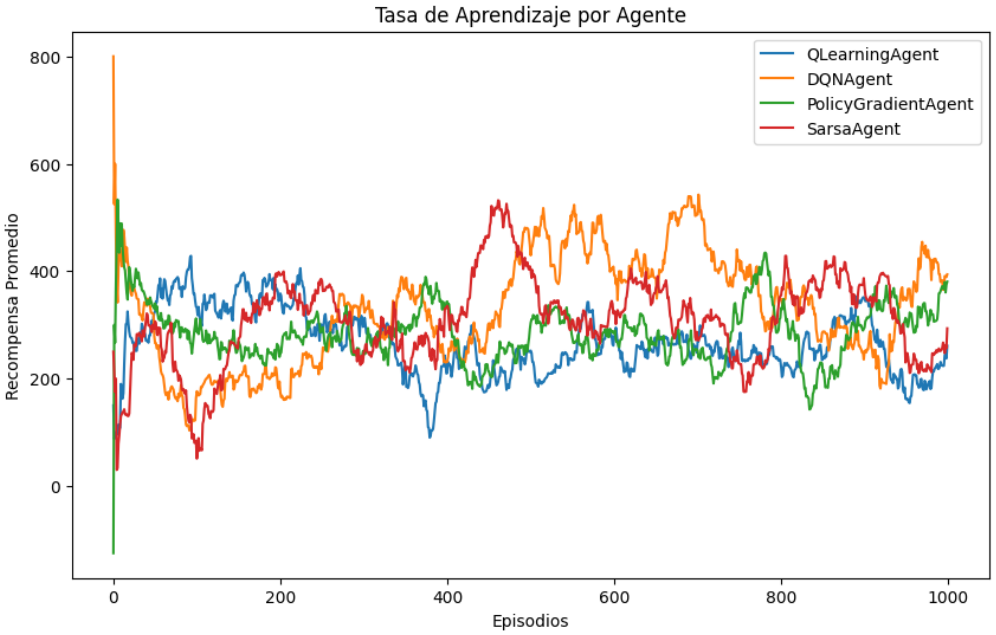
VI. RESULTADOS

Los resultados obtenidos por los diferentes algoritmos de aprendizaje por refuerzo, brindan información valiosa sobre su desempeño en relación de conocer nuevas estrategias y victorias obtenidas. Se realizaron simulaciones durante 1000 episodios, recopilando las siguientes métricas.

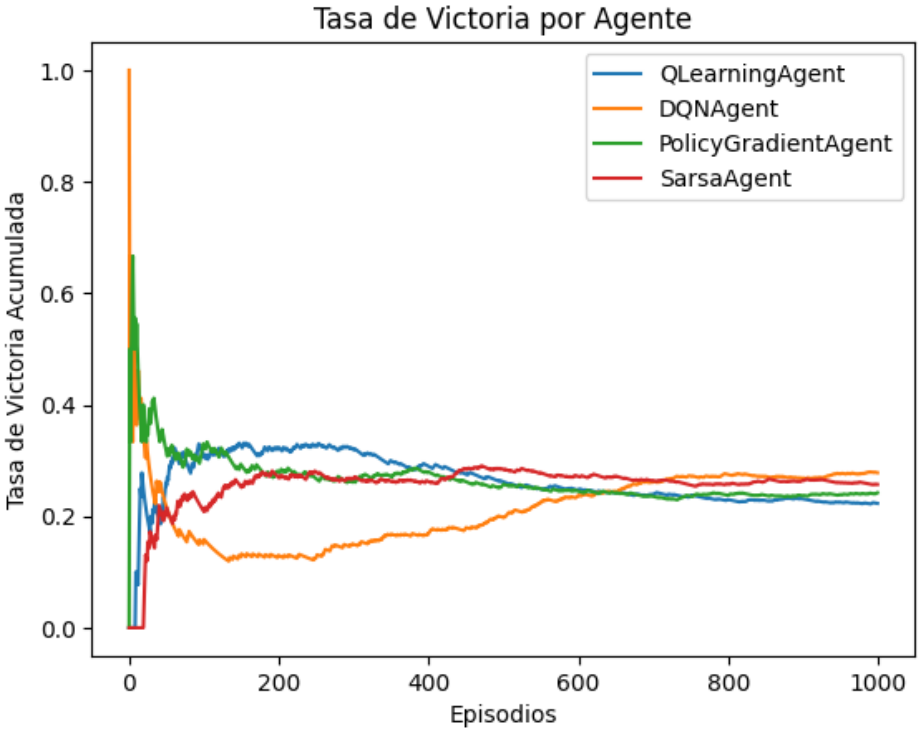
Iniciando con el objetivo de ganar la partida, se obtuvo en la simulación que DQN fue el agente con más victorias, seguido de SARSA, Policy Gradient, y finalmente QLearning. Esto muestra que los agentes obtuvieron un buen aprendizaje del juego y fueron oponentes buenos durante la simulación.



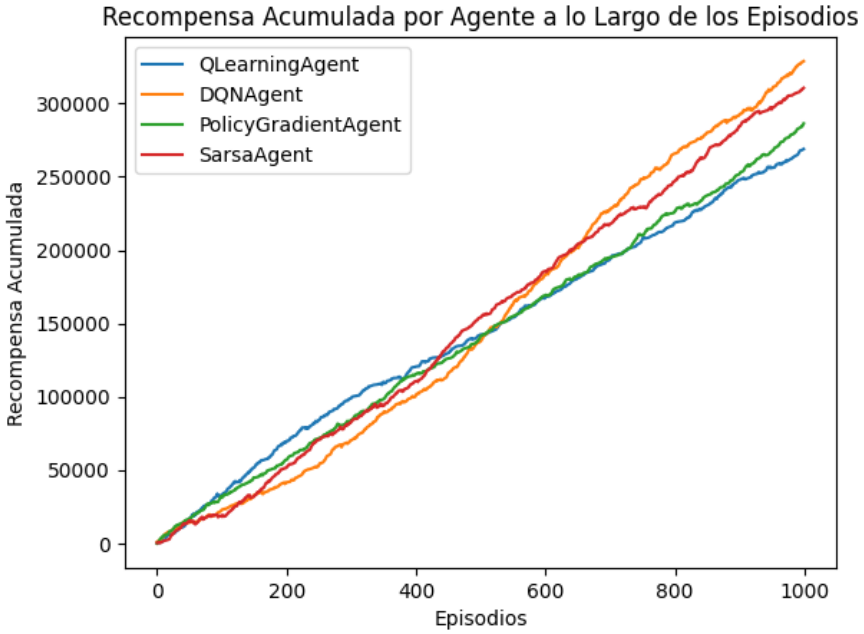
En general, se observa una tasa de aprendizaje con un promedio estable a lo largo de la simulación, donde DQN y Policy Gradient empiezan con un promedio relativamente alto, lo que indica que convergen y aprenden estrategías ganadoras más rápido. Sin embargo, es importante considerar otras métricas para tener una visión más completa del rendimiento de los modelos.



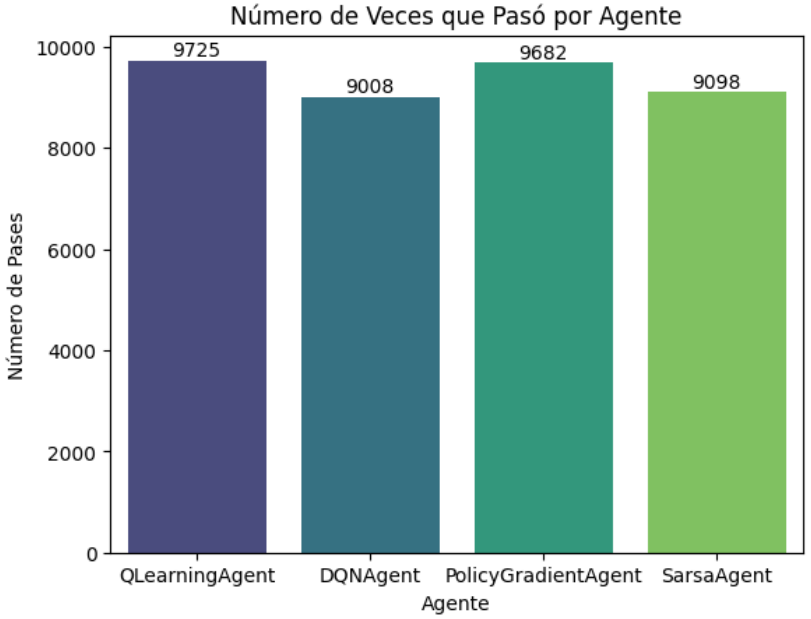
Un objetivo es encontrar mejores estrategias de juego para ganar, por lo que es importante conocer como fue el desempeño de victorias en la simulación. Se observa que conforme avanzan los episodios, empieza a ser reñida las partidas y no hay un ganador declarado, al inicio DQN y Policy Gradient van a la cabeza, pero son rebasados por QLearning y SARSA, lo que muestra que todos los agentes lograron aprender buenas estrategias para poder jugar y tener estrategias ganadoras.



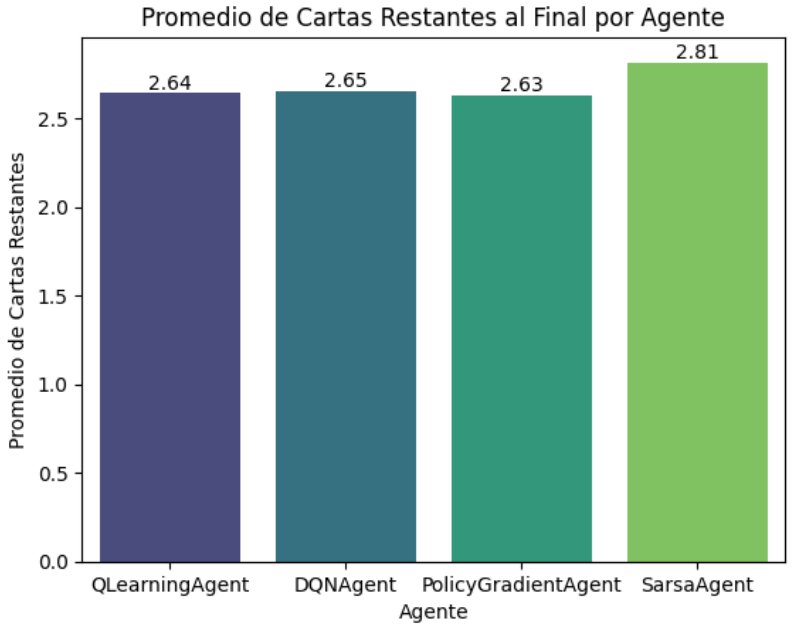
En la recompensa acumulada vemos el rendimiento de los agentes conforme a realizar jugadas válidas y obtener mejor recompensa para encontrar las acciones que tienen más retorno. Se observa que DQN a lo largo del tiempo va rebasando a los otros agentes en recompensas y Policy Gradients con QLearning.



Una de las acciones que se castiga por realizar es el pasar el turno, ya que el objetivo del juego es quedarse sin cartas, pasar el turno no es eficiente para lograr ganar. En la siguiente gráfica se observa la cantidad de veces que pasó cada agente en la simulación, donde se observa que DQN fue el que menos veces pasó al encontrar que esa acción era contraproducente.

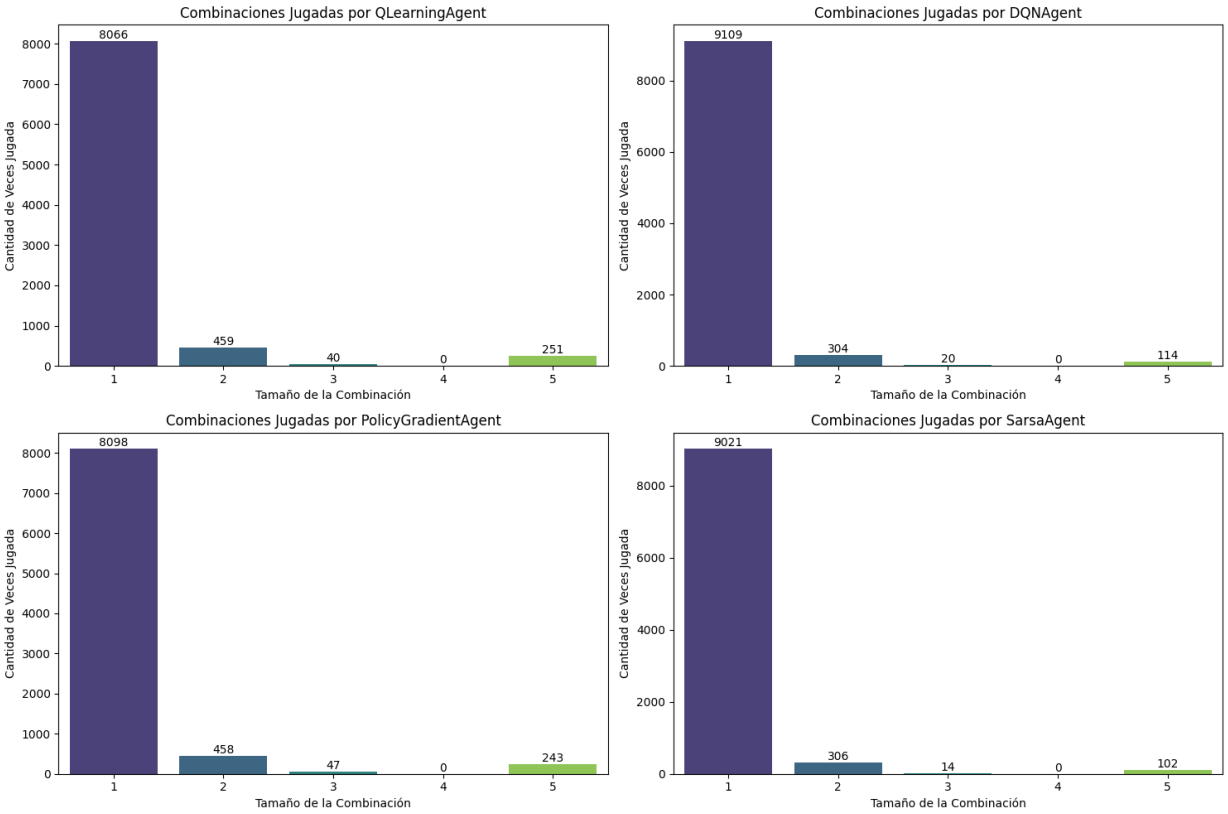


Por otro lado, quedarse con muchas cartas es una consecuencia de no ser eficiente en el juego, por lo que se encontró que los agentes lograron utilizar bien las estrategias para poder deshacerse de sus cartas lo más pronto y no quedarse con muchas en la mano. Se observa que en promedio todos los agentes mantuvieron un promedio entre 2 y 3 cartas al final de las partidas, lo que indica que siempre lograban estar cerca de ganar la partida.



Finalmente vemos los tipos de jugada que hacían los agentes, se obtiene que la mayoría de veces, los agentes realizaban jugadas de 1 carta, seguido de 2 y luego de 5, esto muestra que

todos los agentes lograron hacer buen uso de la estrategia de usar combinaciones de bastantes, así deshacerse de la mayor parte de su mano.



VII. CONCLUSIÓN

- Los agentes fueron capaces de aprender estrategias para jugar el juego, reflejado en el aumento de las recompensas acumuladas y victorias a lo largo de los episodios.

- El mejor agente fue DQN que aprendió y convergió rápidamente para encontrar las mejores estrategias, lo que le dio ventaja al inicio y mantuvo su promedio de recompensas a lo largo de la simulación.

- Ajustar las recompensas y penalizaciones fue importante para mejorar el aprendizaje de los agentes y obtener comportamientos deseados. Así como los otros hiperparámetros que fueron ajustándose conforme se realizaron las simulaciones.

- Los resultados obtenidos muestran que de utilizar técnicas de aprendizaje por refuerzo encontrar mejores estrategias en juegos de cartas es viable.

VIII. BIBLIOGRAFÍA

- Karunakaran, D. (2020, September 17). **Q-learning: A value-based reinforcement learning algorithm.** Medium. https://medium.com/intro-to-artificial-intelligence/q-learning-a-value-based-reinforcement-learning-algorithm-272706d835cf

- Dhumne, S. (2023, June 30). **Deep Q-Network (DQN)** - Shruti Dhumne – Medium Medium. https://medium.com/@shruti.dhumne/deep-q-network-dqn-90e1a8799871

- Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual Multi-Agent Policy Gradients. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1). https://doi.org/10.1609/aaai.v32i1.11794

- Tostaeva, G. (2021, December 14). Learning with Deep SARSA & OpenAI Gym - The Startup - Medium. Medium. https://medium.com/swlh/learning-with-deep-sarsa-openai-gym-c9a470d027a

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