Futbolito para el aprendizaje de Agente-Agente con Aprendizaje por Refuerzo

Inteligencia artificial avanzada para la ciencia de datos II (Gpo. 501)

Tecnológico de Monterrey

Elaborado por:
Alejandro Fernández del Valle Herrera
A01024998

Profesores:
Lizbeth P.
Eduardo B.
Octavio N.
Gerardo Jesús C.
Esteban C.

30 de noviembre del 2024

I. Contenido

II. Introducción

El propósito de este proyecto es el aprendizaje por refuerzo, donde se explora la solución aplicada a un futbolito (foosball). A través de este proyecto se busca aplicar técnicas avanzadas de Aprendizaje por Máquina, donde se descubra diversos algoritmos y soluciones. Este proyecto es principalmente de investigación y no es a nivel del estado de arte, si no, explora conceptos avanzados para el inicio del aprendizaje.

a. Criterios

Durante la realización del presente proyecto, los criterios de excito son los que siguen:

A) Se utiliza una técnica de Aprendizaje por Refuerzo en PyTorch.

B) Se investiga la viabilidad de redes convolucionales para el entendimiento de datos.

C) Los agentes son capaces de meter gol.

b. Tecnologías usadas

Se han utilizado diversas tecnologías para completar el proyecto. Cada una tiene un propósito diferente, y cuenta con diversos usos. Las tecnologías que se van a usar son:

i. Godot

Godot se utiliza como el ambiente en donde se va a correr la simulación. La justificación de usar godot a diferencia de otros fameworks, son:

- Godot cuenta con un sistema de física.
- Es gratuito, por lo que es mas fácil de correr para otras personas.
- Se puede realizar ejercicios de visión en un futuro.

ii. PyTorch

PyTorch es usado para el modelo de IA. El uso de PyTorch permite flexibilidad para las pruebas de ML.

iii. FastAPI

Se utiliza para la comunicación entre Godot y PyTorch. Es lo que se tuvo que implementar para el funcionamiento correcto.

III. Planteamiento

La realización de un proyecto de este tipo debe de tomar muchas cosas en cuenta, por lo que se va a iniciar por analizar lo que es el futbolito.

a. Futbolito

El futbolito consiste en 2 jugadores, donde cada uno tiene control de cada lado y el propósito final es meter una bola a la meta del oponente contrario. Cada jugador tiene control de 4 varas, donde al manipularlas, se puede pegar la bola que inicia en el centro.

El Futbolito se puede representar de la siguiente manera:

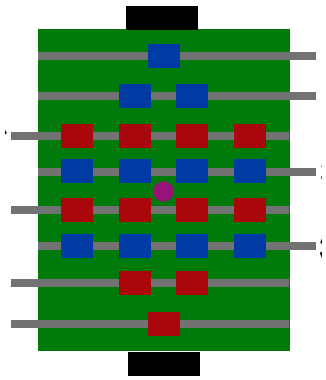


Fig. 1: Representación gráfica del futbolito

i. Movimientos

Dentro del futbolito, la única forma de interactuar con el juego es por medio de los palos (representados en gris en la *fig. 1*) donde se puede tomar 2 acciones por cada palo:

1. Mover hacia adelante o atrás
2. Girar para poder realizar el movimiento

Cada jugador cuenta con 4 varas, donde mezclando los movimientos entre ellos, puede realizar jugadas para poder defender o atacar al oponente.

ii. Posicionamientos

En el futbolito, se pueden observar 2 posiciones defensivas, y 2 posiciones ofensivas. La que se encuentra mas cerca de la portería del jugador, se conoce como el portero, los que siguen son las defensas, el que sigue es el central, y al final es la ofensa.

Cada jugador tiene su rol y un rango de movimiento diferente. Esto es dado que las defensas son mayor numero, por lo que son mas pesadas y difíciles de mover. A su vez, utilizan mas espacio en la cancha, realizando que sea complicado moverlo. El portero cuenta con una mayor movilidad, pero se limita su rango de movimiento a la portería, y las defensas están en un punto medio.

b. Red neuronal

Una vez conociendo los posibles estados del futbolito, se puede crear una red neuronal. Lo que puede ver un jugador en todo tiempo, es la posición de todo el juego, por lo que pueden simplificarse los datos a lo que sigue:

i. Posición del jugador

El jugador cuenta con 4 varas, donde cada vara tiene 2 variables. Es decir, cuenta con una matriz de 4x2.

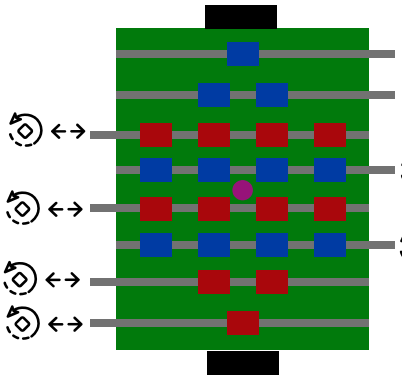


Fig 2. Estado del jugador

La posición se puede representar en un valor de -1 a 1, ya que las redes neuronales son mas eficientes con valores dentro del rango de -1 a 1 o 0 a 1. 0 representa el centro, mientras que -1 la izquierda, y 1 la derecha. Para la rotación, se puede tomar la misma metodología, donde una rotación hacia adelante (de o a 180 grados) puede ser transformada de 0 a 1, y hacia atrás (de 0 a -180 grados) puede ser transformada de 0 a -1, 0 siendo la posición donde se puede pegar la bola.

ii. Posición del enemigo

El enemigo se puede clasificar de la misma forma que el jugador. Cuenta con 4 varas donde cada una puede contar con una rotación y posición. Se puede entonces crear una traducción de valores de -1 a 1, con la vista inversa:

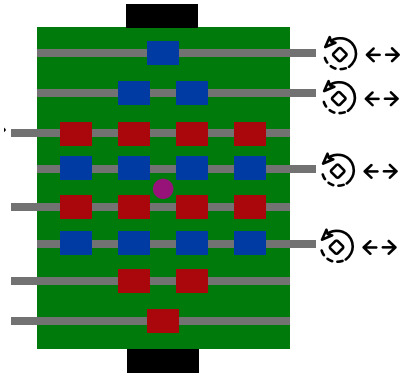


Fig 3. Estado del enemigo

iii. Posición de la pelota

La pelota se puede visualizar como posición en ‘x’ y ‘z’, donde ‘x’ es la posición vertical, y ‘z’ la posición horizontal. La posición se puede representar como -1 siendo tu portería/la izquierda, y 1 siendo la portería del enemigo/derecha, centro siempre siendo 0.

iv. Espacio de acción.

Se puede definir el espacio de acción como las acciones que puede tomar un agente. Como se había mencionado antes, cada jugador puede rotar y deslizar sus varas, creando un espacio de 4x2 acciones. Estas son relativas a fuerzas, ya que en vez de definir la velocidad, se define la fuerza que se va a aplicar en la siguiente iteración de la simulación. Cada una se puede definir de -1 a 1, donde -1 significa rotar hacia su portería, y 1 hacia el lado de la portería del enemigo.

El deslizamiento es lo mismo, sin embargo -1 es la izquierda, y 1 la derecha. Esto para seguir el mismo estándar que se usa en los gráficos y procesamientos de este tipo.

v. Red Neuronal

Como se menciono el la introducción, uno de los experimentos es probar el uso de Redes Neuronales Convolucionales (CNN) para poder obtener información sobre como se puede relación la posición con relación a la rotación.

Para cumplir con este requisito, se puede acomodar la información de la siguiente forma:

$$I = \begin{bmatrix} A & \cdots & B & \cdots & \begin{bmatrix} C \\ \vdots \\ C \end{bmatrix} \end{bmatrix}$$

- A = 4x2 el estado del jugador
- B = 4x2 el estado del enemigo
- C = 2x2 la posición de la pelota y velocidad en ‘x’ y ‘z’: $\begin{bmatrix} x_p & z_p \\ x_v & z_v \end{bmatrix}$
- I = 4x6 la posición de la pelota y velocidad en ‘x’ y ‘z’

Una vez obtenida la matriz de entrada, se puede crear una CNN de 2 dimensiones, donde se puede simplificar cada área de la matriz con un kernel de tamaño 2x2 para obtener una lectura de la siguiente manera:

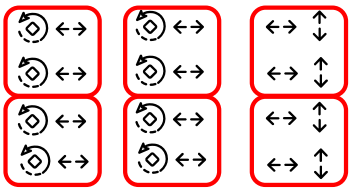


Fig 4. Representación del kernel

El kernel se puede representar de la siguiente forma:

$$\begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

Donde ‘a’ representa la defensa del jugador, ‘b’ su ofensa, ‘c’ la defensa del enemigo, ‘d’ la ofensa del enemigo, ‘e’ y ‘f’ la información de la posición del jugador.

Una vez creada la CNN, se puede aplanar para poder tener una capa completamente conectada (FC).

Esta es la que puede clasificar los filtros de la CNN y convertirlo en acciones. Finalmente, se simplifica hasta llegar a una matriz de 2x4, que significan las acciones que puede tomar el jugador. La arquitectura final se visualiza de la siguiente forma:

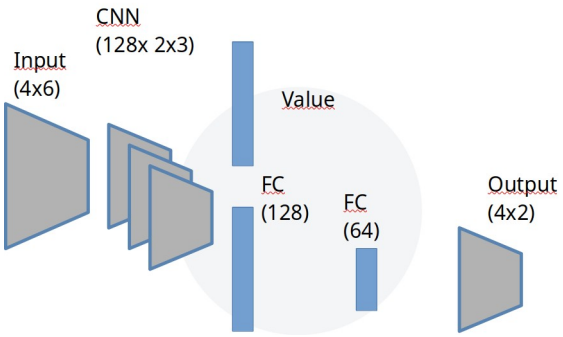


Fig 5. Arquitectura

c. Aprendizaje por refuerzo

Para poder entrenar el modelo, se requiere una metodología de aprendizaje. En este caso, se utiliza el aprendizaje por refuerzo gracias a que permite el uso de un aprendizaje sin conocer los datos de entrada, pero si conociendo cuales son los datos esperados, es decir, meter gol.

El aprendizaje por refuerzo requiere de 3 elementos principales:

1. El premio
2. Metodología de aprendizaje
3. Optimizador

i. El premio

El premio define que se esta haciendo bien, y que no. Un modelo va a optimizar para poder encontrar un premio mayor. Existen 2 formas de otorgar premios: Excéntrico o intrinco, o bien, que el ambiente te otorgue un premio, o que el mismo agente sepa que tan bien lo hace.

En este caso, se puede calcular el premio basado en el ambiente. Se mete gol, se otorga un premio al modelo, le meten gol, se otorga un premio negativo.

Para poder tener premios intermedios, se considera el siguiente modelo matemático:

$$r = x_p \cdot 0.5 + (1 - \|z_p\|) \cdot 0.4 + x_v \cdot 6 + u * 10$$

Donde:

- $x_p \cdot 0.5$ representa que tan cerca esta la bola a su portería, y le da importancia baja.
- $(1 - \|z_p\|) \cdot 0.4$ representa si la bola esta centrada no, y le da importancia mas baja.
- $0.4 + x_v \cdot 6$ representa si la bola se aleja o aproxima a su portería, y le da alta importancia.
- $u * 10$ representa si se ha metido gol (u = -1) o metió gol (u = 1) y le da importancia extrema.

De esta forma, la IA puede entrenar basado en el movimiento de la pelota en todo momento, y si le pega a la pelota puede ver si la acción fue buena o mala.

ii. Metodología de aprendizaje

Para el aprendizaje, se requiere los siguiente:

- Un espacio de acción continuo: ya que los valores es cualquiera dentro de -1 a 1

- Un método que permita calificar que tan bueno es un movimiento a futuro
- Que aprenda basado en premios futuros (esperados)

Existen métodos como Q-learning, pero en este caso no son utilizables, ya que ellos dependen de espacios segmentados, cosa que no se tiene, por lo que PPO es el modelo a usar.

PPO permite utilizar un espacio de acción continua, pero lo mas importante, se puede a adaptar a un modelo de actor-critico, y puede tomar un modelo de cualquier tipo.

Para el uso de aprendizaje, se usa GAE (Generalized Advantage Estimation), que consiste en:

$$R_t = r_t + \gamma \cdot R_{t-1} \qquad A_t = R_t - v_t$$

Donde:

- R = el promedio obtenido del premio
- A = el desfase del criterio contra el premio (esperado – retornado)
- r = el premio otorgado
- v = el valor esperado (proporcionado por el critico)

Una vez obtenido, se toma un valor de seguridad, donde intenta optimizar el modelo en pasos pequeños dentro de un limite seguro. Para cada iteración, se calcula la perdida utilizando el cuadrado promedio de cada elemento de R vs A, y restando la diferencial de los movimientos predichos previamente.

$$loss = mse(A, R) - A\frac{\hat{y}_n}{\hat{y}_p}$$

iii. Optimizador

Como simpatizador del modelo, se escogió Adam, gracias a que todos los resultados dependen del resto (a diferencia de Cross Entroy Validation), y permite evitar tener un sobre entrenamiento.

IV. Ejecución

El modelo fue entrenado durante un periodo de 48 horas con el modelo corriendo en una computadora de 32 Gb de RAM, y un procesador Intel de 16 hilos. Se corrió el modelo y el aprendizaje en la misma computadora, haciendo innecesario el uso de API, pero fue requerido su proceso debido a que no se pudo correr PyTorch usando IronPython.

Durante el entrenamiento, se observo el método de entrenamiento, y se obtuvieron las siguientes observaciones:

a. Al inicio

Al inicio, el modelo mostraba un comportamiento aleatorio, con movimientos mínimos, y sin una estrategia.

Durante esta fase, la IA estaba explorando, por lo que no se ven muchas acciones.

b. Durante el aprendizaje

Una vez entendido que se debe de meter gol al otro lado, los jugadores iniciaron a rotar hacia enfrente, y se intentaron mantener centrados.

Esto permitía que la pelota se fuera al otro lado de la cancha y pudieran obtener su premio.

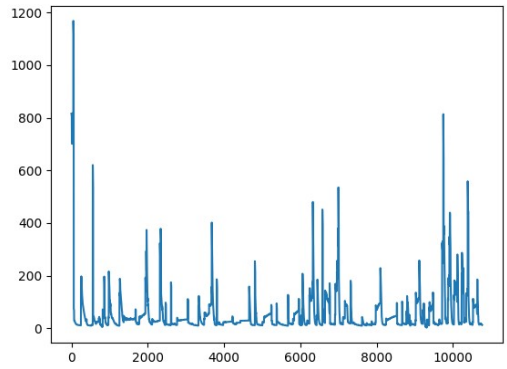
c. Al finalizar

Durante el aprendizaje encontraron que si la pelota al inicio se va de su lado, lo mejor es meter gol en su lado, y en base a ello, poder conseguir un premio lo antes posible.

Este no era el comportamiento deseado, sin embargo se puede mitigar empeorando el premio negativo otorgado por que el otro jugador pueda meter gol.

d. Gráficos

En entrenamiento se puede visualizar como el siguiente grafico:



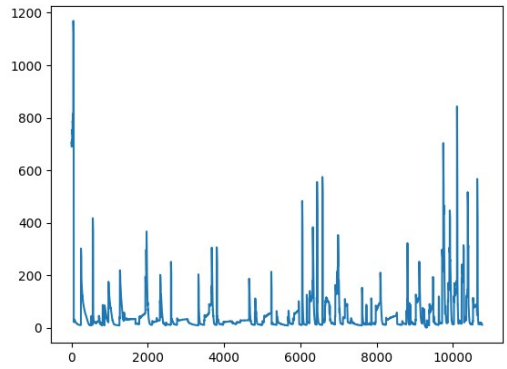


Fig 6. Loss

i. Interpretación

El grafico muestra que el inicio creían saber que estaban haciendo bien el trabajo, sin embargo los premios negativos hacían que su grado de seguridad baje, por lo que se ven picos grandes entre 5 y 100. Una vez avanzado, el progreso, entendió mejor como sirven los premios.

V. Conclusión

a. Optimizaciones

PPO y el uso de Aprendizaje por Refuerzo es algo excepcional. Me encanto experimentar con el modelo. Sin embargo el tiempo que se tarda en ejecutar y entrenar el modelo es mucho. Esto es debido a que encima de que RL es mas lento que un modelo de IA convencional, mi implementación de RL es una donde la mayoría del tiempo se tarda en la comunicación de la simulación con la IA.

En un futuro el usar algo que sea mas rápido o aprender a integrar la IA dentro del mismo modelo usando ML.NET puede significar que el aprendizaje puede correr completamente en GPU y ser mas eficiente.

b. Uso del IA

La IA si es una herramienta muy poderosa, pero poderosa no significa que sea un remplazo de todo. Dado su baja visibilidad, es difícil predecir que es la acción que va a tomar o como es que aprendió. El tiempo de entrenamiento de una Red Neuronal igualmente es alto, y lo más importante es que existen soluciones para todos los elementos.

c. Uso de CNN en RL

El uso de una CNN puede mostrar patrones cuando se tiene una entrada en formato de matriz. Sin embargo no se tiene un control, por lo que no se puede llegar a una conclusión de si es mas eficiente o no. Sin embargo, se muestra que si puede servir una CNN para el control de un juego.

VI. Trabajo a futuro

Los posibles avances que se pueden realizar son:

1. Probar por mas tiempo la IA. La IA no pudo ser entrenada completamente para que pueda generalizar.
2. Eficientemente manejar la comunicación entre IA y juego.
3. Probar solo con capas FC, para visualizar si una CNN ayuda a la interpretación de datos.
4. Probar distintas CNN. Se puede agregar a la arquitectura lo que sigue:
    1. Separar acciones del jugador y bola (introducir movimiento de la bola en un momento futuro)
    2. Crear un cuello de botella en la CNN de 16 x 1 x 3 antes de pasar al FC
5. Crear un mejor simulador. La física a niveles pequeños no sirve de forma correcta. Se puede escalar el simulador par ano tener tan poca precesión.

VII. Referencias

(a) Chatzimparmpas, A., Martins, R. M., & Kerren, A. (2020, March 19). A survey of surveys on the use of visualization for interpreting machine learning models. https://doi.org/10.1177/1473871620904671

(b) Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), Sakheer, Bahrain, 2020, pp. 1-5, doi: 10.1109/ICDABI51230.2020.9325622. keywords: {Reinforcement learning;Training;Long short term memory;Games;Robots;Libraries;Industries;Deep Reinforcement Learning;Autonomous Navigations;Deep Q-Learning;PyGame},

(c) Ilyas, A., Madry, A., Rudolph, L., Tsipras, D., Santurkar, S., & Engstrom, L. (2019, December 19). Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO. https://arxiv.org/pdf/1907.10273

(d) Tai, L., Liu, M. Mobile robots exploration through cnn-based reinforcement learning. *Robot. Biomim.* **3**, 24 (2016). https://doi.org/10.1186/s40638-016-0055-x